Xspec

An X-Ray Spectral Fitting Package

Users' Guide for version 12.9.1

Keith Arnaud, Craig Gordon & Ben Dorman

HEASARC
Astrophysics Science Division
NASA/GSFC
Greenbelt, MD 20771

Jan 2017

Contents

1 Introduction **1**

1.1 New in v12.9.1 2

1.2 How to find out more information 4

1.3 History 5

1.4 Acknowledgements 5

1.5 References 5

2 Spectral Fitting and XSPEC **7**

2.1 Introduction 7

2.2 The Basics of Spectral Fitting 7

2.3 The XSPEC implementation 9

2.4 Fits and Confidence Intervals 11

2.5 A more abstract and generalized approach 12

2.6 XSPEC Data Analysis 14

2.7 References 16

3 XSPEC Overview and Helpful Hints **17**

3.1 Syntax 17
3.2 How to return to the XSPEC12>prompt 17
3.3 Getting Help 18
3.4 Commands 18
3.5 Issuing Commands 19
3.6 Control Commands 19
3.7 Query, chatter and shutting XSPEC up (somewhat) 20
3.8 Scripts and the Save command 21
3.9 Miscellaneous 21
3.10 Data Commands 21
3.11 Model Commands 23
3.12 Fitting Commands 25
3.13 Binning and Grouping data 26
3.14 Plotting Commands 26
3.15 Setting Commands 27
3.16 Breaking With Ctrl-C 28
3.17 Customizing XSPEC 29
3.18 Customizing system-wide 30

4 Walks through XSPEC **31**

4.1 Introduction 31
4.2 Brief Discussion of XSPEC Files 31
4.3 Fitting Models to Data: An Old Example from EXOSAT 32
4.4 Simultaneous Fitting 58

4.5 Multiple Models: a Background Modeling Example 62

4.6 Using XSPEC to Simulate Data: an Example for Chandra 65

4.7 Producing Plots: Modifying the Defaults 69

4.8 Markov Chain Monte Carlo Example 72

4.9 INTEGRAL/SPI: A Walk Through Example 74

4.10 INTEGRAL Specific Command Line Scripts 81

5 XSPEC Commands **85**

5.1 Summary of Commands 85

5.2 Description of Syntax 89

5.3 Control Commands 90

5.3.1 autosave 90

5.3.2 chatter 90

5.3.3 exit, quit 91

5.3.4 help 91

5.3.5 log 92

5.3.6 parallel 93

5.3.7 query 95

5.3.8 save 95

5.3.9 script 96

5.3.10 show 96

5.3.11 syscall 98

5.3.12 tclout 98

5.3.13 tcloutr 104

5.3.14 time 104
5.3.15 undo 104
5.3.16 version 105
5.4 Data Commands 105
5.4.1 arf 105
5.4.2 backgrnd 106
5.4.3 corfile 107
5.4.4 cornorm 108
5.4.5 data 109
5.4.6 diagrsp 113
5.4.7 fakeit 114
5.4.8 ignore 119
5.4.9 notice 120
5.4.10 response 122
5.5 Fit Commands 123
5.5.1 bayes 123
5.5.2 chain 124
5.5.3 error (and rerror) 128
5.5.4 fit 129
5.5.5 freeze (and rfreeze) 131
5.5.6 ftest 131
5.5.7 goodness 132
5.5.8 margin 132

5.5.9 renorm 133

5.5.10 steppar 134

5.5.11 thaw (and rthaw) 135

5.5.12 weight 136

5.6 Model Commands 136

5.6.1 addcomp 136

5.6.2 addline 138

5.6.3 delcomp 139

5.6.4 dummyrsp 139

5.6.5 editmod 142

5.6.6 energies 143

5.6.7 eqwidth 145

5.6.8 flux 146

5.6.9 gain 148

5.6.10 identify 151

5.6.11 initpackage 152

5.6.12 lmod 153

5.6.13 lumin 154

5.6.14 mdefine 155

5.6.15 model (and rmodel) 158

5.6.16 modid 163

5.6.17 newpar (and rnewpar) 163

5.6.18 systematic 168

5.6.19 untie (and runtie) 168

5.7 Plot Commands 169

5.7.1 cpd 169

5.7.2 hardcopy 171

5.7.3 iplot 171

5.7.4 plot 171

5.7.5 setplot 176

5.8 Setting Commands 182

5.8.1 abund 182

5.8.2 cosmo 183

5.8.3 method 184

5.8.4 statistic 186

5.8.5 xsect 187

5.8.6 xset 187

5.9 Tcl Scripts 190

5.9.1 lrt 190

5.9.2 multifake 191

5.9.3 rescalecov 191

5.9.4 simftest 191

5.9.5 writefits 192

6 XSPEC Models **193**

6.1 Alphabetical Summary of Models 193

6.2 Additive Model Components 197

6.2.1 agauss, zagauss: gaussian line profile in wavelength space 197

6.2.2 apec, vapec, vvapec: APEC emission spectrum 198

6.2.3 atable: tabulated additive model 200

6.2.4 bapec, bvapec, bvvapec: velocity broadened APEC thermal plasma model 200

6.2.5 bbody, zbbody: blackbody 202

6.2.6 bbodyrad: blackbody spectrum, area normalized 203

6.2.7 bexrav: reflected e-folded broken power law, neutral medium 203

6.2.8 bexriv: reflected e-folded broken power law, ionized medium 204

6.2.9 bknpower: broken power law 205

6.2.10 bkn2pow: broken power law, 2 break energies 206

6.2.11 bmc: Comptonization by relativistic matter 206

6.2.12 bremss, vbremss, zbremss: thermal bremsstrahlung 207

6.2.13 btapec, bvtapec, bvvtapec: velocity broadened APEC emission spectrum with separate continuum and line temperatures 208

6.2.14 c6mekl, c6vmekl, c6pmekl, c6pvmkl: differential emission measure using Chebyshev representations with multi-temperature mekal 210

6.2.15 carbatm: Nonmagnetic carbon atmosphere of a neutron star 211

6.2.16 cemekl, cevmkl: plasma emission, multi-temperature using mekal 211

6.2.17 cflow: cooling flow 212

6.2.18 compbb: Comptonization, black body 213

6.2.19 compLS: Comptonization, Lamb & Sanford 213

6.2.20 compmag: Thermal and bulk Comptonization for cylindrical accretion onto the polar cap of a magnetized neutron star 213

6.2.21 compPS: Comptonization, Poutanen & Svenson 214

6.2.22 compST: Comptonization, Sunyaev & Titarchuk 218

6.2.23 comptb: Thermal and bulk Comptonization of a seed blackbody-like spectrum 218

6.2.24 compTT: Comptonization, Titarchuk 219

6.2.25 cplinear: a non-physical piecewise-linear model for low count background spectra. 219

6.2.26 cutoffpl: power law, high energy exponential cutoff 220

6.2.27 disk: accretion disk, black body 221

6.2.28 diskbb: accretion disk, multi-black body components . . . 221

6.2.29 diskir: Irradiated inner and outer disk 221

6.2.30 diskline: accretion disk line emission, relativistic 222

6.2.31 diskm: accretion disk with gas pressure viscosity 223

6.2.32 disko: accretion disk, inner, radiation pressure viscosity . 223

6.2.33 diskpbb: accretion disk, power-law dependence for T(r) . 223

6.2.34 diskpn: accretion disk, black hole, black body 224

6.2.35 eplogpar: log-parabolic blazar model with νF_ν normalization 224

6.2.36 Eqpair, eqtherm, compth: Paolo Coppi's hybrid (thermal/non-thermal) hot plasma emission models. 225

6.2.37 equil, vequil: collisional plasma, ionization equilibrium . . 226

6.2.38 expdec: exponential decay 228

6.2.39 ezdiskbb: multiple blackbody disk model with zero-torque inner boundary 228

6.2.40 gadem, vgadem: plasma emission, multi-temperature with gaussian distribution of emission measure. 228

6.2.41 gauss, zgauss: gaussian line profile 229

6.2.42 gnei, vgnei, vvgnei: collisional plasma, non-equilibrium, temperature evolution 230

6.2.43 grad: accretion disk, Schwarzschild black hole 232

6.2.44 grbm: gamma-ray burst continuum 232

6.2.45 hatm: Nonmagnetic hydrogen atmosphere of a neutron star 233

6.2.46 kerrbb: multi-temperature blackbody model for thin accretion disk around a Kerr black hole 234

6.2.47 kerrd: optically thick accretion disk around a Kerr black hole 235

6.2.48 kerrdisk: accretion disk line emission with BH spin as free parameter 235

6.2.49 laor: accretion disk, black hole emission line 236

6.2.50 laor2: accretion disk with broken-power law emissivity profile, black hole emission line 236

6.2.51 logpar: log-parabolic blazar model 236

6.2.52 lorentz: lorentz line profile 237

6.2.53 meka, vmeka: emission, hot diffuse gas (Mewe-Gronenschild) 237

6.2.54 mekal, vmekal: emission, hot diffuse gas (Mewe-Gronenschild-Kaastra) 238

6.2.55 mkcflow, vmcflow: cooling flow, mekal 239

6.2.56 nei, vnei, vvnei: collisional plasma, non-equilibrium, constant temperature 240

6.2.57 nlapec: continuum-only APEC emission spectrum 242

6.2.58 npshock, vnpshock, vvnpshock: shocked plasma, plane parallel, separate ion, electron temperatures 243

6.2.59 nsa: neutron star atmosphere 245

6.2.60 nsagrav: NS H atmosphere model for different g 246

6.2.61 nsatmos: NS Hydrogen Atmosphere model with electron conduction and self-irradiation 247

6.2.62 nsmax: Neutron Star Magnetic Atmosphere 247

6.2.63 nsmaxg: Neutron Star with a Magnetic Atmosphere . . . 249

6.2.64 nsx: neutron star with a non-magnetic atmosphere 251

6.2.65 nteea: non-thermal pair plasma 251

6.2.66 Nthcomp: Thermally comptonized continuum 252

6.2.67 Optxagnf, optxagn: Colour temperature corrected disc and energetically coupled Comptonisation model for AGN. 253

6.2.68 pegpwrlw: power law, pegged normalization 255

6.2.69 pexmon: neutral Compton reflection with self-consistent Fe and Ni lines. 256

6.2.70 pexrav: reflected powerlaw, neutral medium 257

6.2.71 pexriv: reflected powerlaw, ionized medium 257

6.2.72 plcabs: powerlaw observed through dense, cold matter . . 258

6.2.73 posm: positronium continuum 259

6.2.74 powerlaw, zpowerlw: power law photon spectrum 259

6.2.75 pshock, vpshock, vvpshock: plane-parallel shocked plasma, constant temperature 260

6.2.76 raymond, vraymond: emission, hot diffuse gas, Raymond-Smith 262

6.2.77 redge: emission, recombination edge 263

6.2.78 refsch: reflected power law from ionized accretion disk . . 263

6.2.79 rnei, vrnei, vvrnei: non-equilibrium recombining collisional plasma . . . 264

6.2.80 sedov, vsedov, vvsedov: sedov model, separate ion, electron temperatures . . . 266

6.2.81 sirf: self-irradiated funnel . . . 268

6.2.82 slimbh: Stationary slim accretion disk . . . 268

6.2.83 smaug: optically-thin, spherically-symmetric thermal plasma. 269

6.2.84 snapec: galaxy cluster spectrum using SN yields . . . 271

6.2.85 srcut: synchrotron spectrum, cutoff power law . . . 274

6.2.86 sresc: synchrotron spectrum, cut off by particle escape . . . 275

6.2.87 step: step function convolved with gaussian . . . 275

6.2.88 tapec, vtapec, vvtapec: APEC emission spectrum with separate continuum and line temperatures . . . 275

6.2.89 voigt: Voigt line profile . . . 278

6.3 Multiplicative Model Components . . . 278

6.3.1 absori: ionized absorber . . . 278

6.3.2 acisabs: Chandra ACIS q.e. decay . . . 279

6.3.3 cabs: Optically-thin Compton scattering. . . . 280

6.3.4 constant: energy-independent factor . . . 280

6.3.5 cyclabs: absorption line, cyclotron . . . 280

6.3.6 dust: dust scattering . . . 281

6.3.7 edge, zedge: absorption edge . . . 281

6.3.8 etable: exponential tabular model . . . 281

6.3.9 expabs: exponential roll-off at low E . . . 282

6.3.10 expfac: exponential modification . . . 282

6.3.11 gabs: gaussian absorption line 283

6.3.12 heilin: Voigt absorption profiles for He I series 283

6.3.13 highecut, zhighect: high-energy cutoff 283

6.3.14 hrefl: reflection model 284

6.3.15 ismabs: A high resolution ISM absorption model with variable columns for individual ions 285

6.3.16 lyman: Voigt absorption profiles for H I or He II Lyman series 287

6.3.17 mtable: multiplicative tabular model 287

6.3.18 notch: absorption line, notch 287

6.3.19 pcfabs, zpcfabs: partial covering fraction absorption . . . 288

6.3.20 phabs, vphabs, zphabs, zvphabs: photoelectric absorption 288

6.3.21 plabs: power law absorption 289

6.3.22 pwab: power-law distribution of neutral absorbers 290

6.3.23 recorn: change correction norm for a spectrum 290

6.3.24 redden: interstellar extinction 290

6.3.25 smedge: smeared edge 291

6.3.26 spexpcut: super-exponential cutoff absorption 291

6.3.27 spline: spline modification 292

6.3.28 SSSice: Einstein SSS ice absorption 292

6.3.29 swind1: absorption by partially ionized material with large velocity shear 292

6.3.30 tbabs, ztbabs, tbfeo, tbgas, tbgrain, tbpcf, tbvarabs, tbrel: ISM grain absorption 293

6.3.31 uvred: interstellar extinction, Seaton Law 295

6.3.32 varabs, zvarabs: photoelectric absorption 295

6.3.33 wabs, zwabs: photoelectric absorption, Wisconsin cross-sections 296

6.3.34 wndabs, zwndabs: photo-electric absorption, warm absorber 296

6.3.35 xion: reflected spectrum of photo-ionized accretion disk/ring 297

6.3.36 xscat: dust scattering 299

6.3.37 zbabs: EUV ISM attenuation 299

6.3.38 zdust: extinction by dust grains 300

6.3.39 zigm: UV/Optical attenuation by the intergalactic medium 300

6.3.40 zredden: redshifted version of redden 301

6.3.41 zsmdust: extinction by dust grains in starburst galaxies . 301

6.3.42 zvfeabs: photoelectric absorption with free Fe edge energy 301

6.3.43 zxipcf: partial covering absorption by partially ionized material 302

6.4 Convolution Model Components 302

6.4.1 cflux: calculate flux 302

6.4.2 clumin: calculate luminosity 303

6.4.3 cpflux: calculate photon flux 304

6.4.4 gsmooth: gaussian smoothing 305

6.4.5 ireflect: reflection from ionized material 305

6.4.6 kdblur: convolve with the laor model shape 307

6.4.7 kdblur2: convolve with the laor2 model shape 307

6.4.8 kerrconv: accretion disk line shape with BH spin as free parameter 307

6.4.9 lsmooth: lorentzian smoothing 308

6.4.10 partcov: partial covering 308

6.4.11 rdblur: convolve with the diskline model shape 309

6.4.12 reflect: reflection from neutral material 309

6.4.13 rfxconv: angle-dependent reflection from an ionized disk . 310

6.4.14 simpl: comptonization of a seed spectrum 311

6.4.15 xilconv: angle-dependent reflection from an ionized disk . 312

6.4.16 vashift: velocity shift an additive model 313

6.4.17 vmshift: velocity shift a multiplicative model 314

6.4.18 zashift: redshift an additive model 314

6.4.19 zmshift: redshift a multiplicative model 315

6.5 Pile-Up Model Components 315

6.5.1 pileup: CCD pile-up model for Chandra 315

6.6 Mixing Model Components 316

6.6.1 ascac: ASCA surface brightness model 317

6.6.2 projct: project 3-D ellipsoidal shells onto 2-D elliptical annuli 317

6.6.3 suzpsf: suzaku surface brightness model 319

6.6.4 xmmpsf: xmm surface brightness model 320

A The User Interface **323**

A.1 Introduction 323

A.2 XSPEC and tcl/tk 323

A.3 A note on command processing 324

A.4 Command Recall/Editing 324

A.5 Logging 325

A.6 Command Completion 326

A.7 Unknown Procedure 327

A.8 Aliases 327

A.9 Initialization Script 328

A.10 XSPEC Command Result 328

A.11 Script Files 328

A.12 Unix Shell Commands 330

A.13 Writing Custom XSPEC commands 331

A.14 Scripting commands that prompt the user 333

A.15 Script Example 334

B Statistics in XSPEC **337**

B.1 Introduction 337

B.2 Parameter Estimation 338

B.3 Parameter confidence regions 344

B.4 Goodness-of-fit 346

C Adding Models to XSPEC **351**

C.1 Table models 351

C.2 Loading a new model function 352

C.3 Writing a new model function 353

C.4 Third-Party Libraries In Local Models Build 358

C.5 Writing new mixing models 359

D Overview of PLT **361**

D.1 Getting started with PLT 362

D.2 PLT Command summary 362

E Associated Programs **365**

E.1 Introduction 365

E.2 HEAsoft reading tasks 365

E.3 HEAsoft manipulation tasks 366

E.4 HEAsoft subroutines 366

F Using the XSPEC Models Library in Other Programs **367**

F.1 Calling Model Functions From C And Fortran 367

F.2 Interface Routines 368

F.3 Initializing the Models Library 370

F.4 Building with the Models Library 370

G Adding a Custom Chain Proposal Algorithm **371**

G.1 The `randomize.dat` Initialization File 372

G.2 Writing a Chain Proposal Class 373

G.3 Building and Loading the Proposal Class Library 375

H Changes between v11 and v12 **377**

H.1 Integral Spectrometer/Coded Mask Instrument Support 379

H.2 Current Exclusions 379

I Older Release Notes **381**

I.1 v12.9.0 Jul 2015 381

I.2 V12.8.2 Jul 2014 384

I.3 V12.8.1 Aug 2013 385

I.4 v12.8.0 Dec 2012 387

I.5 v12.7.1 March 2012 389

I.6 v12.7.0 May 2011 391

I.7 v12.6.0 March 2010 393

I.8 v12.5.1 Aug. 2009 395

Chapter 1

Introduction

XSPEC is a command-driven, interactive, X-ray spectral-fitting program, designed to be completely detector-independent so that it can be used for any spectrometer. XSPEC has been used to analyze data from HEAO-1 A2, Einstein Observatory, EXOSAT, Ginga, ROSAT, BBXRT, ASCA, CGRO, IUE, RXTE, Chandra, XMM-Newton, Integral/SPI, Swift and Suzaku. There now almost 8000 papers listed on ADS which mention XSPEC.

This manual describes XSPEC v12, which is available on Linux (gcc 3.2.2 and later), MacOSX/Darwin (gcc 3.3 and later) , Solaris (2.6-9; WS6.0 and later). New users are advised to read Chapter 2, which introduces spectral fitting and the XSPEC approach, Chapter 3, which gives an overview of the program commands, and Chapter 4, which contains walkthroughs of XSPEC sessions. They should then experiment with XSPEC and, if necessary, look up individual commands in Chapter 5 , or descriptions of the spectral models in use, in Chapter 6.

The user interface for XSPEC, which employs a tcl scripting shell and the XSPEC parser are described in Appendix A. The statistical methods used within XSPEC are described in Appendix B. Users interested in adding their own models can read how to do so in Appendix C. Appendix D describes the PLT plotting package which XSPEC currently uses for graphical output. Some of the tools (FTOOLS, programs, scripts) that can operate on XSPEC files are listed in Appendix E. The XSPEC model library can be linked into other programs following the instructions in Appendix F. Appendix G describes how to use your own proposal distribution for Markov Chain Monte Carlo. Appendix H includes some notes on the changes between XSPEC v11 and v12. Finally, Appendix I lists release notes for older versions of v12.

1.1 New in v12.9.1

New features

- New Models:

carbatm	NS non-magnetic C atmosphere
clumin	calculates luminosity
hatm	NS non-magnetic H atmosphere
ismabs	High-resolution ISM absorption
rfxconv	reflection from ionized disk
slimbh	slim accretion disk
snapec	galaxy cluster spectrum using SN yields
(b)(v)(v)tapec	apec model with different temperatures for continuum and emission lines
tbfeo	tbabs model with O and Fe only variable
tbgas	tbabs model with no grains
tbpcf	tbabs model with partial covering
tbrel	tbabs model that allows negative columns
vashift	velocity shift an additive component
vmshift	velocity shift a multiplicative component
voigt	emission line with a Voigt profile
xilconv	reflection from ionized disk
xscat	dust scattering out of the beam

- Updated Models:
  - **apec** models updated to AtomDB 3.0.7
  - **nei** models updated to AtomDB 3.0.4 eigen files
  - The **apec** code now has the ability to add new line shapes for broadening although at present only a single Gaussian line shape is operational.
  - A new **xset** option APECMINFLUX can be used to specify a minimum line flux below which lines are not broadened. Also the maximum flux in any line in the **apec** model is saved as `apecmaxlineflux` and this can be recovered using **tcloutr** `modkeyval apecmaxlineflux`. At the moment both this and APECMINFLUX do not include the 1e14 normalization factor and any time dilation correction.
  - The code for the various CEI models was rationalized and there is a new C++ routine calcMultiTempPlasma which can be used by other models. The old Fortran sumdem routine remains as a wrapper for old models.
  - **tbabs** model code has been updated. The old version is available by using **xset** TBABSVERSION 1.

  - Updated **nsmaxg** model with new tabulated model files.
  - Increased the speed for accessing values from table models.
  - Modified table model code to support new keywords LOELIMIT and HIELIMIT to specify model values to be taken below and above the tabulated energies.

- Plotting:
  - Added low and high energy limits to **setplot id** so lines are only shown for the specified energy range.
  - Added **setplot** `(no)contimage` to turn on (off) the background image in contour plots.
  - The keV to Angstrom conversion value stored in Numerics.h has been updated to that from CODATA 2014.

- Fitting:
  - Added parallelization to chain walker calculations and to the **goodness** simulations.
  - The **error** command can now process parameters from more than 1 model in a single call.
  - In the various cstat statistics any correction file spectrum is now taken account. This does implicitly assume that the correction file has Poisson statistics.
  - The Tcl script `lrt.tcl` now uses the **fakeit** `nowrite` option so intermediate faked files are not written out. Also, now works for multiple data groups.

- Input/Output:
  - Increased the precision of the reported output from the **tclout** command options.
  - Increased the allowed flux (counts) to beyond the 4-byte limit when running **fakeit** with counting statistics selected.
  - New **tclout** option `fileinfo` writes the value of the keyword given as the fileinfo argument. e.g. `tclout fileinfo detnam` returns the value of the `detnam` keyword in the spectrum extension.

- Xspec.init file:
  - Added `PARALLEL` to set default parallelization options
  - Added `NEI_VERSION` to set default version for NEI models
  - Added `CONTOUR_IMAGE` to set default for adding image to contour plots
  - Changed default cross-sections from `bcmc` to `vern`

- The manual has now been rewritten in LaTeX. The appendix on statistics has been expanded. Cross-referencing has been improved.

Bug fixes

All bug fixes to v12.9.0 released as patches are included in v12.9.1. In addition the following problems have been corrected:

- A previous revision to the **goodness** command removed the requirement that the fit must be in a valid state at the start, however it was incorrectly marking the fit as being in a valid state at the end.
- When re-reading a previously loaded table model file, XSPEC was not correctly recognizing whether or not parameter error values existed.
- A spectrum's displayed net count rate variance needs to be updated when the fit weighting method gets changed.

1.2 How to find out more information

XSPEC is distributed and maintained under the aegis of the GSFC High Energy Astrophysics Science Archival Research Center (HEASARC). It can be downloaded as part of HEAsoft from
http://heasarc.gsfc.nasa.gov/docs/software/lheasoft/download.html

XSPEC is available either as binaries or source. We recommend downloading the source and compiling locally to avoid potential system library conflicts and allow installation of any patches we release. If you have any problems compiling or running XSPEC please e-mail us at xspec12@athena.gsfc.nasa.gov

The XSPEC Web page comprises links to the anonymous ftp site, a Web version of the manual, and several useful documents, including a list of known bugs. The Web address is
http://xspec.gsfc.nasa.gov/
with the list of issues and available patches at
http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/bugs.html
and additional models which can be added at
http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/newmodels.html
Further useful information can be found on the XSPEC Wiki at
https://astrophysics.gsfc.nasa.gov/XSPECwiki/XSPECPage
and via the XSPEC Facebook group.

1.3 History

The first version of XSPEC was written in 1983 at the Institute of Astronomy, Cambridge, under VAX/VMS by Rick Shafer. It was written to perform spectral analysis of data from the ESA EXOSAT X-ray observatory, which was launched that year. XSPEC is a descendant of a series of spectral-fitting programs written at NASA/Goddard Space Flight Center for the OSO-8, HEAO-1 and EO missions. The initial design was the fruit of many discussions between Rick Shafer and Andy Szymkowiak.

Rick Shafer subsequently joined the EXOSAT group, where he enhanced the VAX/VMS version in collaboration with Frank Haberl. Meanwhile, Keith Arnaud ported XSPEC to a Sun/UNIX operating system. The two implementations of XSPEC diverged for several years until a determined and coordinated effort was made to recover a single version. The results of that effort was XSPECv6, described in the first edition of this manual. Subsequent editions have covered later versions of the program.

An extensive re-engineering effort was started in the fall of 1998 to improve long-term maintainability, allow significant new features to be added, and support the analysis of spectra from coded-mask instruments.

1.4 Acknowledgements

This project would not have been possible without ignoring the advice of far more people than can be mentioned here. We would like to thank all our colleagues for their suggestions, bug reports, and (especially) source code. The GSFC X-ray astronomy group are particularly thanked for their patience exhibited while functioning as the beta test site for new releases. The initial development of XSPEC was funded by a Royal Society grant to Prof. Andy Fabian and subsequent development was partially funded by the European Space Agency's EXOSAT project and is now funded by the HEASARC at NASA/GSFC.

1.5 References

- Arnaud, K.A., 1996, Astronomical Data Analysis Software and Systems V, eds. G. Jacoby and J. Barnes,p17, ASP Conf. Series volume 101.
- Dorman, B., and Arnaud, K. A. 2001, Astronomical Data Analysis Software and Systems X eds. F.R. Harnden, Jr., F.A. Primini, and H. E. Payne, vol. 238, p. 415

- Dorman, B., Arnaud, K. A., and Gordon, C. A. XSPEC12: Object-Oriented X-Ray Data Analysis, AAS HEAD meeting No. 35, #22.10

Chapter 2

Spectral Fitting and XSPEC

2.1 Introduction

This chapter comprises a brief description of the basics of spectral fitting, their application in XSPEC, and some helpful hints on how to approach particular problems. We then provide more details on the way XSPEC provides flexibility in its approach to the minimization problem. We also describe the data formats accepted.

2.2 The Basics of Spectral Fitting

Although we use a spectrometer to measure the spectrum of a source, what the spectrometer obtains is not the actual spectrum, but rather photon counts (C) within specific instrument channels, (I). This observed spectrum is related to the actual spectrum of the source ($f(E)$) by:

$$C(I) = \int f(E)R(I,E)dE \tag{2.1}$$

Where $R(I,E)$ is the instrumental response and is proportional to the probability that an incoming photon of energy E will be detected in channel I. Ideally, then, we would like to determine the actual spectrum of a source, $f(E)$, by

inverting this equation, thus deriving $f(E)$ for a given set of $C(I)$. Regrettably, this is not possible in general, as such inversions tend to be non-unique and unstable to small changes in $C(I)$. (For examples of attempts to circumvent these problems see Blissett & Cruise 1979; Kahn & Blissett 1980; Loredo & Epstein 1989).

The usual alternative is to choose a model spectrum, $f(E)$, that can be described in terms of a few parameters (i.e., $f(E, p1, p2, ...)$), and match, or "fit" it to the data obtained by the spectrometer. For each $f(E)$, a predicted count spectrum ($C_p(I)$) is calculated and compared to the observed data ($C(I)$). Then a "fit statistic" is computed from the comparison and used to judge whether the model spectrum "fits" the data obtained by the spectrometer.

The model parameters then are varied to find the parameter values that give the most desirable fit statistic. These values are referred to as the best-fit parameters. The model spectrum, $f_b(E)$, made up of the best-fit parameters is considered to be the best-fit model.

The most common fit statistic in use for determining the "best-fit" model is χ^2, defined as follows:

$$\chi^2 = \sum \frac{(C(I) - C_p(I))^2}{(\sigma(I))^2} \tag{2.2}$$

where $\sigma(I)$ is the (generally unknown) error for channel I (e.g., if $C(I)$ are counts then $\sigma(I)$ is usually estimated by $\sqrt{C(I)}$; see e.g. Wheaton et.al. 1995 for other possibilities).

Once a "best-fit" model is obtained, one must ask two questions:

1. How confident can one be that the observed $C(I)$ can have been produced by the best-fit model $f_b(E)$? The answer to this question is known as the "goodness-of-fit" of the model. The χ^2 statistic provides a well-known-goodness-of-fit criterion for a given number of degrees of freedom (ν, which is calculated as the number of channels minus the number of model parameters) and for a given confidence level. If χ^2 exceeds a critical value (tabulated in many statistics texts) one can conclude that $f_b(E)$ is *not* an adequate model for $C(I)$. As a general rule, one wants the "reduced χ^2" ($\sim \chi^2/\nu$) to be approximately equal to one (i.e. $\chi^2 \sim \nu$). A reduced χ^2 that is much greater than one indicates a poor fit, while a reduced χ^2 that is much less than one indicates that the errors on the data have been over-estimated. Even if the best-fit model ($f_b(E)$) does pass the "goodness-of-fit" test, one still cannot say that $f_b(E)$ is the only acceptable model. For example, if the data used in the fit are not particularly

good, one may be able to find many different models for which adequate fits can be found. In such a case, the choice of the correct model to fit is a matter of scientific judgment.

2. For a given best-fit parameter ($p1$), what is the range of values within which one can be confident the true value of the parameter lies? The answer to this question is the "confidence interval" for the parameter. The confidence interval for a given parameter is computed by varying the parameter value until the χ^2 increases by a particular amount above the minimum, or "best-fit" value. The amount that the χ^2 is allowed to increase (also referred to as the critical $\Delta\chi^2$) depends on the confidence level one requires, and on the number of parameters whose confidence space is being calculated. The critical for common cases are given in the following table (from Avni, 1976):

Confidence	Parameters		
	1	2	3
0.68	1.00	2.30	3.50
0.90	2.71	4.61	6.25
0.99	6.63	9.21	11.30

2.3 The XSPEC implementation

To summarize the preceding section, the main components of spectral fitting are as follows:

- A set of one or more observed spectra $D(I)$ with background measurements $B(I)$ where available
- The corresponding instrumental responses $R(I, E)$
- A set of model spectra $M(E)$

These components are used in the following manner:

- Choose a parameterized model which is thought to represent the actual spectrum of the source.
- Choose values for the model parameters.
- Based on the parameter values given, predict the count spectrum that would be detected by the spectrometer in a given channel for such a model.
- Compare the predicted spectrum to the spectrum actually obtained by the instrument.

- Manipulate the values of the parameters of the model until the best fit between the theoretical model and the observed data is found.
- Then calculate the "goodness" of the fit to determine how well the model explains the observed data, and calculate the confidence intervals for the model's parameters.

This section describes how XSPEC performs these tasks.

C(I): The Observed Spectrum

To obtain each observed spectrum, $C(I)$, XSPEC uses two files: the data (spectrum) file, containing $D(I)$, and the background file, containing $B(I)$. The data file tells XSPEC how many total photon counts were detected by the instrument in a given channel. XSPEC then uses the background file to derive the set of background-subtracted spectra C(I) in units of counts per second. The background-subtracted count rate is given by, for each spectrum:

$$C(I) = \frac{D(I)}{a_{D(I)} t_D} - \frac{b_{D(I)}}{b_{B(I)}} \frac{B(I)}{a_{b(I)} t_B} \tag{2.3}$$

where $D(I)$ and $B(I)$ are the counts in the data and background files; t_D and t_B are the exposure times in the data and background files; $b_{D(I)}$ and $b_{B(I)}$, $a_{D(I)}$ and $a_{B(I)}$ are the background and area scaling values from the spectrum and background respectively, which together refer the background flux to the same area as the observation as necessary. When this is done, XSPEC has an observed spectrum to which the model spectrum can be fit.

R(I,E): The Instrumental Response

Before XSPEC can take a set of parameter values and predict the spectrum that would be detected by a given instrument, XSPEC must know the specific characteristics of the instrument. This information is known as the detector response. Recall that for each spectrum the response $R(I, E)$ is proportional to the probability that an incoming photon of energy E will be detected in channel I. As such, the response is a continuous function of E. This continuous function is converted to a discrete function by the creator of a response matrix who defines the energy ranges E_j such that:

$$R_D(I, J) = \frac{\int_{E_{J-1}}^{E_J} R(I, E) dE}{E_J - E_{J-1}} \tag{2.4}$$

XSPEC reads both the energy ranges, E_J, and the response matrix $R_D(I, J)$ from a response file in a compressed format that only stores non-zero elements.

XSPEC also includes an option to use an auxiliary response file, which contains an array $A_D(J)$ that is multiplied into $R_D(I, J)$ as follows:

$$R_D(I, J) \rightarrow R_D(I, J) \times A_D(J) \tag{2.5}$$

This array is designed to represent the efficiency of the detector with the response file representing a normalized Redistribution Matrix Function, or RMF. Conventionally, the response is in units of cm2.

M(E): The Model Spectrum

The model spectrum, $M(E)$, is calculated within XSPEC using the energy ranges defined by the response file :

$$M_D(J) = \int_{E_{J-1}}^{E_J} M(E)dE \tag{2.6}$$

and is in units of photons/cm^2/s. XSPEC allows the construction of composite models consisting of additive components representing X-ray sources (e.g., power-laws, blackbodies, and so forth), multiplicative components, which modify additive components by an energy-dependent factor (e.g., photoelectric absorption, edges, ...). Convolution and mixing models can then perform sophisticated operations on the result. Models are defined in algebraic notation.

For example, the following expression:

```
phabs (power + phabs (bbody))
```

defines an absorbed blackbody, phabs(bbody), added to a power-law, power. The result then is modified by another absorption component, phabs. For a list of available models, see Chapter 6.

2.4 Fits and Confidence Intervals

Once data have been read in and a model defined, XSPEC uses a fitting algorithm to find the best-fit values of the model parameter. The default is a modified Levenberg-Marquardt algorithm (based on CURFIT from Bevington, 1969). The algorithm used is local rather than global, so be aware that it is possible for the fitting process to get stuck in a local minimum and not find the

global best-fit. The process also goes much faster (and is more likely to find the true minimum) if the initial model parameters are set to sensible values.

The Levenberg-Marquardt algorithm relies on XSPEC calculating the 2nd derivatives of the fit statistic with respect to the model parameters. By default these are calculated analytically, with the assumption that the 2nd derivatives of the model itself may be ignored. This can be changed by setting the USE_NUMERICAL_DIFFERENTIATION flag to "true" in the Xspec.init initialization file, in which case XSPEC will perform numerical calculations of the derivatives (which are slower).

At the end of a fit, XSPEC will write out the best-fit parameter values, along with estimated confidence intervals. These confidence intervals are one sigma and are calculated from the second derivatives of the fit statistic with respect to the model parameters at the best-fit. These confidence intervals are not reliable and should be used for indicative purposes only.

XSPEC has a separate command (**error** or **uncertain**) to derive confidence intervals for one interesting parameter, which it does by fixing the parameter of interest at a particular value and fitting for all the other parameters. New values of the parameter of interest are chosen until the appropriate delta-statistic value is obtained. XSPEC uses a bracketing algorithm followed by an iterative cubic interpolation to find the parameter value at each end of the confidence interval.

To compute confidence regions for several parameters at a time, XSPEC can run a grid on these parameters. XSPEC also will display a contour plot of the confidence regions of any two parameters.

2.5 A more abstract and generalized approach

The sections above provide a simple characterization of the problem. XSPEC actually operates at a more abstract level and considers the following:

Given a set of spectra $C(I)$, each supplied as a function of detector channels, a set of theoretical models $\{M(E)_j\}$ each expressed in terms of a vector of energies together with a set of functions $\{R(I,E)_j\}$ that map channels to energies, minimize an objective function S of C, $\{R(I,E)_j\}$, $\{M(E)_j\}$ using a fitting algorithm, i.e.

$$S = S(C, \sum_j M_j \times R_j) \tag{2.7}$$

In the default case, this reduces to the specific expression for χ^2fitting of a single

source

$$S = \chi^2 = \sum_i (C_i - R_i \times M_i)^2 \tag{2.8}$$

where i runs over all of the channels in all of the spectra being fitted, and using the Levenberg-Marquadt algorithm to perform the fitting.

This differs from the previous formulation in that the operations that control the fitting process make fewer assumptions about how the data are formatted, what function is being minimized, and which algorithm is being employed. At the calculation level, XSPEC requires spectra, backgrounds, responses and models, but places fewer constraints as to how they are represented on disk and how they are combined to compute the objective function (statistic). This has immediate implications for the extension of XSPEC analysis to future missions.

New data formats can be implemented independently of the existing code, so that they may be loaded during program execution. The "data format" includes the specification not only of the files on disk but how they combine with models.

Multiple sources may be extracted from a spectrum. For example, it generalizes the fitting problem to minimizing, in the case of the χ^2 statistic,

$$\chi^2 = \sum_i (C_i - \sum_j R_{ij} \times M_j)^2 \tag{2.9}$$

and j runs over one or more models. This allows the analysis of, for instance, coded aperture data where multiple sources may be spatially resolved.

Responses, which abstractly represent a mapping from the theoretical energy space of the model to the detector channel space, may be represented in new ways. For example, the INTEGRAL/SPI responses are implemented as a linear superposition of 3 (fixed) components.

Instead of explicitly combining responses and models through convolution XSPEC places no prior constraint on how this combination is implemented. For example, analysis of data collected by future large detectors might take advantage of the form of the instrumental response by decomposing the response into components of different frequency.

Other differences of approach are in the selection of the statistic of the techniques used for deriving the solution. Statistics and fitting methods may be added to XSPEC at execution time rather than at installation time, so that the analysis package as a whole may more easily keep apace of new techniques.

2.6 XSPEC Data Analysis

XSPEC is designed to support multiple input data formats. Support for the earlier SF and Einstein FITS formats are removed. Support for ASCII data is planned, which will allow XSPEC to analyze spectra from other wavelength regions (optical, radio) transparently to the user.

OGIP Data

The OGIP data format both for single spectrum files (Type I) and multiple spectrum files (Type II) is fully supported. These files can be created and manipulated with programs described in Appendix E and the provided links. The programs are described more fully in George et al. 1992. (the directories below refer to the HEAsoft distribution).

- Spectral Data: callib/src/gen/rdpha2.f, wtpha3.f
- Responses: callib/src/gen rdebd4.f, rdrmf5.f, wtebd4.f, wtrmf5.f. The "rmf" programs read and write the RMF extension, while the "ebd" programs write an extension called EBOUNDS that contains nominal energies for the detector channels.
- Auxiliary Responses: callib/src/gen rdarf1.f, and wtarf1.f

INTEGRAL/SPI Data

XSPEC also includes an add-in module to read and simulate INTEGRAL/SPI data, which can be loaded by the user on demand. The INTEGRAL/SPI datasets are similar to OGIP Type II, but contain an additional FITS extension that stores information on the multiple files used to construct the responses.

The INTEGRAL Spectrometer (SPI) is a coded-mask telescope, with a 19-element Germanium detector array. The Spectral resolution is 500, and the angular resolution is 3deg. Unlike focusing instruments however, the detected photons are not directionally tagged, and a statistical analysis procedure, using for example cross-correlation techniques, must be employed to reconstruct an image. The description of the XSPEC analysis approach which follows assumes that an image reconstruction has already been performed; see the SPIROS utility within the INTEGRAL offline software analysis package (OSA), OR, the positions on the sky of all sources to be analyzed are already known (which is often the case). Those unfamiliar with INTEGRAL data analysis should refer to the OSA documentation. Thus, the INTEGRAL/SPI analysis chain must be run up to the event binning level [if the field of view (FoV) source content is known, e.g. from published catalogs, or from IBIS image analysis], or the image

reconstruction level. SPIHIST should be run selecting the "PHA" output option, and selecting detectors 0-18. This will produce an OGIP standard type-II PHA spectral file, which contains multiple, detector count spectra. In addition, the SPIARF procedure should be run once for each source to be analyzed, plus one additional time to produce a special response for analysis of the instrumental background. If this is done correctly, and in the proper sequence, SPIARF will create a table in the PHA-II spectral file, which will associate each spectrum with the appropriate set of response matrices. The response matrices are then automatically loaded into XSPEC upon execution of the data command in a manner very transparent to the user. You will also need to run SPIRMF (unless you have opted to use the default energy bins of the template SPI RMFs). Finally, you will need to run the FTOOL SPIBKG_INIT. Each of these utilities - SPIHIST, SPIARF, SPIRMF and SPIBKG_INIT - are documented elsewhere, either in the INTEGRAL or (for SPIBKG_INIT the HEAsoft) software documentation.

There are several complications regarding the spectral de-convolution of coded-aperture data. One already mentioned is the source confusion issue; there may be multiple sources in the FoV, which are lead to different degrees of shadowing on different detectors. Thus, a separate instrumental response must be applied to a spectral model for each possible source, for each detector. This is further compounded by the fact that INTEGRAL's typical mode of observation is "dithering." A single observation may consist of 10's of individual exposures at raster points separated by $\sim 2\,\mathrm{deg}$. This further enumerates the number of individual response matrices required for the analysis. If there are multiple sources in the FoV, then additional spectral models can be applied to an additional set of response matrices, enumerated as before over detector and dither pointing. This capability - to model more than one source at a time in a given χ^2 (or alternative) minimization procedure - did not exist in previous versions of XSPEC. For an observation with the INTEGRAL/SPI instrument, where the apparent detector efficiency is sensitive to the position of the source on the sky relative to the axis of the instrument, the statistic is:

$$\chi^2 = \sum_P \sum_d \sum_I \left(\frac{D_{d,p}(I) - \sum_j \sum_E R_{d,p;j} \times M_j(E;x_s) - B_{d,p}(I;x_b)}{\sigma_{d,p}(I)}\right)^2 \quad (2.10)$$

where p, d run over instrument pointings and detectors; I runs over individual detector channels; j enumerates the sources detected in the field at different position (θ, ϕ); E indexes the energies in the source model; x_s are the parameters of the source model, which is combined with the response; and x_b are parameters of the background model.

Examination of this equation reveals one more complication; the term B represents the background, which, unlike for chopping, scanning or imaging experiments, must be solved for simultaneously with the desired source content.

The proportion of background-to-source counts for a bright source such as the Crab is 1%. Furthermore, the background varies as a function of detector, and time (dither-points), making simple subtraction implausible. Thus, a model of the background is applied to a special response matrix, and included in the de-convolution algorithm.

2.7 References

- Arnaud, K.A., George, I.M., Tennant, A.F., 1992. Legacy, 2, 65.
- Avni, Y., 1976. ApJ, 210, 642.
- Bevington, P.R., 2002, 3rd Edition. Data Reduction and Error Analysis for the Physical Sciences, McGraw-Hill.
- Blissett, R.J., Cruise, A.M., 1979. MNRAS, 186, 45.
- George, I.M., Arnaud, K.A., Pence, W., Ruamsuwan, L., 1992. Legacy, 2, 51.
- Kahn, S.M., Blissett, R.J., 1980. ApJ, 238, 417.
- Loredo, T.J., Epstein, R.I., 1989. ApJ, 336, 896.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes (2nd edition), p687ff, CUP.
- Wheaton, W.A. et.al., 1995. ApJ, 438, 322.

Chapter 3

XSPEC Overview and Helpful Hints

3.1 Syntax

XSPEC is a command-driven, interactive program. You will see a prompt

```
XSPEC12>
```

whenever input is required. Command recall and inline editing are available using the arrow keys. XSPEC uses Tcl as its user interface, providing looping, conditionals, file I/O and so on. For further details of the Tcl syntax, consult the "Description of Syntax" section, appendix A, and links therein.

3.2 How to return to the XSPEC12>prompt

The string /* acts as an emergency escape back to the XSPEC prompt. This string in answer to any question should bounce XSPEC out of whatever it is doing and back to the command prompt.

3.3 Getting Help

Quick help: If you are uncertain about command syntax, typing a command followed by a "?" will print a one-line summary. The **help** command:

```
XSPEC12> help
```

without arguments will bring up the full XSPEC manual in a PDF document reader (e.g. Adobe™ Acrobat Reader), or will open a browser to the XSPEC manual home page either locally or on the HEASARC site. See "Customizing XSPEC" later in this chapter to see how to select between these options, and how to assign a PDF reader and web browser to XSPEC. Typing

```
XSPEC12> help <command>
```

will display the manual section corresponding to ¡command¿. Help for individual model components can be displayed by

```
XSPEC12> help model <modelName>
```

if all else fails you can e-mail your questions to the XSPEC team at xspec12@athena.gsfc.nasa.gov

3.4 Commands

XSPEC commands can be divided into 6 categories: Control, Data, Model, Fitting, Plotting and Setting, as follows:

- Control commands include items such as controlling logging, obtaining help, executing scripts, and other miscellaneous items to do with the program control rather than manipulating data or theoretical models.
- Data commands load spectral data and calibration data such as backgrounds and responses, and specify channel ranges to be fit.
- Model commands define and manipulate theoretical models and their parameters, and compute additional information such as fluxes or line identifications.

- Fit commands initiate the fitting routines, control the parameter set, perform statistical tests and compute confidence levels.
- Plot commands generate about 50 different kinds of 2-dimensional plots
- Setting commands change a variety of XSPEC internals which control details of models, statistics, and fitting methods.

These command types are summarized below. For full details see Chapter 5.

3.5 Issuing Commands

In an interactive session, the command interpreter accepts the shortest unambiguous abbreviation for any command. Since the interpreter is built on the Tcl language, some possible XSPEC command abbreviations might coincide with both XSPEC and Tcl commands. In this case, the interpreter will print the multiple possibilities and stop. Command abbreviations may be specified in a start-up script (`$HOME/.xspec/xspec.rc`), see Appendix A for details.

Inside a script, command abbreviations are not recognized. This behavior can be overridden but we do not recommended it: however, specific abbreviations can be defined in the custom startup script.

Operating-system commands can be given from within XSPEC simply by typing the command, as at the shell prompt: however, if wild cards are needed (e.g. ls *.pha) the operating system command must be preceded by **syscall**. Note that an XSPEC abbreviation which corresponds to a system command will run the latter.

3.6 Control Commands

Control commands include those for:

- controlling parallel operations: **parallel**
- writing program information: **log**, (save session to an ASCII file) **script** (record a set of commands), **save** (save commands needed to restore the current program state), **autosave** (automatically run the save command at specified intervals);

- controlling program output: **chatter** (control the amount of program output), **query** (give an automatic response to prompts), **tclout** and **tcloutr** (create Tcl variables for manipulation in scripts)
- displaying status information: **show**, **time**, and **version**
- ending the session: **exit** (or **quit**)
- displaying online help **help** and **?**. Help can be given either in summary, in specific manual pages, a manual section, or the entire manual.

3.7 Query, chatter and shutting XSPEC up (somewhat)

The **fit** command will run a certain number of iterations and then query the user whether he or she wants to continue. While useful under most circumstances, the constant questioning can be a pain if one leaves XSPEC running and expects to find it finished when one gets back, or if one habitually runs scripts. One way around this problem is to reset the number of iterations before the question is asked by giving a single argument. For example,

```
XSPEC12> fit 100
```

will run 100 iterations before asking a question.

A more drastic solution is to use the **query** command.

```
XSPEC12> query yes
```

This will suppress all questions and assume that their answer is yes, while

```
XSPEC12> query no
```

will suppress all questions but assume that their answer is no.

The amount of output that XSPEC writes is set by the **chatter** command, which takes two arguments applying to the terminal and to the log file.

3.8 Scripts and the Save command

XSPEC commands can be written into a file and then executed by entering

```
XSPEC12> @filename
```

Alternatively, from the shell prompt

```
% xspec - filename &
```

for batch execution (remember to end the script in file filename.xcm with exit if you want the program to terminate after completion!). Note that the default suffix for xspec scripts is .xcm

The **save** command writes the current XSPEC status to a command file, which later can be run to reset XSPEC to the same configuration. XSPEC has a mechanism to save the current status automatically. This is controlled through the **autosave** command.

This command is very useful when reading a large number of data sets and/or fitting complicated models. If autosaving is operating (the default) then the equivalent of

```
XSPEC12> save all xautosav.xcm
```

is run after each command, so if a disaster occurs it is possible to recover.

3.9 Miscellaneous

Information on the current XSPEC status can be printed out using the **show** command. The **time** command writes out system-timing information, and the version command writes out the version number and the build time and date. Finally, XSPEC can be terminated with the **exit** or **quit** commands.

3.10 Data Commands

XSPEC is designed to allow complicated, multi-instrument analysis, so most commands can take arguments specifying more than one data set. Arguments

in XSPEC are separated by either blanks or commas. A single argument can define a range. The ranges are delimited by a dash (-). A colon (:) is used to separate ranges (e.g., the phrase 1-2:11-24 refers to channels 11-24 in files 1 and 2).

Reading data and modifying calibration and auxiliary files

XSPEC reads in spectra from spectral files using the **data** command. Several datasets may be specified in one command. Several datasets may be stored in a single file and accessed separately. A particular dataset in use may be replaced by another or dropped entirely. The input data file contains pointers to background, redistribution and auxiliary response files, but these pointers may be overridden by the **backgrnd**, **response**, and **arf** commands. All these commands have the same syntax as data. An auxiliary background file, called the correction file (an absolute subtraction with zero variance), also can be included using the **corfile** command. Its use is described in the section on fitting. The current response can be replaced by a diagonal version using **diagrsp**. A dummy response for testing purposes can be defined using **dummyrsp**.

Controlling channels being fitted

PHA channels may be left out of fitting using the **ignore** command and included again using the **notice** command. These commands have a syntax allowing the same channels to be specified for more than one input file. The ignored and noticed ranges can be specified either as channels or as energies. If the command **setplot wave** has been entered, real ranges are interpreted as wavelengths.

Simulations

The **fakeit** command is used to generate simulated data. The current response matrix and model (a model must be defined prior to using the fakeit command) are used to create fake data. The user is prompted for various options. To make fake data when only a response matrix is available, give the command

```
XSPEC12> fakeit none.
```

XSPEC will prompt the user for the response and ancillary filenames from which to build the simulated data. It is important to note that a model must be defined prior to issuing this command.

Data groups

The most common use of XSPEC is to fit one or more data sets with responses to a particular model. However, it is often useful to be able to fit simultaneously several data sets with a model whose parameters can be different for each data

set. A simple example would be a number of data sets that we expect to have the same model spectrum shape but different normalizations. XSPEC caters to this need through the use of data groups. When files are read in they can be labeled as belonging to a particular data group. When a model is defined a set of model parameters is allocated for each data group. These parameters can all vary freely or they can be linked together across data groups as required.

To set up data groups, the **data** command should be given as in the following example :

```
XSPEC12> data 1:1 file1 1:2 file2 2:3 file3
```

which sets up two data groups. The first data group comprises data sets from file1 and file2, and the second data group takes the data set from file3. Now when a model is defined, XSPEC will give two sets of model parameters, one for the first datagroup and one for the second.

3.11 Model Commands

XSPEC allows users to fit data with models constructed from individual components. These components may be either additive, multiplicative, mixing, or convolution. Multiplicative components simply multiply the model by an energy-dependent factor. Convolutions apply a transformation to the model component they operated on whereby the output can be affected by a range of input energies, such as in smoothing operations. Mixing components are two dimensional and designed to transform fluxes between different spatial regions (such as in projection). Multiplicative, and convolution components can act on individual components, on groups of components, or on the entire model, whereas mixing transformations apply to the whole model.

The **model** command defines the model to be used and prompts for the starting values of its parameters. The user also can set the allowed ranges of the parameter. Parameters can be linked to an algebraic function of the other parameters, and unlinked using the **untie** command. The value of an individual parameter can be changed with the command **newpar** (and the current setting queried with **newpar 0**). Parameters can be fixed at their current value with the **freeze** command and allowed to vary freely with the **thaw** command. Individual components can be added or subtracted from the model using **addcomp**, **delcomp**, and **editmod**. The plasma emission and photoelectric absorption models require an assumption about relative elemental abundances.

The **flux** command calculates the flux from the current model in the given

energy range. This energy range must be within that defined by the current response matrix. If a larger energy range is required, then the **energies** command can be given to compute the model over the desired range. The **lumin** command calculates the luminosity for the source redshift given. The **eqwidth** command determines the equivalent width of a model component, usually a line. The user of either of these last two commands should read the help descriptions carefully. The Tcl script **addline** can be used to automatically add lines to a model. These can be identified using **identify** and **modid**.

New model components which can be described by a simple algebraic formula can be set up using **mdefine** and used in the same way as the standard models except they will run slower being interpreted rather than compiled.

Models with multiple responses and background models

Multiple models and responses can be assigned to a single spectrum. This generalizes and replaces the "/b" technique of specifying background models in v11. In the FITS file format, a single response file can be associated with a spectrum either through a header keyword or a table column entry. XSPEC always assigns this response to a spectrum's source number 1. The **model** command by default also creates new models for source number 1. The **response** command in tandem with **model** can be used to create additional sources. For example to add a background model to loaded spectrum 1, first load a 2nd response:

```
XSPEC12> response 2:1 resp2.rsp
```

then define a background model to apply to source 2:

```
XSPEC12> model 2:my_background_model_name wa(po)
```

This model will now apply to spectrum 1 and any other spectrum that has a response loaded for source 2. To apply a different background model to spectrum 2, load a response for source 3 rather than 2:

```
XSPEC12> response 3:2 another_response.rsp
XSPEC12> model 3:another_background_model ga
```

An arf can also be assigned to a particular source number and spectrum:

```
XSPEC12> arf 2:1 arf_file.pha
```

Source numbers do not need to be entered in consecutive order for a given spectrum, and gaps in numbering are allowed. Please see the individual model

and response entries in the "XSPEC Commands" section for more information and examples.

3.12 Fitting Commands

The basic fit command is called **fit**. This command performs a minimization using the currently selected algorithm (default: Levenberg-Marquardt). **fit** takes arguments that are passed to the fitting method: by default, these are the number of iterations to execute before asking the user whether to continue, and the numerical convergence criterion.

A systematic model uncertainty can be included using the **systematic** command. The **error** or **uncertain** command calculates error bounds for one interesting parameter for the specified parameters and confidence levels. To produce multi-dimensional errors the **steppar** command is used to generate a fit-statistic grid. Two-dimensional grids may be expressed as contour plots (using **plot contour**). The model normalization can be set using the **renorm** command. The normalization of the correction file background can be set with **cornorm**. **ftest** and the Tcl script **simftest** can be used to calculate F-test probabilities.

Markov Chain Monte Carlo runs can be performed using the **chain** command. The proposal distribution covariance can be rescaled using **chain rescale** if the Metropolis-Hastings algorithm is selected. If an MCMC chain has been loaded then **error** uses this chain. The analog of **steppar** is the **margin** command with the results being plotted using **plot margin** or **plot integprob**.

What to do when you have Poisson data

The χ^2 statistic assumes that all the spectral channels are Gaussian distributed and that the estimate for the variance is uncorrelated with the observed counts. If the data are Poisson then these are bad assumptions especially if there are small numbers of counts in a channel. An alternative fit statistic, the C-statistic, should be used in this case. The C-statistic can also provide confidence intervals in exactly the same way as χ^2. To use, give the command

```
XSPEC12> statistic cstat
```

and then use the **fit** and **error** commands as usual. An alternative (and deprecated) approach is to continue using the χ^2 statistic but change the weighting to provide a better estimate of the variance in the small number limit. This can be done using the **weight gehrels** or **weight churazov** commands. The

latter is to be preferred.

The goodness-of-fit statistic can be set using the command statistic test. There are a number of options available. They can be interpreted using the **goodness** command, which utilizes Monte Carlo methods.

3.13 Binning and Grouping data

Often one does not want to use the full resolution of a spectrum, either because the channels over-sample the spectral resolution or because the S/N is low. XSPEC and the associated programs provide a number of ways of handling this. Firstly, the XSPEC command **setplot rebin** can be used to add channels together in the plot. It is important to realize that this effects only the plot and not the data being fitted.

Two FTOOLS are available to bin and group data for fitting purposes. RBNPHA bins up the data in a non-reversible manner and should only be used to ensure that the number of bins in the spectrum is the same as that in the response. GRPPHA is the tool of choice for grouping the data to get adequate S/N or number of counts in each channel. GRPPHA does not actually add together channels, but instead sets a flag which is read by XSPEC and causes XSPEC to sum the appropriate channels. If a data file is read with some grouping then XSPEC will apply the same operation to any background or response files used.

3.14 Plotting Commands

XSPEC plotting is currently performed using the PLT interface. There are two basic commands: **plot** and **iplot**. The **plot** command makes a plot and returns the user to the XSPEC prompt, while the **iplot** command leaves the user in the interactive plotting interface, thus allowing the user to edit the plot. A variety of different quantities may be plotted, including the data and the current model; the integrated counts; the fit residuals; the ratio of data to model; the contributions to the fit statistic; the theoretical model; the unfolded (incident) spectrum; the detector efficiency; the results of the goodness command; and the fit-statistic contours. All data plots can have an x-axis of channels, energy, or wavelength, which are specified with **setplot channel**, **setplot energy**, **setplot wave** respectively. A number of different units are available for energy or wavelength. The plotting device to be used is set using **setplot device** or **cpd**. Separate spectra may be added together and channels binned up (for plotting

purposes only) using **setplot group** (and ungrouped with **setplot ungroup**) and **setplot rebin**. There is an option to plot individual additive model components on data plots, this option is enabled by **setplot add** and disabled by **setplot noadd**. The effective area can be divided out of data plots using **setplot area** (which option can be turned off using **setplot noarea**). Line IDs can be plotted using **setplot id** and turned off by **setplot noid**. A stack of PLT commands can be created and manipulated with **setplot command**, **setplot delete**, and **setplot list**. This command stack then is applied to every plot.

PLT is built on top of the PGPLOT package, which comes with a standard set of device drivers. Any machine running X-windows should support /xs and /xw, while xterm windows should support /xt. PGPLOT supports monochrome and color postscript and both landscape and portrait orientation with the drivers /ps, /cps, /vps, and /vcps. The easiest way to make a hardcopy of an XSPEC plot is to use

```
XSPEC12> iplot
```

command and then at the PLT prompt to enter

```
PLT> hard /ps
```

This will make a file called `pgplot.ps` which can be printed. Alternatively, the sequence

```
XSPEC12> cpd  <filename>/ps
XSPEC12> ... plot commands ...
XSPEC12> cpd none
```

will place the plots in a PostScript file <`filename`>.

3.15 Setting Commands

The fit and goodness-of-fit test statistics are set using the **statistic** command. Other fit-minimization algorithms are available, and can be selected using the **method** command. The various fit methods require first and in some cases second derivatives of the statistic with respect to the parameters. By default XSPEC calculates these analytically, using an approximation for the second derivatives. This may be changed by setting the USE_NUMERICAL_DIFFERENTIATION

flag in the user's startup Xspec.init file. The weighting algorithm used to calculate χ^2 can be altered by the **weight** command.

Other setting commands modify:

- cosmological parameters used to calculate luminosity (cosmo)
- solar abundances for 18 elements (abund)
- photoionization cross-sections (xsect)

The **xset** command can be used as an interface for **abund**, **cosmo**, **method**, **statistic**, and **xsect**. Additionally, **xset** may set string expressions that are used by models, for example the path to, and version number of AtomDB atomic line calculations, or the coordinate system for surface brightness calculations used in the **xmmpsf** mixing model.

3.16 Breaking With Ctrl-C

Ctrl-C can be used to break out of the **data**, **chain**, **error**, **fit**, and **steppar** commands. If a Ctrl-C is entered elsewhere, it will have no effect.

When a break is entered during the fitting commands (**error**, **fit**, and **steppar**), the fit will proceed until the end of the current fit iteration (ie. current lambda value when using Levenberg-Marquardt) before breaking. This is to ensure the program remains in a stable well-defined state. Therefore on slower machines, a user may notice a slight delay before the program actually breaks. Ctrl-C breaking is currently only implemented for the Levenberg-Marquardt fitting method.

Breaking is implemented for the data command primarily for users who load a large number of Type-II spectra with one data command. So if you enter

```
XSPEC12> data my_data{1-1000}
```

and decide it is taking too long to load, you can break out at any time. However, if you do choose to break, all spectra loaded from that particular data set will be lost. For example, if the command below is entered and a Ctrl-C is sent while the spectra from my_data2 are loading, the 50 spectra from my_data1 will be retained while none will be from my_data2:

```
XSPEC12> data my_data1{1-50} mydata2{1-50}
```

3.17 Customizing XSPEC

The XSPEC environment can be customized using two separate files, both of which are searched for in the directory $HOME/.xspec

The first file, Xspec.init contains a number of settings that control items in XSPEC. The settings available are as follows.

USE_ONLINE_HELP	whether to use on-line or local help files
LOCAL_HELP_FORMAT	local help format (html or pdf)
PDF_COMMAND	command to read pdf
HTML_COMMAND	command to read html
GUI	whether to use GUI mode (false only)
DUMMY	argument for dummyrsp command
CHAT	terminal and log chatter levels
XSECT	photoelectric cross-section table
ABUND	solar abundance table
LEVEL	fitting method
STATISTIC	fit statistic
WEIGHT	weighting for S^2 statistic
USE_NUMERICAL_DIFFER	calculate parameter derivatives numerically
COSMO	cosmology parameters (H_0, q_0, λ_0)
GRAPH	graphics package (plt only)
PLOTDEVICE	plotting device
WAVE_PLOT_UNITS	units for setplot wave mode
USER_SCRIPT_DIRECTORY	directory for user scripts
FIT_DELTAS	choice for parameters' fit delta values
ATOMDB_VERSION	the AtomDB version number
PARALLEL	Parallel processing settings
CONTOUR_IMAGE	Put background image on contour plots

A copy of this file is placed in the $HOME/.xspec directory on XSPEC12's first start-up or when it is not found. After this users can modify settings such as default cosmology or the energy range for dummy response for their own requirements.

This is also the place where users can select if they want to view PDF help files from the XSPEC distribution or HTML either locally or from the HEASARC site. Setting USE_ONLINE_HELP to true uses the remote HTML files while false will use either PDF or HTML local files depending on the value of LOCAL_HELP_FORMAT.

The PDF_COMMAND and HTML_COMMAND entries determine which applications are run for the two viewing cases. The HTML_COMMAND value should typically just be the name of a web browser, or "open" for Mac OS X

users. The default settings for the PDF_COMMAND entry are what work best for launching Adobe Acrobat Reader 7.0.x on Linux/Unix systems. For those launching earlier versions, the "-openInNewWindow" flag should be replaced with "-useFrontEndProgram". For Mac users, again we recommend the entire entry simply be replaced with "open".

The second file that is searched for is the xspec.rc file. This contains users' own customizations, for example Tcl or XSPEC command abbreviations, packages to be loaded on startup, or Tcl scripts containing procedures that are to be executed as commands. Please consult Appendix A and references/links therein for details of Tcl commands and scripting.

3.18 Customizing system-wide

When an XSPEC build is intended for many users across a system, it is also possible for the installer (or whoever has write access to the distribution and installation areas) to globally customize XSPEC. This is done through the file global_customize.tcl, located in the Xspec/src/scripts directory. (This was done in the xspec.tcl file prior to v12.2.1) Any of the customizations mentioned above for the individual's own xspec.rc file can also be placed in the global_customize.tcl file. After making the additions, run "hmake install" out of the Xspec/src/scripts directory in order to copy the modified global_customize.tcl file to the installation area. This additional code will be executed for all users upon startup, BEFORE any of their own customizations in their xspec.rc files.

Chapter 4

Walks through XSPEC

4.1 Introduction

This chapter demonstrates the use of XSPEC. The brief discussion of data and response files is followed by fully worked examples using real data that include all the screen input and output with a variety of plots. The topics covered are as follows: defining models, fitting data, determining errors, fitting more than one set of data simultaneously, simulating data, and producing plots.

4.2 Brief Discussion of XSPEC Files

At least two files are necessary for use with XSPEC: a data file and a response file. In some cases, a file containing background may also be used, and, in rare cases, a correction file is needed to adjust the background during fitting. If the response is split between an rmf and an arf then an ancillary response file is also required. However, most of the time the user need only specify the data file, as the names and locations of the correct response and background files should be written in the header of the data file by whatever program created the files.

4.3 Fitting Models to Data: An Old Example from EXOSAT

Our first example uses very old data which is much simpler than more modern observations and so can be used to better illustrate the basics of XSPEC analysis. The 6s X-ray pulsar 1E1048.1-5937 was observed by EXOSAT in June 1985 for 20 ks. In this example, we'll conduct a general investigation of the spectrum from the Medium Energy (ME) instrument, i.e. follow the same sort of steps as the original investigators (Seward, Charles & Smale, 1986). The ME spectrum and corresponding response matrix were obtained from the HEASARC On-line service. Once installed, XSPEC is invoked by typing

```
%xspec
```

as in this example:

```
%xspec

                XSPEC version: 12.9.0
        Build Date/Time: Sun Jun 28 17:58:18 2015

XSPEC12>data s54405.pha

1 spectrum  in use

Spectral Data File: s54405.pha  Spectrum 1
Net count rate (cts/s) for Spectrum:1  3.783e+00 +/- 1.367e-01
 Assigned to Data Group 1 and Plot Group 1
  Noticed Channels:  1-125
  Telescope: EXOSAT Instrument: ME  Channel Type: PHA
  Exposure Time: 2.358e+04 sec
 Using fit statistic: chi
 Using test statistic: chi
 Using Response (RMF) File            s54405.rsp for Source 1
```

The **data** command tells the program to read the data as well as the response file that is named in the header of the data file. In general, XSPEC commands can be truncated, provided they remain unambiguous. Since the default extension of a data file is .pha, and the abbreviation of **data** to the first two letters is unambiguous, the above command can be abbreviated to da s54405, if desired. To obtain help on the data command, or on any other command, type **help command** followed by the name of the command.

One of the first things most users will want to do at this stage – even before fitting models – is to look at their data. The plotting device should be set

first, with the command **cpd** (change plotting device). Here, we use the pgplot X-Window server, /xs.

```
XSPEC12> cpd /xs
```

There are more than 50 different things that can be plotted, all related in some way to the data, the model, the fit and the instrument. To see them, type:

```
XSPEC12> plot ?
plot data/models/fits etc
    Syntax: plot commands:
        background     chain          chisq          contour        counts
        data           delchi         dem            emodel         eemodel
        efficiency     eufspec        eeufspec       foldmodel      goodness
        icounts        insensitivity  lcounts        ldata          margin
        model          ratio          residuals      sensitivity    sum
        ufspec
    Multi-panel plots are created by entering multiple commands
      e.g. "plot data chisq"
```

The most fundamental is the data plotted against instrument channel (**data**); next most fundamental, and more informative, is the data plotted against channel energy. To do this plot, use the XSPEC command **setplot energy**. Figure 4.1 shows the result of the commands:

```
XSPEC12> setplot energy
XSPEC12> plot data
```

Note the label on the y-axis. The word "normalized" indicates that this plot has been divided by the value of the EFFAREA keyword in the response file. Usually this is unity so can be ignored. The label also has no cm-2 so the plot is not corrected for the effective area of the detector.

We are now ready to fit the data with a model. Models in XSPEC are specified using the **model** command, followed by an algebraic expression of a combination of model components. There are two basic kinds of model components: *additive*, which represent X-Ray sources of different kinds. After being convolved with the instrument response, the components prescribe the number of counts per energy bin (e.g., a bremsstrahlung continuum); and *multiplicative* models components, which represent phenomena that modify the observed X-Radiation (e.g. reddening or an absorption edge). They apply an energy-dependent multiplicative factor to the source radiation before the result is convolved with the instrumental response.

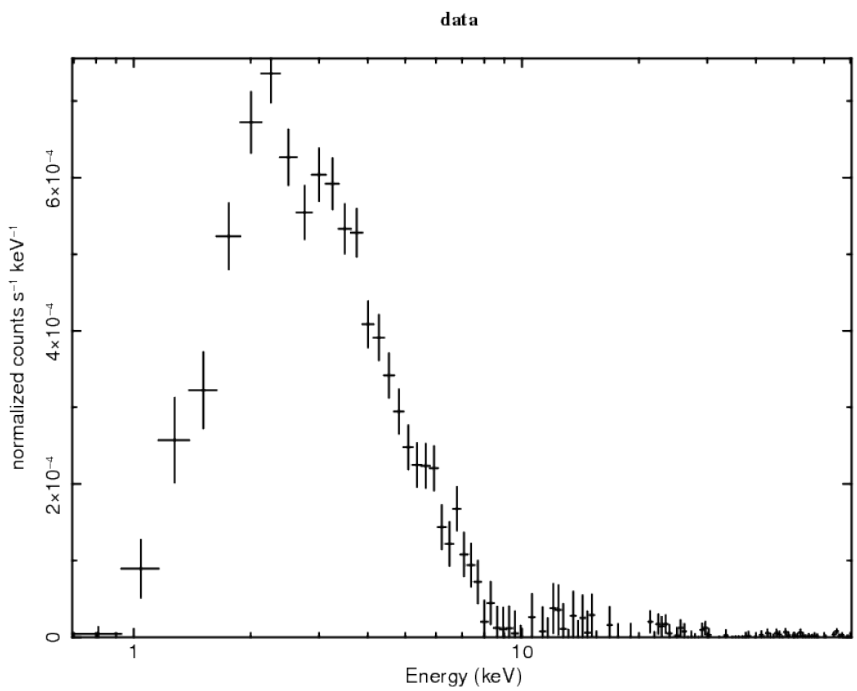

Figure 4.1: The result of the command plot data when the data file in question is the EXOSAT ME spectrum of the 6s X-ray pulsar 1E1048.1–5937 available from the HEASARC on-line service.

More generally, XSPEC allows three types of modifying components: convolutions and mixing models in addition to the multiplicative type. Since there must be a source, there must be least one additive component in a model, but there is no restriction on the number of modifying components. To see what components are available, just type **model** :

```
XSPEC12>model
 Additive Models:
agauss      apec        bapec       bbody       bbodyrad    bexrav
bexriv      bkn2pow     bknpower    bmc         bremss      bvapec
bvvapec     c6mekl      c6pmekl     c6pvmkl     c6vmekl     cemekl
cevmkl      cflow       compLS      compPS      compST      compTT
compbb      compmag     comptb      compth      cplinear    cutoffpl
disk        diskbb      diskir      diskline    diskm       disko
diskpbb     diskpn      eplogpar    eqpair      eqtherm     equil
expdec      ezdiskbb    gadem       gaussian    gnei        grad
grbm        kerrbb      kerrd       kerrdisk    laor        laor2
logpar      lorentz     meka        mekal       mkcflow     nei
nlapec      npshock     nsa         nsagrav     nsatmos     nsmax
nsmaxg      nsx         nteea       nthComp     optxagn     optxagnf
pegpwrlw    pexmon      pexrav      pexriv      plcabs      posm
powerlaw    pshock      raymond     redge       refsch      rnei
sedov       sirf        smaug       srcut       sresc       step
vapec       vbremss     vequil      vgadem      vgnei       vmcflow
vmeka       vmekal      vnei        vnpshock    vpshock     vraymond
vrnei       vsedov      vvapec      vvgnei      vvnei       vvnpshock
vvpshock    vvrnei      vvsedov     zagauss     zbbody      zbremss
zgauss      zpowerlw

 Multiplicative Models:
SSS_ice     TBabs       TBgrain     TBvarabs    absori      acisabs
cabs        constant    cyclabs     dust        edge        expabs
expfac      gabs        heilin      highecut    hrefl       lyman
notch       pcfabs      phabs       plabs       pwab        recorn
redden      smedge      spexpcut    spline      swind1      uvred
varabs      vphabs      wabs        wndabs      xion        zTBabs
zbabs       zdust       zedge       zhighect    zigm        zpcfabs
zphabs      zredden     zsmdust     zvarabs     zvfeabs     zvphabs
zwabs       zwndabs     zxipcf

 Convolution Models:
cflux       cpflux      gsmooth     ireflect    kdblur      kdblur2
kerrconv    lsmooth     partcov     rdblur      reflect     rgsxsrc
simpl       zashift     zmshift

 Mixing Models:
ascac       projct      suzpsf      xmmpsf

 Pile-up Models:
```

```
pileup

 Additional models are available at :
 legacy.gsfc.nasa.gov/docs/xanadu/xspec/newmodels.html
```

For information about a specific component, type **help model** followed by the name of the component):

```
XSPEC12>help model apec
```

Given the quality of our data, as shown by the plot, we'll choose an absorbed power law, specified as follows :

```
XSPEC12> model phabs(powerlaw)
```

Or, abbreviating unambiguously:

```
XSPEC12> mo pha(po)
```

The user is then prompted for the initial values of the parameters. Entering ¡return¿ or / in response to a prompt uses the default values. We could also have set all parameters to their default values by entering /* at the first prompt. As well as the parameter values themselves, users also may specify step sizes and ranges (value, delta, min, bot, top, and max values), but here, we'll enter the defaults:

```
XSPEC12>mo pha(po)

Input parameter value, delta, min, bot, top, and max values for ...
              1      0.001(      0.01)          0          0     100000   1E+06
1:phabs:nH>/*

=====================================================================
Model: phabs<1>*powerlaw<2> Source No.: 1  Active/On
Model  Model Component  Parameter  Unit    Value
 par  comp
  1     1    phabs      nH         10^22   1.00000       +/-  0.0
  2     2    powerlaw   PhoIndex           1.00000       +/-  0.0
  3     2    powerlaw   norm               1.00000       +/-  0.0
 ---------------------------------------------------------------------


Fit statistic : Chi-Squared =   4.864244e+08 using 125 PHA bins.
```

```
Test statistic : Chi-Squared =   4.864244e+08 using 125 PHA bins.
 Reduced chi-squared =   3.987085e+06 for    122 degrees of freedom
 Null hypothesis probability =   0.000000e+00
 Current data and model not fit yet.
```

The current statistic is χ^2 and is huge for the initial, default values – mostly because the power law normalization is two orders of magnitude too large. This is easily fixed using the **renorm** command.

```
XSPEC12> renorm
Fit statistic : Chi-Squared =         852.19 using 125 PHA bins.

Test statistic : Chi-Squared =         852.19 using 125 PHA bins.
 Reduced chi-squared =         6.9852 for    122 degrees of freedom
 Null hypothesis probability =  7.320765e-110
 Current data and model not fit yet.
```

We are not quite ready to fit the data (and obtain a better χ^2), because not all of the 125 PHA bins should be included in the fitting: some are below the lower discriminator of the instrument and therefore do not contain valid data; some have imperfect background subtraction at the margins of the pass band; and some may not contain enough counts for χ^2 to be strictly meaningful. To find out which channels to discard (ignore in XSPEC terminology), consult mission-specific documentation that will include information about discriminator settings, background subtraction problems and other issues. For the mature missions in the HEASARC archives, this information already has been encoded in the headers of the spectral files as a list of "bad" channels. Simply issue the command:

```
XSPEC12> ignore bad

ignore:    40 channels ignored from  source number 1
Fit statistic : Chi-Squared =         799.74 using 85 PHA bins.

Test statistic : Chi-Squared =         799.74 using 85 PHA bins.
 Reduced chi-squared =         9.7529 for     82 degrees of freedom
 Null hypothesis probability =  3.545709e-118
 Current data and model not fit yet.
XSPEC12> plot ldata chi
```

Giving two options for the plot command generates a plot with vertically stacked windows. Up to six options can be given to the plot command at a time. Forty channels were rejected because they were flagged as bad – but do we need to ignore any more? Figure 4.2 shows the result of plotting the data and the model

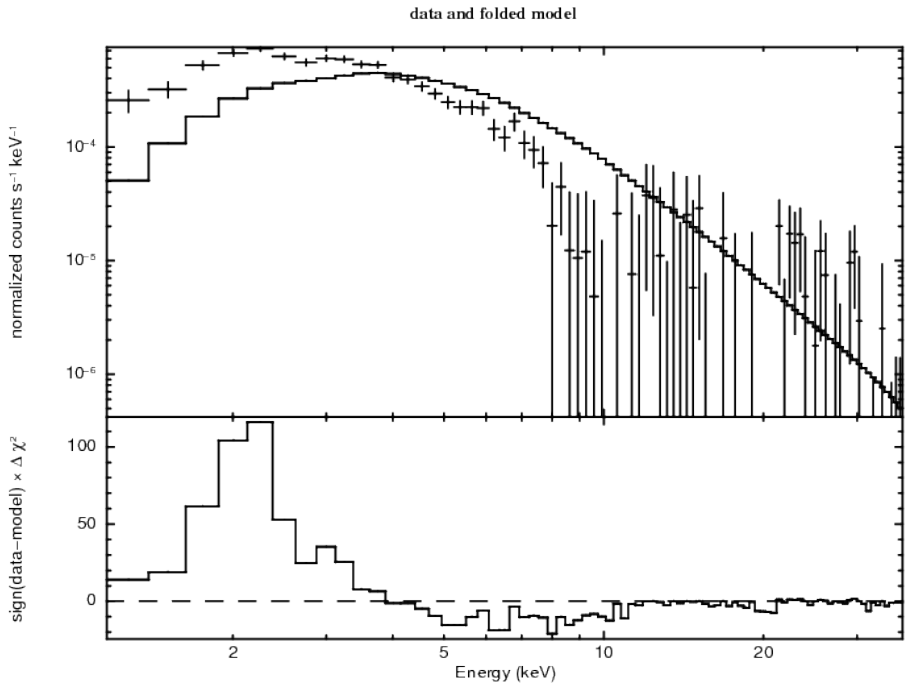

Figure 4.2: The result of the command plot ldata chi after the command ignore bad on the EXOSAT ME spectrum 1E1048.1-5937.

(in the upper window) and the contributions to χ^2 (in the lower window). We see that above about 15 keV the S/N becomes small. We also see, comparing Figure 4.2 with Figure 4.1, which bad channels were ignored. Although visual inspection is not the most rigorous method for deciding which channels to ignore (more on this subject later), it's good enough for now, and will at least prevent us from getting grossly misleading results from the fitting. To ignore energies above 15 keV:

```
XSPEC12> ignore 15.0-**
     78 channels (48-125) ignored in spectrum #     1

Fit statistic : Chi-Squared =         721.56 using 45 PHA bins.

Test statistic : Chi-Squared =         721.56 using 45 PHA bins.
 Reduced chi-squared =         17.180 for     42 degrees of freedom
 Null hypothesis probability =  1.253044e-124
 Current data and model not fit yet.
```

If the **ignore** command is handed a real number it assumes energy in keV while if it is handed an integer it will assume channel number. The "**" just means the highest energy. Starting a range with "**" means the lowest energy. The inverse of **ignore** is **notice**, which has the same syntax.

We are now ready to fit the data. Fitting is initiated by the command **fit**. As the fit proceeds, the screen displays the status of the fit for each iteration until either the fit converges to the minimum χ^2, or we are asked whether the fit is to go through another set of iterations to find the minimum. The default number of iterations before prompting is ten.

```
XSPEC12>fit
                                    Parameters
Chi-Squared  |beta|/N    Lvl          1:nH    2:PhoIndex        3:norm
451.814      150.854      -3     0.0961968       1.60719    0.00386478
413.575      63220.2      -3      0.264111       2.30580    0.00908043
53.9398      28104.3      -4      0.517263       2.14118     0.0121356
43.816       4617.17      -5      0.551755       2.23947     0.0130926
43.802       139.682      -6      0.538816       2.23680     0.0130394
43.802       0.58082      -7      0.537846       2.23646     0.0130320
========================================
 Variances and Principal Axes
                 1        2        3
 4.7889E-08| -0.0025  -0.0151   0.9999
 8.6827E-02| -0.9153  -0.4026  -0.0084
 2.2916E-03| -0.4027   0.9153   0.0128
----------------------------------------

====================================
```

```
  Covariance Matrix
        1           2           3
   7.312e-02   3.115e-02   6.565e-04
   3.115e-02   1.599e-02   3.208e-04
   6.565e-04   3.208e-04   6.562e-06
------------------------------------

========================================================================
Model phabs<1>*powerlaw<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    0.537846     +/-  0.270409
   2    2   powerlaw   PhoIndex            2.23646      +/-  0.126458
   3    2   powerlaw   norm                1.30320E-02  +/-  2.56170E-03
------------------------------------------------------------------------


Fit statistic : Chi-Squared =          43.80 using 45 PHA bins.

Test statistic : Chi-Squared =          43.80 using 45 PHA bins.
 Reduced chi-squared =          1.043 for     42 degrees of freedom
 Null hypothesis probability =   3.949507e-01
```

There is a fair amount of information here so we will unpack it a bit at a time. One line is written out after each fit iteration. The columns labeled Chi-Squared and Parameters are obvious. The other two provide additional information on fit convergence. At each step in the fit a numerical derivative of the statistic with respect to the parameters is calculated. We call the vector of these derivatives beta. At the best-fit the norm of beta should be zero so we write out —beta— divided by the number of parameters as a check. The actual default convergence criterion is when the fit statistic does not change significantly between iterations so it is possible for the fit to end while —beta— is still significantly different from zero. The —beta—/N column helps us spot this case. The Lvl column also indicates how the fit is converging and should generally decrease. Note that on the first iteration only the powerlaw norm is varied. While not necessary this simple model, for more complicated models only varying the norms on the first iteration helps the fit proper get started in a reasonable region of parameter space.

At the end of the fit XSPEC writes out the Variances and Principal Axes and Covariance Matrix sections. These are both based on the second derivatives of the statistic with respect to the parameters. Generally, the larger these second derivatives, the better determined the parameter (think of the case of a parabola in 1-D). The Covariance Matrix is the inverse of the matrix of second derivatives. The Variances and Principal Axes section is based on an eigenvector decomposition of the matrix of second derivatives and indicates which parameters are correlated. We can see in this case that the first eigenvector depends

almost entirely on the powerlaw norm while the other two are combinations of the nH and powerlaw PhoIndex. This tells us that the norm is independent but the other two parameters are correlated.

The next section shows the best-fit parameters and error estimates. The latter are just the square roots of the diagonal elements of the covariance matrix so implicitly assume that the parameter space is multidimensional Gaussian with all parameters independent. We already know in this case that the parameters are not independent so these error estimates should only be considered guidelines to help us determine the true errors later.

The final section shows the statistic values at the end of the fit. XSPEC defines a fit statistic, used to determine the best-fit parameters and errors, and test statistic, used to decide whether this model and parameters provide a good fit to the data. By default, both statistics are χ^2. When the test statistic is χ^2 we can also calculate the reduced χ^2 and the null hypothesis probability. This latter is the probability of getting a value of χ^2 as large or larger than observed if the model is correct. If this probability is small then the model is not a good fit. The null hypothesis probability can be calculated analytically for χ^2 but not for some other test statistics so XSPEC provides another way of determining the meaning of the statistic value. The **goodness** command performs simulations of the data based on the current model and parameters and compares the statistic values calculated with that for the real data. If the observed statistic is larger than the values for the simulated data this implies that the real data do not come from the model. To see how this works we will use the command for this case (where it is not necessary)

```
XSPEC12>goodness 1000
50.40% of realizations are < best fit statistic 43.80  (nosim)
XSPEC12>plot goodness
```

Approximately half of the simulations give a statistic value less than that observed, consistent with this being a good fit. Figure 4.3 shows a histogram of the χ^2 values from the simulations with the observed value shown by the vertical dotted line.

So the statistic implies the fit is good but it is still always a good idea to look at the data and residuals to check for any systematic differences that may not be caught by the test. To see the fit and the residuals, we produce figure 4.4 using the command

```
XSPEC12>plot data resid
```

Now that we think we have the correct model we need to determine how well the parameters are determined. The screen output at the end of the fit shows the

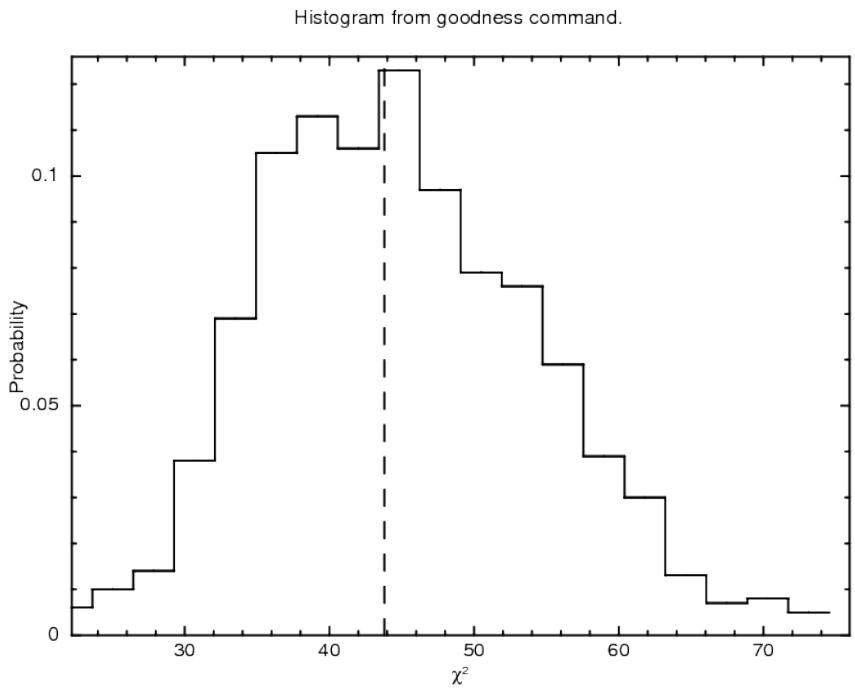

Figure 4.3: The result of the command plot goodness. The histogram shows the fraction of simulations with a given value of the statistic and the dotted line marks that for the observed data. There is no reason to reject the model.

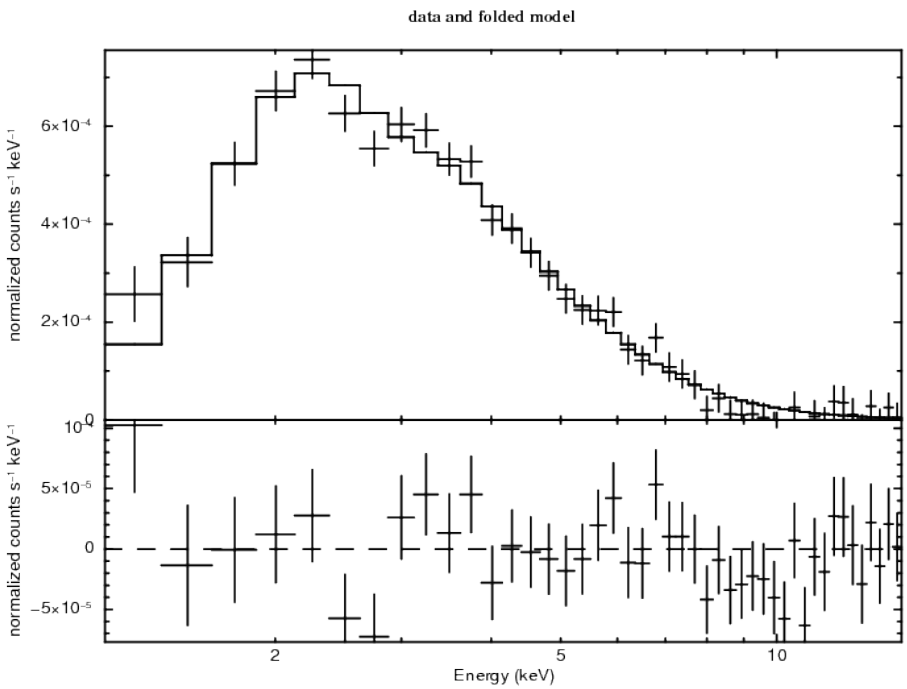

Figure 4.4: The result of the command plot data resid with: the ME data file from 1E1048.1-5937; "bad" and negative channels ignored; the best-fitting absorbed power-law model; the residuals of the fit.

best-fitting parameter values, as well as approximations to their errors. These errors should be regarded as indications of the uncertainties in the parameters and should not be quoted in publications. The true errors, i.e. the confidence ranges, are obtained using the **error** command. We want to run error on all three parameters which is an intrinsically parallel operation so we can use XSPEC's support for multiple cores and run the error estimations in parallel:

```
XSPEC12>parallel error 3
XSPEC12>error 1 2 3
 Parameter   Confidence Range (2.706)
     1     0.107599      1.00722    (-0.430231,0.469393)
     2      2.03775      2.44916    (-0.198718,0.212699)
     3   0.00954178    0.0181617    (-0.00349016,0.00512979)
```

Here, the numbers 1, 2, 3 refer to the parameter numbers in the Model par column of the output at the end of the fit. For the first parameter, the column of absorbing hydrogen atoms, the 90% confidence range is . This corresponds to an excursion in of 2.706. The reason these "better" errors are not given automatically as part of the **fit** output is that they entail further fitting. When the model is simple, this does not require much CPU, but for complicated models the extra time can be considerable. The error for each parameter is determined allowing the other two parameters to vary freely. If the parameters are uncorrelated this is all the information we need to know. However, we have an indication from the covariance matrix at the end of the fit that the column and photon index are correlated. To investigate this further we can use the command **steppar** to run a grid over these two parameters:

```
XSPEC12>steppar 1 0.0 1.5 25 2 1.5 3.0 25

     Chi-Squared    Delta                nH      PhoIndex
                 Chi-Squared              1             2

          162.65      118.84    0           0    0         1.5
          171.59      127.79    1        0.06    0         1.5
          180.87      137.06    2        0.12    0         1.5
          190.44      146.64    3        0.18    0         1.5
          200.29      156.49    4        0.24    0         1.5
          . . . . . . .
          316.02      272.22    4        0.24   25           3
          334.68      290.88    3        0.18   25           3
           354.2       310.4    2        0.12   25           3
          374.62      330.82    1        0.06   25           3
          395.94      352.14    0           0   25           3
```

and make the contour plot shown in figure 4.5 using:

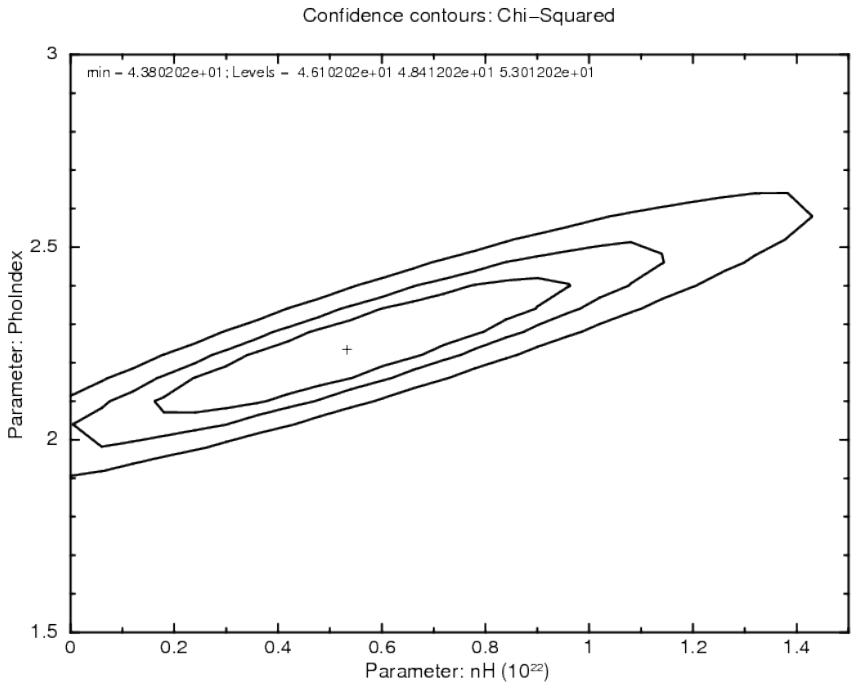

Figure 4.5: The result of the command plot contour. The contours shown are for one, two and three sigma. The cross marks the best-fit position.

```
XSPEC12>plot contour
```

What else can we do with the fit? One thing is to derive the flux of the model. The data by themselves only give the instrument-dependent count rate. The model, on the other hand, is an estimate of the true spectrum emitted. In XSPEC, the model is defined in physical units independent of the instrument. The command flux integrates the current model over the range specified by the user:

```
XSPEC12> flux 2 10
  Model Flux  0.003539 photons (2.2321e-11 ergs/cm^2/s) range (2.0000 - 10.000 keV)
```

Here we have chosen the standard X-ray range of 2–10 keV and find that the energy flux is 2.2×10^{-11} ergs/cm^2/s. Note that **flux** will integrate only within the energy range of the current response matrix. If the model flux outside this range is desired – in effect, an extrapolation beyond the data – then the command **energies** should be used. This command defines a set of energies on which the model will be calculated. The resulting model is then remapped onto the response energies for convolution with the response matrix. For example, if

we want to know the flux of our model in the ROSAT PSPC band of 0.2–2 keV, we enter:

```
XSPEC12>energies extend low 0.2 100

Models will use response energies extended to:
   Low:  0.2 in 100 log bins

Fit statistic : Chi-Squared =          43.80 using 45 PHA bins.

Test statistic : Chi-Squared =          43.80 using 45 PHA bins.
 Reduced chi-squared =          1.043 for     42 degrees of freedom
 Null hypothesis probability =   3.949507e-01
 Current data and model not fit yet.

XSPEC12>flux 0.2 2.
  Model Flux  0.0043484 photons (8.8419e-12 ergs/cm^2/s) range (0.20000 - 2.0000 keV)
```

The energy flux, at 8.8×10^{-12} ergs/cm^2/s is lower in this band but the photon flux is higher. The model energies can be reset to the response energies using **energies reset**. Calculating the flux is not usually enough, we want its uncertainty as well. The best way to do this is to use the cflux model. Suppose further that what we really want is the flux without the absorption then we include the new cflux model by:

```
XSPEC12>editmod pha*cflux(pow)

Input parameter value, delta, min, bot, top, and max values for ...
            0.5       -0.1(     0.005)          0          0      1e+06      1e+06
2:cflux:Emin>0.2
             10       -0.1(       0.1)          0          0      1e+06      1e+06
3:cflux:Emax>2.0
            -12       0.01(      0.12)       -100       -100        100        100
4:cflux:lg10Flux>-10.3

Fit statistic : Chi-Squared =          52.01 using 45 PHA bins.

Test statistic : Chi-Squared =          52.01 using 45 PHA bins.
 Reduced chi-squared =          1.268 for     41 degrees of freedom
 Null hypothesis probability =   1.163983e-01
 Current data and model not fit yet.

========================================================================
Model phabs<1>*cflux<2>*powerlaw<3> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    0.537843     +/-  0.270399
   2    2   cflux      Emin       keV      0.200000     frozen
```

```
   3    2   cflux      Emax       keV      2.00000      frozen
   4    2   cflux      lg10Flux   cgs      -10.3000     +/-  0.0
   5    3   powerlaw   PhoIndex            2.23646      +/-  0.126455
   6    3   powerlaw   norm                1.30320E-02  +/-  2.56146E-03

_______________________________________________________________________
```

The Emin and Emax parameters are set to the energy range over which we want the flux to be calculated. We also have to fix the norm of the powerlaw because the normalization of the model will now be determined by the lg10Flux parameter. This is done using the **freeze** command:

```
XSPEC12>freeze 6
```

We now run fit to get the best-fit value of lg10Flux as -10.2903 then:

```
XSPEC12>error 4
 Parameter   Confidence Range (2.706)
     4      -10.458     -10.0789    (-0.167672,0.211462)
```

for a 90% confidence range on the 0.2–2 keV unabsorbed flux of 3.49×10^{-11} – 8.33×10^{-11} ergs/cm^2/s.

The fit, as we've remarked, is good, and the parameters are constrained. But unless the purpose of our investigation is merely to measure a photon index, it's a good idea to check whether alternative models can fit the data just as well. We also should derive upper limits to components such as iron emission lines and additional continua, which, although not evident in the data nor required for a good fit, are nevertheless important to constrain. First, let's try an absorbed black body:

```
XSPEC12>mo pha(bb)

Input parameter value, delta, min, bot, top, and max values for ...
              1      0.001(      0.01)          0          0     100000      1e+06
1:phabs:nH>/*

=======================================================================
Model phabs<1>*bbody<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    1.00000      +/-  0.0
   2    2   bbody      kT         keV      3.00000      +/-  0.0
   3    2   bbody      norm                1.00000      +/-  0.0
_______________________________________________________________________
```

```
Fit statistic : Chi-Squared =   3.377094e+09 using 45 PHA bins.

Test statistic : Chi-Squared =   3.377094e+09 using 45 PHA bins.
 Reduced chi-squared =   8.040700e+07 for     42 degrees of freedom
 Null hypothesis probability =   0.000000e+00
 Current data and model not fit yet.
XSPEC12>fit
                                   Parameters
Chi-Squared  |beta|/N    Lvl          1:nH          2:kT        3:norm
1535.61      63.3168       0      0.334306       3.01647   0.000673086
1523.48      112166        0      0.157480       2.96616   0.000613284
1491.73      170831        0     0.0668724       2.87680   0.000570109
1444.74      204638        0     0.0228530       2.76753   0.000535209
1387.84      226856        0    0.00206016       2.64901   0.000504576
1325.39      243767        0   0.000851460       2.52617   0.000476469
1256.03      258162        0   0.000299777       2.40098   0.000449911
1180.44      271941        0   4.82004e-05       2.27529   0.000425076
1093.77      284819        0   1.99048e-05       2.14630   0.000402842
976.95       291176        0   9.75848e-06       1.99181   0.000380678
Number of trials exceeded: continue fitting? Y
 ...
 ...
 123.773      25.397       -8   1.87147e-08      0.890295   0.000278599
Number of trials exceeded: continue fitting?
***Warning: Zero alpha-matrix diagonal element for parameter 1
 Parameter 1 is pegged at 1.87147e-08 due to zero or negative pivot element, likely
 caused by the fit being insensitive to the parameter.
123.773      1.92501      -3   1.87147e-08      0.890205   0.000278596
==============================
 Variances and Principal Axes
                 2        3
 2.8677E-04| -1.0000  -0.0000
 2.2370E-11|  0.0000  -1.0000
------------------------------

========================
  Covariance Matrix
        1           2
   2.867e-04   9.315e-09
   9.315e-09   2.267e-11
------------------------

=========================================================================
Model phabs<1>*bbody<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    2.24621E-09  +/-  -1.00000
   2    2   bbody      kT         keV      0.890178     +/-  1.69312E-02
```

```
   3    2   bbody      norm                2.78595E-04  +/-  4.76156E-06
_______________________________________________________________________


Fit statistic : Chi-Squared =         123.77 using 45 PHA bins.

Test statistic : Chi-Squared =         123.77 using 45 PHA bins.
 Reduced chi-squared =         2.9470 for     42 degrees of freedom
 Null hypothesis probability =   5.417154e-10
```

Note that after each set of 10 iterations you are asked whether you want to continue. Replying *no* at these prompts is a good idea if the fit is not converging quickly. Conversely, to avoid having to keep answering the question, i.e., to increase the number of iterations before the prompting question appears, begin the fit with, say **fit 100**. This command will put the fit through 100 iterations before pausing. To automatically answer yes to all such questions use the command **query yes**.

Note that the fit has written out a warning about the first parameter and its estimated error is written as -1. This indicates that the fit is unable to constrain the parameter and it should be considered indeterminate. This usually indicates that the model is not appropriate. One thing to check in this case is that the model component has any contribution within the energy range being calculated. Plotting the data and residuals again we obtain Figure 4.6.

The black body fit is obviously not a good one. Not only is χ^2 large, but the best-fitting N_H is indeterminate. Inspection of the residuals confirms this: the pronounced wave-like shape is indicative of a bad choice of overall continuum.

Let's try thermal bremsstrahlung next:

```
XSPEC12>mo pha(br)
Input parameter value, delta, min, bot, top, and max values for ...
              1      0.001(      0.01)          0          0     100000      1e+06
1:phabs:nH>/*

=======================================================================
Model phabs<1>*bremss<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    1.00000      +/-  0.0
   2    2   bremss     kT         keV      7.00000      +/-  0.0
   3    2   bremss     norm                1.00000      +/-  0.0
_______________________________________________________________________


Fit statistic : Chi-Squared =   4.534834e+07 using 45 PHA bins.
```

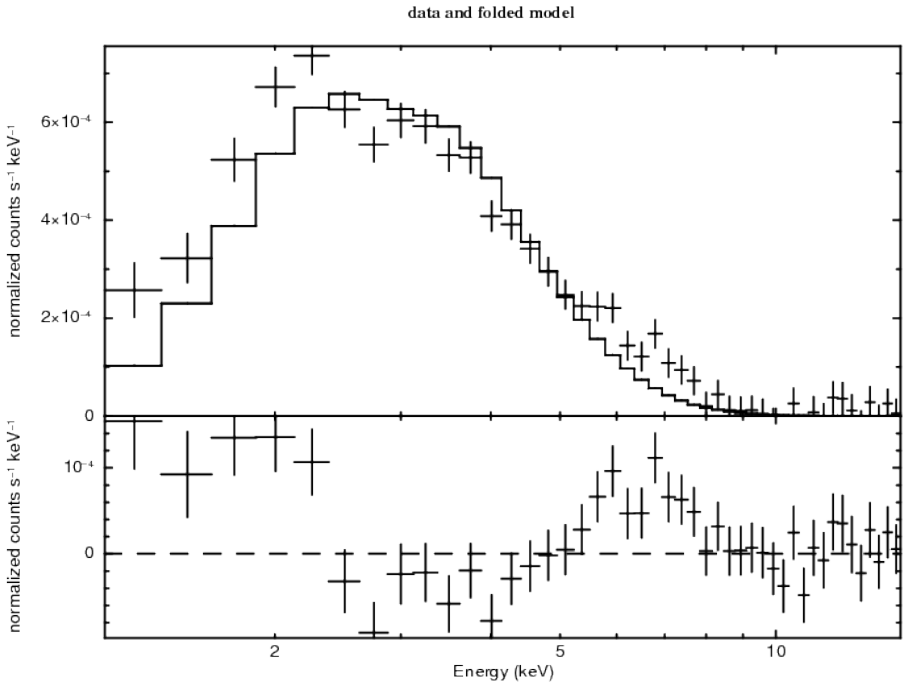

Figure 4.6: As for Figure 4.4, but the model is the best-fitting absorbed black body. Note the wave-like shape of the residuals which indicates how poor the fit is, i.e. that the continuum is obviously not a black body.

```
Test statistic : Chi-Squared =   4.534834e+07 using 45 PHA bins.
 Reduced chi-squared =   1.079722e+06 for     42 degrees of freedom
 Null hypothesis probability =   0.000000e+00
 Current data and model not fit yet.
XSPEC12>fit
                                   Parameters
Chi-Squared  |beta|/N    Lvl          1:nH          2:kT        3:norm
105.28       24.2507      -3      0.273734       6.18714    0.00724161
46.8022      16593.4      -4     0.0371200       5.59937    0.00785588
...
...
40.0373      270.662       0   7.62629e-05       5.28989    0.00830799
========================================
 Variances and Principal Axes
                 1        2        3
 1.9514E-08| -0.0016   0.0007   1.0000
 1.1574E-02|  0.9736   0.2281   0.0014
 5.3111E-01|  0.2281  -0.9736   0.0011
----------------------------------------

====================================
  Covariance Matrix
        1           2           3
   3.862e-02  -1.148e-01   1.431e-04
  -1.148e-01   5.015e-01  -5.379e-04
   1.431e-04  -5.379e-04   6.290e-07
------------------------------------

=========================================================================
Model phabs<1>*bremss<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    7.62629E-05  +/-  0.196509
   2    2   bremss     kT         keV      5.28989      +/-  0.708184
   3    2   bremss     norm                8.30799E-03  +/-  7.93082E-04
-------------------------------------------------------------------------


Fit statistic : Chi-Squared =          40.04 using 45 PHA bins.

Test statistic : Chi-Squared =          40.04 using 45 PHA bins.
 Reduced chi-squared =         0.9533 for     42 degrees of freedom
 Null hypothesis probability =   5.574343e-01
```

Bremsstrahlung is a better fit than the black body – and is as good as the power law – although it shares the low N_H. With two good fits, the power law and the bremsstrahlung, it's time to scrutinize their parameters in more detail.

First, we reset our fit to the powerlaw (output omitted):

```
XSPEC12>mo pha(po)
```

From the EXOSAT database on HEASARC, we know that the target in question, 1E1048.1–5937, has a Galactic latitude of , i.e., almost on the plane of the Galaxy. In fact, the database also provides the value of the Galactic N_H based on 21-cm radio observations. At 4×10^{22} cm^{-2}, it is higher than the 90 percent-confidence upper limit from the power-law fit. Perhaps, then, the power-law fit is not so good after all. What we can do is fix (**freeze** in XSPEC terminology) the value of N_H at the Galactic value and refit the power law. Although we won't get a good fit, the shape of the residuals might give us a clue to what is missing. To freeze a parameter in XSPEC, use the command **freeze** followed by the parameter number, like this:

```
XSPEC12> freeze 1
```

The inverse of **freeze** is **thaw**:

```
XSPEC12> thaw 1
```

Alternatively, parameters can be frozen using the **newpar** command, which allows all the quantities associated with a parameter to be changed. We can flip between frozen and thawed states by entering 0 after the new parameter value. In our case, we want N_H frozen at 4×10^{22} cm^{-2}, so we go back to the power law best fit and do the following :

```
XSPEC12>newpar 1
Current value, delta, min, bot, top, and max values
       0.537843      0.001(0.00537843)          0          0     100000      1e+06
1:phabs[1]:nH:1>4,0

Fit statistic : Chi-Squared =         823.34 using 45 PHA bins.

Test statistic : Chi-Squared =         823.34 using 45 PHA bins.
 Reduced chi-squared =         19.148 for     43 degrees of freedom
 Null hypothesis probability =  6.152922e-145
 Current data and model not fit yet.
```

The same result can be obtained by putting everything onto the command line, i.e., **newpar 1 4, 0**, or by issuing the two commands, **newpar 1 4** followed by **freeze 1**. Now, if we fit and plot again, we get the following model (Fig. 4.7).

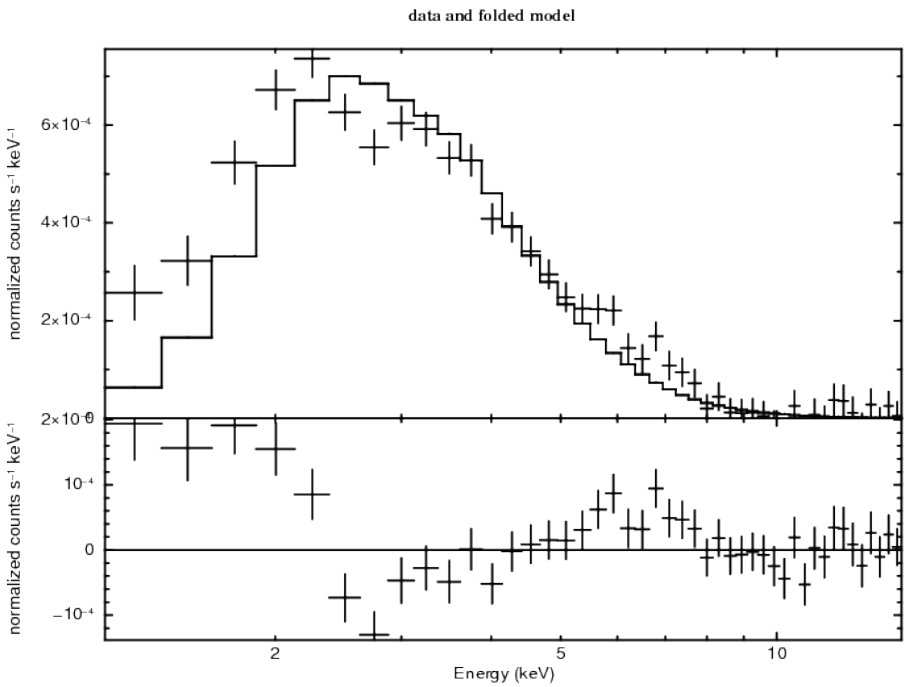

Figure 4.7: As for Figure 4.4 & 4.6, but the model is the best-fitting power law with the absorption fixed at the Galactic value. Under the assumptions that the absorption really is the same as the 21-cm value and that the continuum really is a power law, this plot provides some indication of what other components might be added to the model to improve the fit.

```
XSPEC12>fit
...
 ========================================================================
Model phabs<1>*powerlaw<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    4.00000      frozen
   2    2   powerlaw   PhoIndex            3.59784      +/-  6.76670E-02
   3    2   powerlaw   norm                0.116579     +/-  9.43208E-03
________________________________________________________________________


Fit statistic : Chi-Squared =         136.04 using 45 PHA bins.
```

The fit is not good. In Figure 4.7 we can see why: there appears to be a surplus of softer photons, perhaps indicating a second continuum component. To investigate this possibility we can add a component to our model. The **editmod** command lets us do this without having to respecify the model from scratch. Here, we'll add a black body component.

```
XSPEC12>editmod pha(po+bb)

Input parameter value, delta, min, bot, top, and max values for ...
              3       0.01(      0.03)     0.0001       0.01        100        200
4:bbody:kT>2,0
              1       0.01(      0.01)          0          0      1e+24      1e+24
5:bbody:norm>1e-5

Fit statistic : Chi-Squared =         132.76 using 45 PHA bins.

Test statistic : Chi-Squared =         132.76 using 45 PHA bins.
 Reduced chi-squared =         3.1610 for     42 degrees of freedom
 Null hypothesis probability =   2.387580e-11
 Current data and model not fit yet.

========================================================================
Model phabs<1>(powerlaw<2> + bbody<3>) Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    4.00000      frozen
   2    2   powerlaw   PhoIndex            3.59784      +/-  6.76670E-02
   3    2   powerlaw   norm                0.116579     +/-  9.43208E-03
   4    3   bbody      kT         keV      2.00000      frozen
   5    3   bbody      norm                1.00000E-05  +/-  0.0

________________________________________________________________________
```

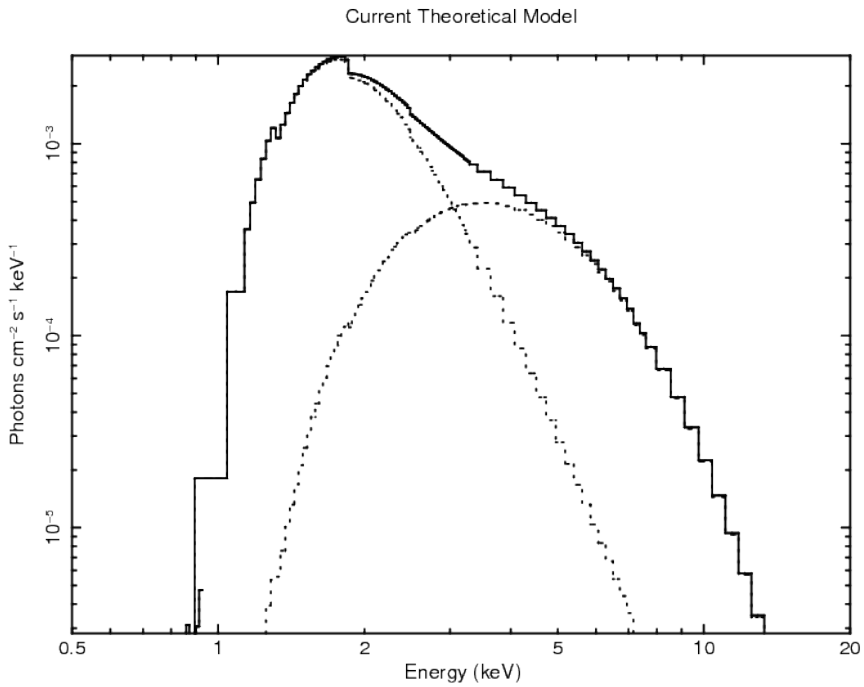

Figure 4.8: The result of the command plot model in the case of the ME data file from 1E1048.1-5937. Here, the model is the best-fitting combination of power law, black body and fixed Galactic absorption. The three lines show the two continuum components (absorbed to the same degree) and their sum.

Notice that in specifying the initial values of the black body, we have frozen kT at 2 keV (the canonical temperature for nuclear burning on the surface of a neutron star in a low-mass X-ray binary) and started the normalization small. Without these measures, the fit might "lose its way". Now, if we fit, we get (not showing all the iterations this time):

```
========================================================================
Model phabs<1>(powerlaw<2> + bbody<3>) Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    4.00000      frozen
   2    2   powerlaw   PhoIndex            4.89633      +/-  0.158893
   3    2   powerlaw   norm                0.365391     +/-  5.26695E-02
   4    3   bbody      kT         keV      2.00000      frozen
   5    3   bbody      norm                2.29745E-04  +/-  2.03915E-05
------------------------------------------------------------------------


Fit statistic : Chi-Squared =          69.53 using 45 PHA bins.
```

The fit is better than the one with just a power law and the fixed Galactic column, but it is still not good. Thawing the black body temperature and fitting does of course improve the fit but the powerl law index becomes even steeper. Looking at this odd model with the command

```
XSPEC12> plot model
```

We see, in Figure 4.8, that the black body and the power law have changed places, in that the power law provides the soft photons required by the high absorption, while the black body provides the harder photons. We could continue to search for a plausible, well-fitting model, but the data, with their limited signal-to-noise and energy resolution, probably don't warrant it (the original investigators published only the power law fit).

There is, however, one final, useful thing to do with the data: derive an upper limit to the presence of a fluorescent iron emission line. First we delete the black body component using **delcomp** then thaw N_H and refit to recover our original best fit. Now, we add a gaussian emission line of fixed energy and width then fit to get:

```
========================================================================
Model phabs<1>(powerlaw<2> + gaussian<3>) Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    0.753994     +/-  0.320347
```

```
   2    2   powerlaw   PhoIndex            2.38165      +/-  0.166974
   3    2   powerlaw   norm                1.59131E-02  +/-  3.94933E-03
   4    3   gaussian   LineE      keV      6.40000      frozen
   5    3   gaussian   Sigma      keV      0.100000     frozen
   6    3   gaussian   norm                7.47374E-05  +/-  4.74253E-05
_______________________________________________________________________
```

The energy and width have to be frozen because, in the absence of an obvious line in the data, the fit would be completely unable to converge on meaningful values. Besides, our aim is to see how bright a line at 6.4 keV can be and still not ruin the fit. To do this, we fit first and then use the **error** command to derive the maximum allowable iron line normalization. We then set the normalization at this maximum value with **newpar** and, finally, derive the equivalent width using the **eqwidth** command. That is:

```
XSPEC12>err 6
 Parameter   Confidence Range (2.706)
***Warning: Parameter pegged at hard limit: 0
     6            0  0.000151164    (-7.476e-05,7.64036e-05)
XSPEC12>new 6 0.000151164

Fit statistic : Chi-Squared =          46.03 using 45 PHA bins.

Test statistic : Chi-Squared =          46.03 using 45 PHA bins.
 Reduced chi-squared =          1.123 for     41 degrees of freedom
 Null hypothesis probability =   2.717072e-01
 Current data and model not fit yet.
XSPEC12>eqwidth 3

Data group number: 1
Additive group equiv width for Component 3:  0.784169 keV
```

Things to note:

- The true minimum value of the gaussian normalization is less than zero, but the **error** command stopped searching for a $\Delta\chi^2$ of 2.706 when the minimum value hit zero, the "hard" lower limit of the parameter. Hard limits can be adjusted with the **newpar** command, and they correspond to the quantities min and max associated with the parameter values.

- The command **eqwidth** takes the component number as its argument.

- The upper limit on the equivalent width of a 6.4 keV emission line is high (784 eV)!

4.4 Simultaneous Fitting

XSPEC has the very useful facility of allowing models to be fitted simultaneously to more than one data file. It is even possible to group files together and to fit different models simultaneously. Reasons for fitting in this manner include:

- *The same target is observed at several epochs but, although the source stays constant, the response matrix has changed.* When this happens, the data files cannot be added together; they have to be fitted separately. Fitting the data files simultaneously yields tighter constraints.
- *The same target is observed with different instruments.* All the instruments on Suzaku, for example, observe in the same direction simultaneously. As far as XSPEC is concerned, this is just like the previous case: two data files with two responses fitted simultaneously with the same model.
- *Different targets are observed, but the user wants to fit the same model to each data file with some parameters shared and some allowed to vary separately.* For example, if we have a series of spectra from a variable AGN, we might want to fit them simultaneously with a model that has the same, common photon index but separately vary the normalization and absorption.

Other scenarios are possible—the important thing is to recognize the flexibility of XSPEC in this regard.

As an example we will look at a case of fitting the same model to two different data files but where not all the parameters are identical. Again, this is an older dataset that provides a simpler illustration than more modern data. The massive X-ray binary Centaurus X-3 was observed with the LAC on Ginga in 1989. Its flux level before eclipse was much lower than the level after eclipse. Here, we'll use XSPEC to see whether spectra from these two phases can be fitted with the same model, which differs only in the amount of absorption. This kind of fitting relies on introducing an extra dimension, the *group*, to the indexing of the data files. The files in each group share the same model but not necessarily the same parameter values, which may be shared as common to all the groups or varied separately from group to group. Although each group may contain more than one file, there is only one file in each of the two groups in this example. Groups are specified with the data command, with the group number preceding the file number, like this:

```
XSPEC12>data 1:1 losum 2:2 hisum

2 spectra  in use
```

```
Spectral Data File: losum.pha  Spectrum 1
Net count rate (cts/s) for Spectrum:1  1.401e+02 +/- 3.549e-01
 Assigned to Data Group 1 and Plot Group 1
  Noticed Channels:  1-48
  Telescope: GINGA Instrument: LAC  Channel Type: PHA
  Exposure Time: 1 sec
 Using fit statistic: chi
 Using test statistic: chi
 Using Response (RMF) File            ginga_lac.rsp for Source 1

Spectral Data File: hisum.pha  Spectrum 2
Net count rate (cts/s) for Spectrum:2  1.371e+03 +/- 3.123e+00
 Assigned to Data Group 2 and Plot Group 2
  Noticed Channels:  1-48
  Telescope: GINGA Instrument: LAC  Channel Type: PHA
  Exposure Time: 1 sec
 Using fit statistic: chi
 Using test statistic: chi
 Using Response (RMF) File            ginga_lac.rsp for Source 1
```

Here, the first group makes up the file losum.pha, which contains the spectrum of all the low, pre-eclipse emission. The second group makes up the second file, hisum.pha, which contains all the high, post-eclipse emission. Note that file number is "absolute" in the sense that it is independent of group number. Thus, if there were three files in each of the two groups (lo1.pha, lo2.pha, lo3.pha, hi1.pha, hi2.pha, and hi3.pha, say), rather than one, the six files would be specified as **da 1:1 lo1 1:2 lo2 1:3 lo3 2:4 hi1 2:5 hi2 2:6 hi3**. The **ignore** command works on file number, and does not take group number into account. So, to ignore channels 1–3 and 37–48 of both files:

```
XSPEC12> ignore 1-2:1-3 37-48
```

The model we'll use at first to fit the two files is an absorbed power law with a high-energy cut-off:

```
XSPEC12> mo phabs * highecut (po)
```

After defining the model, we will be prompted for two sets of parameter values, one for the first group of data files (losum.pha), the other for the second group (hisum.pha). Here, we'll enter the absorption column of the first group as 10^{24} cm^{-2} and enter the default values for all the other parameters in the first group. Now, when it comes to the second group of parameters, we enter a column of 10^{22} cm^{-}2 and then enter defaults for the other parameters. The rule being applied here is as follows: to tie parameters in the second group to their equivalents in

the first group, take the default when entering the second-group parameters; to allow parameters in the second group to vary independently of their equivalents in the first group, enter different values explicitly:

```
XSPEC12>mo phabs*highecut(po)

Input parameter value, delta, min, bot, top, and max values for ...
Current:     1   0.001     0     0   1E+05   1E+06
DataGroup 1:phabs:nH>100
Current:     10   0.01  0.0001   0.01   1E+06   1E+06
DataGroup 1:highecut:cutoffE>
Current:     15   0.01  0.0001   0.01   1E+06   1E+06
DataGroup 1:highecut:foldE>
Current:     1   0.01    -3    -2     9    10
DataGroup 1:powerlaw:PhoIndex>
Current:     1   0.01     0     0   1E+24   1E+24
DataGroup 1:powerlaw:norm>
Current:    100   0.001     0     0   1E+05   1E+06
DataGroup 2:phabs:nH>1
Current:     10   0.01  0.0001   0.01   1E+06   1E+06
DataGroup 2:highecut:cutoffE>/*
========================================================================
Model phabs<1>*highecut<2>*powerlaw<3> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
                            Data group: 1
   1    1   phabs      nH         10^22    100.000      +/-  0.0
   2    2   highecut   cutoffE    keV      10.0000      +/-  0.0
   3    2   highecut   foldE      keV      15.0000      +/-  0.0
   4    3   powerlaw   PhoIndex            1.00000      +/-  0.0
   5    3   powerlaw   norm                1.00000      +/-  0.0
                            Data group: 2
   6    1   phabs      nH         10^22    1.00000      +/-  0.0
   7    2   highecut   cutoffE    keV      10.0000      = 2
   8    2   highecut   foldE      keV      15.0000      = 3
   9    3   powerlaw   PhoIndex            1.00000      = 4
  10    3   powerlaw   norm                1.00000      = 5
________________________________________________________________________
```

Notice how the summary of the model, displayed immediately above, is different now that we have two groups, as opposed to one (as in all the previous examples). We can see that of the 10 model parameters, 6 are free (i.e., 4 of the second group parameters are tied to their equivalents in the first group). Fitting this model results in a huge χ^2 (not shown here), because our assumption that only a change in absorption can account for the spectral variation before and after eclipse is clearly wrong. Perhaps scattering also plays a role in reducing the flux before eclipse. This could be modeled (simply at first) by allowing the normalization of the power law to be smaller before eclipse than after eclipse.

To decouple tied parameters, we change the parameter value in the second group to a value – any value – different from that in the first group (changing the value in the first group has the effect of changing both without decoupling). As usual, the **newpar** command is used:

```
XSPEC12>newpar 10 1

Fit statistic : Chi-Squared =   2.025975e+07 using 66 PHA bins.

Test statistic : Chi-Squared =   2.025975e+07 using 66 PHA bins.
 Reduced chi-squared =       343385.7 for     59 degrees of freedom
 Null hypothesis probability =   0.000000e+00
 Current data and model not fit yet.
XSPEC12>fit
...
 ========================================================================
Model phabs<1>*highecut<2>*powerlaw<3> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
                            Data group: 1
   1    1   phabs      nH         10^22    20.1548      +/-  0.181919
   2    2   highecut   cutoffE    keV      14.6847      +/-  5.59260E-02
   3    2   highecut   foldE      keV      7.41661      +/-  8.99590E-02
   4    3   powerlaw   PhoIndex            1.18693      +/-  6.33041E-03
   5    3   powerlaw   norm                5.88350E-02  +/-  9.30517E-04
                            Data group: 2
   6    1   phabs      nH         10^22    1.27017      +/-  3.77710E-02
   7    2   highecut   cutoffE    keV      14.6847      = p2
   8    2   highecut   foldE      keV      7.41661      = p3
   9    3   powerlaw   PhoIndex            1.18693      = p4
  10    3   powerlaw   norm                0.312138     +/-  4.49104E-03
________________________________________________________________________


Fit statistic : Chi-Squared =       15423.79 using 66 PHA bins.
```

After fitting, this decoupling reduces χ^2 by a factor of six to 15,478, but this is still too high. Indeed, this simple attempt to account for the spectral variability in terms of "blanket" cold absorption and scattering does not work. More sophisticated models, involving additional components and partial absorption, should be tried.

4.5 Multiple Models: a Background Modeling Example

In the previous section we showed how to fit the same model to multiple datasets. We now demonstrate how to fit multiple models, each with their own response, to the same dataset. There are several reasons why this may be useful, for instance:

- *We are using data from a coded aperture mask.* If there are multiple sources in the field they will all contribute to the spectrum from each detector. However, each source may have a different response due to its position.
- it We are observing an extended source using a telescope whose PSF is large enough that the signal from different regions are mixed together. In this case we will want to analyze spectra from all regions of the source simultaneously with each spectrum having a contribution from the model in other regions.
- *We wish to model the background spectrum that includes a particle component.* The particle background will have a different response from the X-ray background because the particles come from all directions, not just down the telescope.

We will demonstrate the third example here. Suppose we have a model for the background spectrum that requires a different response to that for the source spectrum. Read in the source and background spectra as separate files:

```
XSPEC12>data 1:1 source.pha 2:2 back.pha
```

The source and background files have their own response matrices:

```
XSPEC12>response 1 source.rsp 2 back.rsp
```

Set up the model for the source. Here we will take the simple case of an absorbed power-law:

```
XSPEC12>model phabs(pow)

Input parameter value, delta, min, bot, top, and max values for ...
1      0.001(      0.01)          0          0     100000      1e+06
1:data group 1::phabs:nH>
```

```
              1       0.01(      0.01)         -3         -2          9         10
2:data group 1::powerlaw:PhoIndex>
              1       0.01(      0.01)          0          0      1e+24      1e+24
3:data group 1::powerlaw:norm>

Input parameter value, delta, min, bot, top, and max values for ...
              1      0.001(      0.01)          0          0     100000      1e+06
4:data group 2::phabs:nH>
              1       0.01(      0.01)         -3         -2          9         10
5:data group 2::powerlaw:PhoIndex>
              1       0.01(      0.01)          0          0      1e+24      1e+24
6:data group 2::powerlaw:norm>0 0
```

Note that we have fixed the normalization of the source model for the background dataset at zero so it doesn't contribute. Now we need to set up the background model for both datasets with the appropriate response matrices.

```
XSPEC12>response 2:1 source.rsp 2:2 back.rsp
```

This tells XSPEC that both these datasets have a second model which must be multiplied by the source.rsp response matrix for its contribution to the source region and back.rsp for its contribution to the background region. This is likely to be the case for a standard imaging data where the response will only depend on the extraction region. Note that for a coded-aperture mask the situation may be more complicated with a different response for the source and the background even if they are extracted from the same region of the detector. We now define the background model to be used. In this case take the simple example of a single power-law

```
XSPEC12>model 2:myback pow

Input parameter value, delta, min, bot, top, and max values for ...
              1       0.01(      0.01)         -3         -2          9         10
1:myback:data group 1::powerlaw:PhoIndex>
              1       0.01(      0.01)          0          0      1e+24      1e+24
2:myback:data group 1::powerlaw:norm>

Input parameter value, delta, min, bot, top, and max values for ...
              1       0.01(      0.01)         -3         -2          9         10
3:myback:data group 2::powerlaw:PhoIndex>
              1       0.01(      0.01)          0          0      1e+24      1e+24
4:myback:data group 2::powerlaw:norm>
```

We have now set up XSPEC so that the source data is compared to a source model multiplied by the source response plus a background model multiplied

by the background response and the background data is compared to the background model multiplied by the background response. The background models fitted to the source and background data are constrained to be the same.

```
XSPEC12>show param
Parameters defined:
========================================================================
Model phabs<1>*powerlaw<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
Data group: 1
   1    1   phabs      nH         10^22    1.00000      +/-  0.0
   2    2   powerlaw   PhoIndex            1.00000      +/-  0.0
   3    2   powerlaw   norm                1.00000      +/-  0.0
                           Data group: 2
   4    1   phabs      nH         10^22    1.00000      = 1
   5    2   powerlaw   PhoIndex            1.00000      = 2
   6    2   powerlaw   norm                0.00000      frozen
------------------------------------------------------------------------

========================================================================
Model myback:powerlaw<1> Source No.: 2   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
                           Data group: 1
   1    1   powerlaw   PhoIndex            1.00000      +/-  0.0
   2    1   powerlaw   norm                1.00000      +/-  0.0
                           Data group: 2
   3    1   powerlaw   PhoIndex            1.00000      = myback:1
   4    1   powerlaw   norm                1.00000      = myback:2
------------------------------------------------------------------------
```

It is often the case that the response information is split into an RMF and ARF, where the RMF describes the instrument response and the ARF the telescope effective area. The particle background can then be included by using the RMF but not the ARF:

```
XSPEC12>data 1:1 source.pha 2:2 back.pha
XSPEC12>response 1 source.rmf 2 source.rmf
XSPEC12>arf 1 source.arf
XSPEC12>response 2:1 source.rmf 2:2 source.rmf
```

4.6 Using XSPEC to Simulate Data: an Example for Chandra

In several cases, analyzing simulated data is a powerful tool to demonstrate feasibility. For example:

- *To support an observing proposal.* That is, to demonstrate what constraints a proposed observation would yield.
- *To support a hardware proposal.* If a response matrix is generated, it can be used to demonstrate what kind of science could be done with a new instrument.
- *To support a theoretical paper.* A theorist could write a paper describing a model, and then show how these model spectra would appear when observed. This, of course, is very like the first case.

Here, we will illustrate the first example. The first step is to define a model on which to base the simulation. The way XSPEC creates simulated data is to take the current model, convolve it with the current response matrix, while adding noise appropriate to the integration time specified. Once created, the simulated data can be analyzed in the same way as real data to derive confidence limits.

Let us suppose that we want to measure the intrinsic absorption of a faint high-redshift source using Chandra. Our model is thus a power-law absorbed both by the local Galactic column and an intrinsic column. First, we set up the model. From the literature we have the Galactic absorption column and redshift so:

```
XSPEC12>mo pha*zpha(zpo)

Input parameter value, delta, min, bot, top, and max values for ...
              1      0.001(      0.01)          0          0     100000      1e+06
1:phabs:nH>0.08
              1      0.001(      0.01)          0          0     100000      1e+06
2:zphabs:nH>1.0
              0      -0.01(      0.01)     -0.999     -0.999         10         10
3:zphabs:Redshift>5.1
              1       0.01(      0.01)         -3         -2          9         10
4:zpowerlw:PhoIndex>1.7
              0      -0.01(      0.01)     -0.999     -0.999         10         10
5:zpowerlw:Redshift>5.1
              1       0.01(      0.01)          0          0      1e+24      1e+24
6:zpowerlw:norm>

======================================================================
```

```
Model phabs<1>*zphabs<2>*zpowerlw<3> Source No.: 1   Active/Off
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    8.00000E-02  +/-  0.0
   2    2   zphabs     nH         10^22    1.00000      +/-  0.0
   3    2   zphabs     Redshift            5.10000      frozen
   4    3   zpowerlw   PhoIndex            1.70000      +/-  0.0
   5    3   zpowerlw   Redshift            5.10000      frozen
   6    3   zpowerlw   norm                1.00000      +/-  0.0
```

Now suppose that we know that the observed 0.5–2.5 keV flux is 1.1×10^{-13} ergs/cm^2/s. We now can derive the correct normalization by using the commands **energies**, **flux** and **newpar**. That is, we'll determine the flux of the model with the normalization of unity. We then work out the new normalization and reset it:

```
XSPEC12>energies 0.5 2.5 1000
XSPEC12>flux 0.5 2.5
Model Flux  0.052736 photons (1.0017e-10 ergs/cm^2/s) range (0.50000 - 2.5000 keV)
XSPEC12> newpar 6 1.1e-3
  3 variable fit parameters
XSPEC12>flux
Model Flux 2.6368e-05 photons (5.0086e-14 ergs/cm^2/s) range (0.50000 - 2.5000 keV)
```

Here, we have changed the value of the normalization (the fifth parameter) from 1.0 to 1.1×10^{-13} / $1.00 \times 10^{-10} = 1.1 \times 10^{-3}$ to give the required flux.

The simulation is initiated with the command **fakeit**. If the argument none is given, we will be prompted for the name of the response matrix. If no argument is given, the current response will be used. We also need to reset the **energies** command before the **fakeit** to ensure that the model is calculated on the entire energy range of the response:

```
XSPEC12>energies reset
XSPEC12>fakeit none
For fake data, file #1 needs response file: aciss_aimpt_cy15.rmf
       ... and ancillary response file: aciss_aimpt_cy15.arf
```

There then follows a series of prompts asking the user to specify whether he or she wants counting statistics (yes!), the name of the fake data (file test.fak in our example), and the integration time (40,000 seconds – the correction norm and background exposure time can be left at their default values).

```
Use counting statistics in creating fake data? (y):
 Input optional fake file prefix:
```

```
 Fake data file name (aciss_aimpt_cy15.fak): test.fak
 Exposure time, correction norm, bkg exposure time (1.00000, 1.00000, 1.00000): 40000.0

No background will be applied to fake spectrum #1

1 spectrum  in use


Fit statistic : Chi-Squared =         350.95 using 1024 PHA bins.

***Warning: Chi-square may not be valid due to bins with zero variance
            in spectrum number(s): 1


Test statistic : Chi-Squared =         350.95 using 1024 PHA bins.
 Reduced chi-squared =        0.34407 for   1020 degrees of freedom
 Null hypothesis probability =   1.000000e+00

***Warning: Chi-square may not be valid due to bins with zero variance
            in spectrum number(s): 1

 Current data and model not fit yet.
```

The first thing we should note is that the default statistics are not correct for these data. For Poisson data and no background we should cstat for the fit statistic and pchi for the test statistic:

```
XSPEC12>statistic cstat
Default fit statistic is set to: C-Statistic
   This will apply to all current and newly loaded spectra.

Fit statistic : C-Statistic =         513.63 using 1024 PHA bins and 1020 degrees of freedom.

Test statistic : Chi-Squared =         350.95 using 1024 PHA bins.
 Reduced chi-squared =        0.34407 for    1020 degrees of freedom
 Null hypothesis probability =   1.000000e+00

***Warning: Chi-square may not be valid due to bins with zero variance
            in spectrum number(s): 1

 Current data and model not fit yet.
XSPEC12>statistic test pchi
Default test statistic is set to: Pearson Chi-Squared
   This will apply to all current and newly loaded spectra.

Fit statistic : C-Statistic =         513.63 using 1024 PHA bins and 1020 degrees of freedom.

Test statistic : Pearson Chi-Squared =         639.35 using 1024 PHA bins.
 Reduced chi-squared =         0.62682 for    1020 degrees of freedom
```

```
 Null hypothesis probability =   1.000000e+00

***Warning: Pearson Chi-square may not be valid due to bins with zero model value
            in spectrum number(s): 1

 Current data and model not fit yet.
```

As we can see from the warning message we need to ignore some channels where there is no effective response. Looking at a plot of the data and model indicates we should ignore below 0.15 keV and above 10 keV so:

```
XSPEC12>ignore **-0.15 10.0-**
    11 channels (1-11) ignored in spectrum #     1
   340 channels (685-1024) ignored in spectrum #     1

Fit statistic : C-Statistic =         510.55 using 673 PHA bins and 669 degrees of freedom.

Test statistic : Pearson Chi-Squared =         635.19 using 673 PHA bins.
 Reduced chi-squared =        0.94947 for    669 degrees of freedom
 Null hypothesis probability =   8.217205e-01
 Current data and model not fit yet.
```

We assume that the Galactic column is known so **freeze** the first parameter. We then perform a **fit** followed by the **error** command:

```
XSPEC12> freeze 1
 ...
XSPEC12>fit
 ...
XSPEC12>parallel error 3
XSPEC12>err 2 4 6
 Parameter  Confidence Range (   2.706)
     2      1.16302      5.64805    (-2.00255,2.48247)
     4      1.73345      1.95111    (-0.106137,0.111521)
     6   0.00126229   0.00221906    (-0.000397759,0.000559019)
```

Note that our input parameters do not lie within the 90% confidence errors however since this will happen one times in ten (by definition) this should not worry us unduly. For a real observing proposal we would likely repeat this experiment with different input values of the intrinsic absorption to learn how well we could constrain it given a range of possible actual values.

4.7 Producing Plots: Modifying the Defaults

The final results of using XSPEC are usually one or more tables containing confidence ranges and fit statistics, and one or more plots showing the fits themselves. So far, the plots shown have generally used the default settings, but it is possible to edit plots to improve their appearance.

The plotting package used by XSPEC is PGPLOT, which is comprised of a library of low-level tasks. At a higher level is QDP/PLT, the interactive program that forms the interface between the XSPEC user and PGPLOT. QDP/PLT has its own manual; it also comes with on-line help. Here, we show how to make some of the most common modifications to plots.

In this example, we'll take the simulated Chandra spectrum and make a better plot. Figure 4.9 shows the basic data and folded model plot. The only additional changes we have made to this plot are to increase the line widths to make them print better. We made this plot as follows:

```
XSPEC12>setplot energy
XSPEC12>iplot data
PLT>lwidth 3
PLT>lwidth 3 on 1..2
PLT>time off
PLT>hard figi.png/ps
```

The first **lwidth** command increases the line widths on the frame while the second increases it on the data and model. The **time off** command just removes a username and time stamp from the bottom right of the plot. The **hard** command makes a "hardcopy", in this case a PostScript file. Before looking at other PLT commands we can use to modify the plot there is one more thing we can try at the XSPEC level. The current bin sizes are much smaller than the resolution so we may as well combine bins in the plot (but not in the fitting) to make it clearer.

```
XSPEC12>setplot rebin 100 4
```

Combines four spectral bins into one. The **setplot rebin** command combines bins till either a S/N of the first argument is reached or the number of bins in the second argument have been combined. We do an **iplot** again then do the following modifications:

```
PLT> viewport 0.2 0.2 0.8 0.7
```

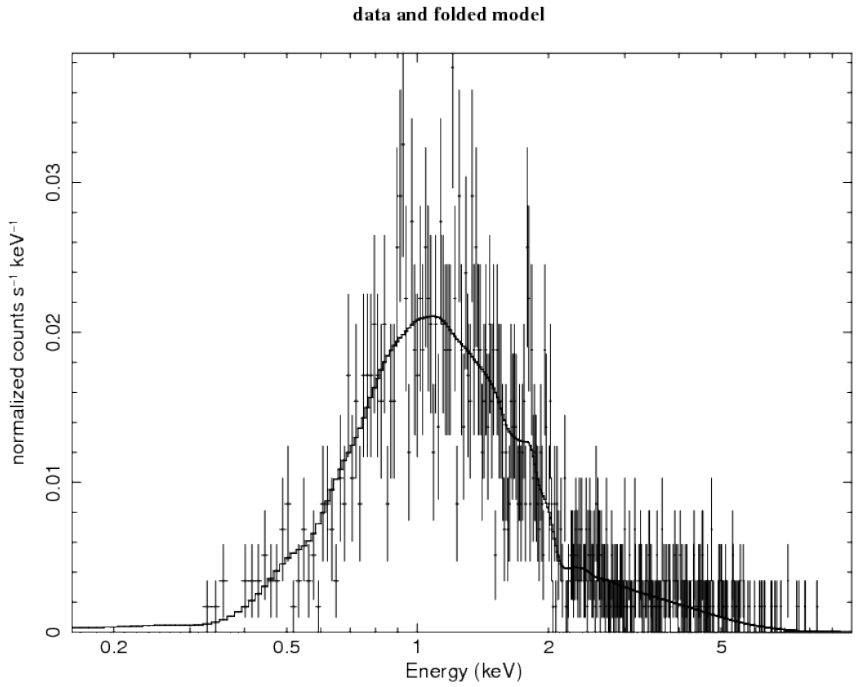

Figure 4.9: The data and folded model for the simulated Chandra ACIS-S spectrum.

The first thing we'll do is change the aspect ratio of the box that contains the plot (viewport in QDP terminology). The viewport is defined as the coordinates of the lower left and upper right corners of the page. The units are such that the full width and height of the page are unity. The labels fall outside the viewport, so if the full viewport were specified, only the plot would appear. The default box has a viewport with corners at (0.1, 0.1) and (0.9, 0.9). For our purposes, we want a viewport with corners at (0.2, 0.2) and (0.8, 0.7): with this size and shape, the hardcopy will fit nicely on the page and not have to be reduced for photocopying.

Next we want to change some of the labels:

```
PLT> label top Simulated Spectrum
PLT> label file Chandra ACIS-S
PLT> label y counts s\u-1\d keV\u-1\d
```

Note the change in the y-axis label is to remove the string "normalized". The normalization referred to is almost always unity so this label can generally be changed. To get help on a PLT command, just type **help** followed by the

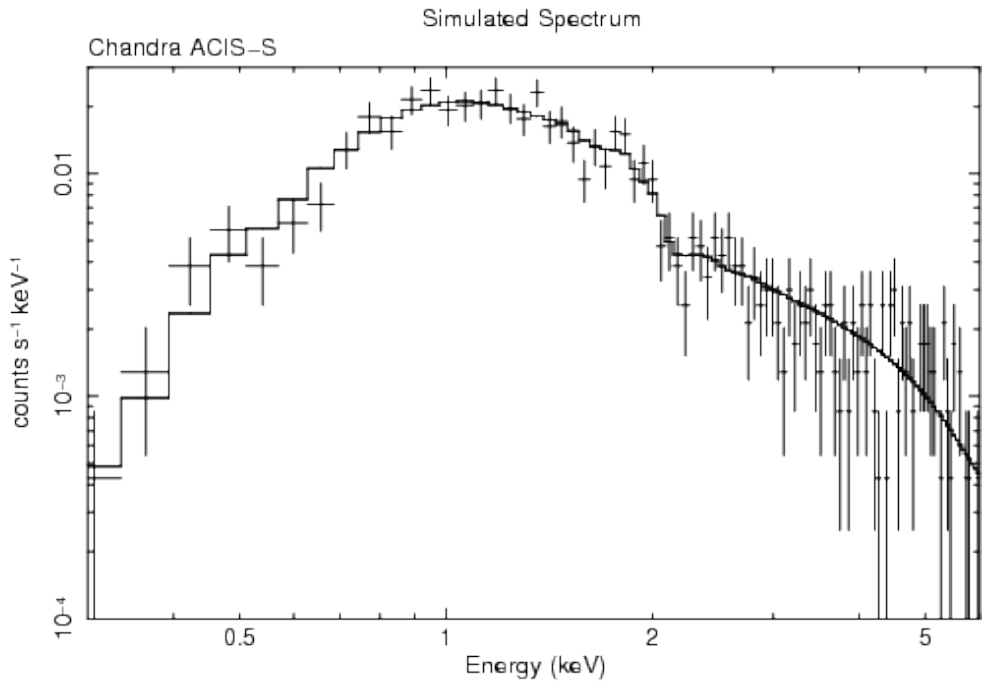

Figure 4.10: A simulated Chandra ACIS-S spectrum produced to show how a plot can be modified by the user.

name of the command. Note that PLT commands can be abbreviated, just like XSPEC commands. To see the results of changing the viewport and the labels, just enter the command **plot**.

The two changes we want to make next are to rescale the axes and to change the y-axis to a logarithmic scale. The commands for these changes also are straightforward: the **rescale** command takes the minimum and maximum values as its arguments, while the **log** command takes x or y as arguments:

```
PLT> rescale x 0.3 6.0
PLT> rescale y 1.0e-4 0.03
PLT> log y
PLT> plot
```

To revert to a linear scale, use the command **log off y**. The only remaining extra change is to reduce the size of the characters. This is done using the **csize** command, which takes the normalization as its argument. One (1) will not change the size, a number less than one will reduce it and a number bigger than one will increase it.

```
PLT> csize 0.8
PLT> plot
```

We make the PostScript file and also save the plot information using the **wenv** command that, in this case, writes files figj.qdp and figj.pco containing the plot data and commands, respectively.

```
PLT> hardcopy figj.png/ps
PLT> wenv figj
PLT> quit
```

The result of all this manipulation is shown proudly in Figure 4.10.

4.8 Markov Chain Monte Carlo Example

To illustrate MCMC methods we will use the same data as the first walkthrough.

```
XSPEC12> data s54405
...
XSPEC12> model phabs(pow)
...
XSPEC12> fit
...
XSPEC12> chain type gw
XSPEC12> chain walkers 8
XSPEC12> chain length 10000
```

We use the Goodman-Weare algorithm with 8 walkers and a total length of 10,000. For the G-W algorithm the actual number of steps are 10,000/8 but we combine the results from all 8 walkers into a single output file. We start the chain at the default model parameters except that we use the **renorm** command to make sure that the model and the data have the same normalization. If we had multiple additive parameters with their own norms then a good starting place would be to use the **fit 1** command to initially set the normalizations to something sensible.

```
XSPEC12> chain run test1.fits
```

The first thing to check is what happened to the fit statistic during the run.

```
XSPEC12> plot chain 0
```

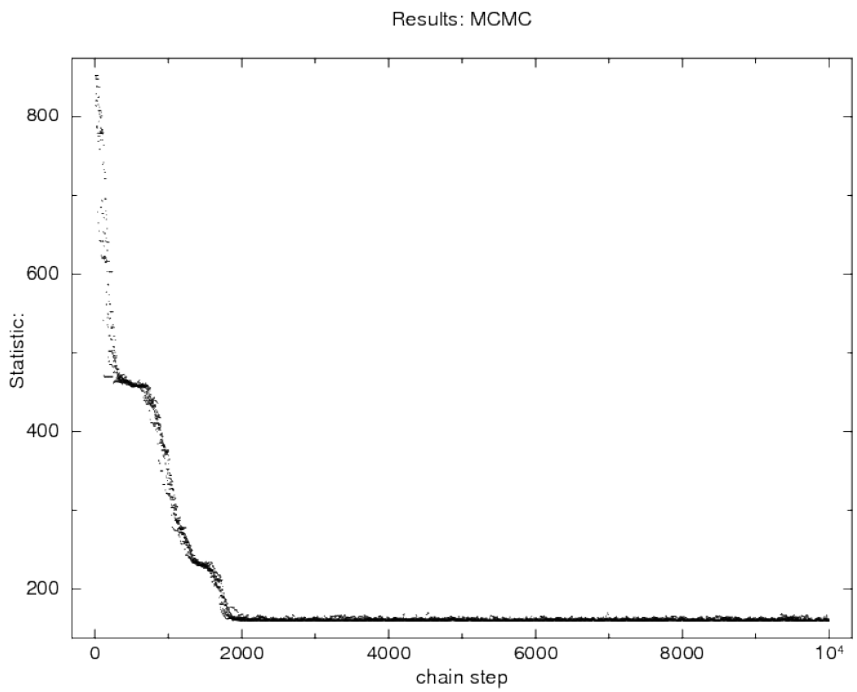

Figure 4.11: The statistic from an MCMC run showing the initial burn-in phase.

The result is shown in Figure 4.11, which plots the statistic value against the chain step. It is clear that after about 2000 steps the chain reached a steady state. We would usually have told XSPEC to discard the first few thousand steps but included them for illustrative purposes. Let us do this again but specifying a burn-in phase that will not be stored.

```
XSPEC12> chain burn 5000
XSPEC12> chain run test1.fits
```

The output chain now comprises 10,000 steady-state samples of the parameter probability distribution. Repeating **plot chain 0** will confirm that the chain is in a steady state. The other parameter values can be plotted either singly using eg **plot chain 2** for the power-law index or in pairs eg **plot chain 1 2** giving a scatter plot as shown in Figure 4.12.

Using the **error** command at this point will generate errors based on the chain values.

```
XSPEC12>error 1 2 3
Errors calculated from chains

 Parameter   Confidence Range (2.706)
     1      0.264971     0.919546
     2        2.1134      2.41307
     3     0.0107304    0.0171814
```

The 90% confidence ranges are determined by ordering the parameter values in the chain then finding the center 90%.

To make confidence regions in two dimensions the **margin** command takes the same arguments as **steppar**.

```
XSPEC12>margin 1 0.0 1.5 25 2 1.5 3.0 25
```

Then use **plot integprob** to produce a plot of confidence regions.

4.9 INTEGRAL/SPI: A Walk Through Example

Consider an observation of the Crab, for which a (standard) 5×5 dithering observation strategy was employed. Since the Crab (pulsar and nebular components are of course un-resolvable at INTEGRAL's spatial resolution) is by

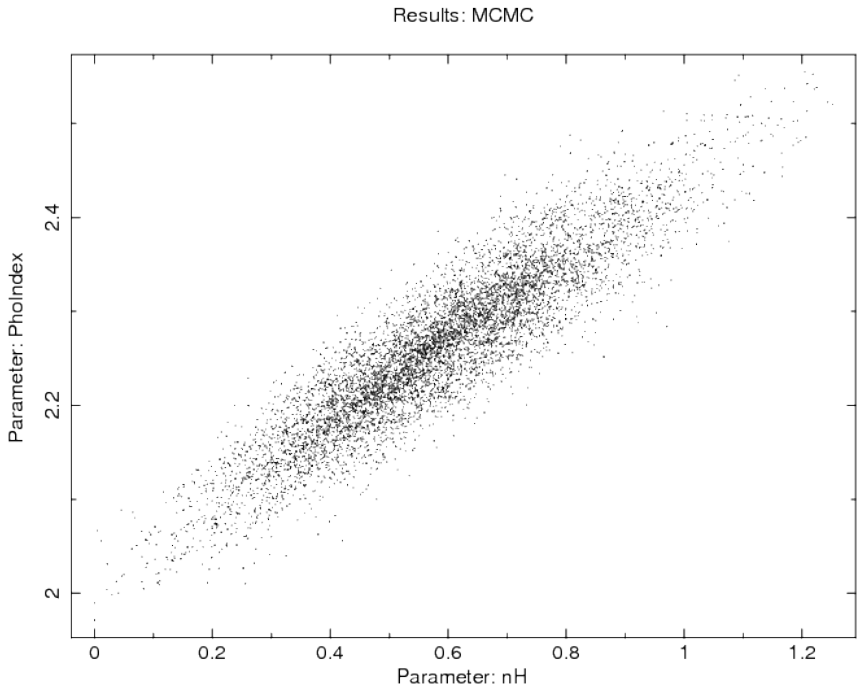

Figure 4.12: The scatter plot from a 10,000 step MCMC run.

far the brightest source in it immediate region of the sky, and its position is precisely known, we can opt not to perform SPI or IBIS imaging analysis prior to XSPEC analysis. We thus run the standard INTEGRAL/SPI analysis chain on detectors 0-18 up to the SPIHIST level for (or BIN_I level in the terminology of the INTEGRAL documentation), selecting the "PHA" output option.

We then run SPIARF, providing the name of the PHA-II file just created, and selecting the "update" option in the spiarf.par parameter file (you should refer to the SPIARF documentation prior to this step if it is unfamiliar). The celestial coordinates for the Crab are provided in decimal degrees (RA,Dec = 83.63,22.01) interactively or by editing the parameter file. This may take a few minutes, depending on the speed of your computer and the length of your observation. Once completed, SPIARF must be run one more time, setting the "bkg_resp" option to "y"; this creates the response matrices to be applied to the background model, and updates the PHA-II response database table accordingly. Then SPIRMF, which interpolates the template RMFs to the users desired spectral binning, also writes information to the PHA response database table to be used by XSPEC. Finally, you should run SPIBKG_INIT, which will construct a set of bbackground spectral templates to initialize the SPI background model currently installed in XSPEC (read the FTOOLS help for that utility carefully your first time). You are now ready to run XSPEC; a sample session might look like this (some repetitive output has been suppressed):

```
%
% xspec

                XSPEC version: 12.2.1
        Build Date/Time: Wed Nov 2 17:14:21 2005

XSPEC12>package require Integral 1.0
1.0
XSPEC12>data ./myDataDir/rev0044_crab.pha{1-19}

19 spectra  in use

RMF # 1
 Using Response (RMF) File            resp/comp1_100x100.rmf
RMF # 2
 Using Response (RMF) File            resp/comp2_100x100.rmf
RMF # 3
 Using Response (RMF) File            resp/comp3_100x100.rmf

Using Multiple Sources

For Source # 1
 Using Auxiliary Response (ARF) Files
     resp/rev0044_100ch_crab_cmp1.arf.fits
```

```
     resp/rev0044_100ch_crab_cmp2.arf.fits
     resp/rev0044_100ch_crab_cmp3.arf.fits

For Source # 2
 Using Auxiliary Response (ARF) Files
     resp/rev0044_100ch_bkg_cmp1.arf.fits
     resp/rev0044_100ch_bkg_cmp2.arf.fits
     resp/rev0044_100ch_bkg_cmp3.arf.fits

Source File: ./myDataDir/rev0044_crab.pha{1}
Net count rate (cts/s) for Spectrum No. 1    3.7011e+01   +/- 1.2119e-01
 Assigned to Data Group No. : 1
 Assigned to Plot Group No. : 1

Source File: ./myDataDir/rev0044_crab.pha{2}
Net count rate (cts/s) for Spectrum No. 2    3.7309e+01   +/- 1.2167e-01
 Assigned to Data Group No. : 1
 Assigned to Plot Group No. : 2
...
Source File: ./myDataDir/rev0044_crab.pha{19}
Net count rate (cts/s) for Spectrum No. 19   3.6913e+01   +/- 1.2103e-01
 Assigned to Data Group No. : 1
 Assigned to Plot Group No. : 19

XSPEC12>mo 1:crab po

Input parameter value, delta, min, bot, top, and max values for ...
    1    PhoIndex              1.0000E+00         1.0000E-02     -3.0000E+00     -2.0000E+00      9.0000E+00
crab::powerlaw:PhoIndex>2.11 0.01 1.5 1.6 2.5 2.6
    2        norm              1.0000E+00         1.0000E-02      0.0000E+00      0.0000E+00      1.0000E+24
crab::powerlaw:norm>8. 0.1 1. 2. 18. 20.
 ...
 XSPEC12>mo 2:bkg spibkg5

Input parameter value, delta, min, bot, top, and max values for ...
    1       Par_1              0.0000E+00         1.0000E-02     -2.0000E-01     -1.5000E-01      1.5000E-01
bkg::spibkg5:Par_1>/*
...
------------------------------------------------------------------------------------------------------

XSPEC12>ign 1-19:68-80
...
XSPEC12>ign 1-19:90-100
...
XSPEC12>fit
Number of trials and critical delta: 10  1.0000000E-02
 ...
=====================================================================
Model bkg:spibkg5 Source No.: 2   Active/On
Model Component Name:  spibkg5  Number:    1
```

```
  N          Name       Unit       Value                 Sigma
    1       Par_1              9.0650E-03       +/-     2.8651E-03
    2       Par_2              1.6174E-02       +/-     3.4778E-03
 ...
   25      Par_25             -1.9537E-02       +/-     6.1429E-03
   26        norm              9.7286E-01       +/-     1.3527E-03
---------------------------------------------------------------------

=====================================================================
Model crab:powerlaw Source No.: 1   Active/On
Model Component Name: powerlaw  Number:    1
  N          Name       Unit       Value                 Sigma
    1    PhoIndex              2.1163E+00       +/-     1.8946E-02
    2        norm              1.1390E+01       +/-     8.1414E-01
---------------------------------------------------------------------

 Chi-Squared =  1.8993005E+03 using 1463 PHA bins.
 Reduced chi-squared =  1.3235544E+00 for   1435 degrees of freedom
 Null hypothesis probability =  1.5268098E-15
XSPEC12>
```

Note that the syntax used for the data statement (appended curly bracket, specifying use of spectra 1-19), and the separate model commands, which are indexed and named (in this case simply "crab" for the source of interest and "bkg" for the background model, "spibkg_lo". These commands are described in detail elsewhere in this document, as are the the spibkg_lo, spibkg_med and spibkg_hi models. In this case, 100 logarithmically-spaced energy bins spanning the nominal 20-8000 keV band of the SPI instrument were used.

In this example, only one dither-point was used to solve for the Crab spectrum, and the background. The simple assumption of a single background spectrum (i.e. no detector-to-detector variations) was assumed. In general, and particularly for fainter sources, a much larger number of spectra will be needed for a solution (and even for the Crab, the quality of the fit, and the accuracy of the inferred parameters can be improved). Also, detector-to-detector and/or time (i.e. pointing-to-pointing) variations will need to be considered. This can be accomplished using the data-grouping feature of XSPEC, which will be described subsequently. Also notice that channels between about 70 and 80 were ignored; this is because there are detector electronic effects contaminating the single-event data for energies from 1250-1400 keV (refer to the SPI data analysis manual for additional discussion), and that there are a lot of (scientifically uninteresting) background model parameters. Also, the highest energies were ignored, since the source flux becomes insignificant relative to the background.

Some results are illustrated below. These plots were generated with the sequence of commands:

```
XSPEC12> setplot group 1-19
XSPEC12> plot ldata res
...
XSPEC12> plot ufspec
```

Note that without the "setplot group" command, XSPEC would plot 19 sets of spectral data, models and residuals. The can become confusing, especially as the number of spectra included in an analysis becomes much larger than 19! On the other hand, it can be useful to divide the data into subsets for plotting purposes, e.g. setplot group 1-6 7-12 13-19, to get an idea of relative shadowing effects of the coded-mask. The left hand plot illustrates the source model, the background model, the total model (i.e. source + background), and the data (here in count rates per channel). The right hand plot illustrates the "unfolded model" (blue, power-law curve), the summed model, and the data as a photon flux. A possible source of confusion is the similarity of the background model curves plotted in theses two separate representations. The explanation is that the background, which is dominated by instrumental contributions, is modeled in detector count space (i.e. the background response matrix has unit effective area. Thus, to be strictly correct, the right-hand plot is a hybrid of the photon source model and the detector-rate background model. We further note that at the present time, XSPEC does not have the capability to plot (or store and manipulate) the background subtracted data. This is a feature under consideration for a future release.

If we had chosen a observation containing more than a single source, the procedure would have been similar, except that the sequence of model commands would be extended, e.g.

```
XSPEC12>data  ./MyDataDir/GCDE_aug_03.pha{1-475}
...
XSPEC12> model 1:1e1740 po
...
XSPEC12> model 2:gx1_4 po
...
XSPEC12> model 3:bkg spibkg_lo
...
```

Here data from the Galactic Center deep exposure campaign were loaded, and two sources are sought. In this case, a much larger number of spectra were loaded (475 spectra corresponds to one full 5×5 dither using all 19 detectors.

In this case, the simple approach of applying constant background (i.e. no detector-to-detector or pointing-to-pointing variation) to the full data set is

likely to be a poor approximation. A more realistic approach would be to use the XSPEC grouping capability to handle such variations in the background solution. This can be accomplished in the usual manner (refer to the description of the grouping command in this document), however, it can become tedious in terms of the required command line inputs. For example, to establish a separate data group for each detector for a long (e.g. 5×5 dither) observations, a sequence of commands such as this would be required:

```
XSPEC12> data   1:1        ./MyDataDir/rev0044_Crab.pha.fits{1}
XSPEC12> data   2:2        ./MyDataDir/rev0044_Crab.pha.fits{2}
XSPEC12> data   3:3        ./MyDataDir/rev0044_Crab.pha.fits{3}
 ...
XSPEC12> data   19:19       ./MyDataDir/rev0044_Crab.pha.fits{19}
XSPEC12> data   1:20        ./MyDataDir/rev0044_Crab.pha.fits{20}
XSPEC12> data   2:21        ./MyDataDir/rev0044_Crab.pha.fits{21}
XSPEC12> data   3:22        ./MyDataDir/rev0044_Crab.pha.fits{22}
 ...
XSPEC12> data   19:38      ./MyDataDir/rev0044_Crab.pha.fits{38}
XSPEC12> data   1:39        /MyDataDir/rev0044_Crab.pha.fits{39}
XSPEC12> data   2:40        ./MyDataDir/rev0044_Crab.pha.fits{40}
XSPEC12> data   3:41        ./MyDataDir/rev0044_Crab.pha.fits{41}
 ...
XSPEC12> data   18:474       ./MyDataDir/rev0044_Crab.pha.fits{474}
XSPEC12> data   19:475       ./MyDataDir/rev0044_Crab.pha.fits{475}
```

One might then for example, make a first cut attempt by fitting a constant background. Then, as a next step, one might allow the normalization terms of the background model to vary over the groups (i.e. over the detector plane). This is accomplished with the "untie" command, using the following sequence:

```
XSPEC12>  untie   bkg:52
XSPEC12>  untie   bkg:78
XSPEC12>  untie   bkg:104
XSPEC12>  untie   bkg:130
XSPEC12>  untie   bkg:156
XSPEC12>  untie   bkg:182
XSPEC12>  untie   bkg:208
XSPEC12>  untie   bkg:234
XSPEC12>  untie   bkg:260
XSPEC12>  untie   bkg:286
XSPEC12>  untie   bkg:312
XSPEC12>  untie   bkg:338
XSPEC12>  untie   bkg:364
XSPEC12>  untie   bkg:390
XSPEC12>  untie   bkg:416
XSPEC12>  untie   bkg:442
XSPEC12>  untie   bkg:468
XSPEC12>  untie   bkg:487
```

Note that use of the "bkg" identifier, which associates the parameters index with the background model. The specific sequence of numbers use here requires some explanation; the particular background model employed has 25 parameters (which simply correspond in rank order to the 25 most variable individual bins), and a normalization term, i.e. parameter 26. Thus, the normalization for the second detector group is parameter 52, for the third parameter 78, and so on. Similar command sequences can be used to untie additional background model parameters. Supposing that we did this and refitted the data. We then might, for example wish to go back and freeze the individual normalization terms with the freeze command:

```
XSPEC12>  freeze  bkg:26
XSPEC12>  freeze  bkg:52
 ...
XSPEC12>  freeze  bkg:487
```

By now though, you probably get the idea that this all requires an unreasonable amount of command-line input. To circumvent this problem, a number of INTEGRAL/SPI specific "tcl" scripts are available which greatly streamline this process.

4.10 INTEGRAL Specific Command Line Scripts

SPIdata

The SPIdata procedure, which when installed can be treated as an XSPEC command, greatly facilitates the data initialization step. For example, the command

```
XSPEC12>  SPIdata  ./MyData/Dir/rev0044_crab.pha 475 det Y
```

Opens the Crab observation spectral data file, reads the 475 spectra into memory, grouping them by detector. The "Y" then indicates that, yes, I wish to ignore the spectral data channels corresponding to the known detector-electronic noise contamination (this is the default). Instead of "det" as the grouping option I could have selected "time" to group by time (quantized into dither-pointing intervals). A "-" lead to the data being initialzed into a single group. The command:

```
XSPEC12>  SPIdata  ./MyData/Dir/rev0044_crab.pha 475
```

Reads the 475 spectra into a single data group, and ignores the undesirable channels. If you forget all this, the command

```
XSPEC12>  SPIdata  -h
```

will remind you. The scripts SPIuntie, and SPIfreeze have similar command-line syntax.

SPIuntie and SPIfreeze

```
XSPEC12>  SPIuntie bkg 475 19 -1
```

The SPIuntie command script will accomplish the same result as the sequence of "untie" commands in the INTEGRAL/SPI example presented in this document. In that case, we had loaded 475 spectra associated with a single 5×5-dither pattern centered on the Crab nebula. The spectra were grouped by detector, which is a common approach to SPI analysis given the known detector-to-detector variations in the background rates. Suppose after an initial fitting pass, for which we assumed a single background spectrum, we know wish to untie the individual data group (i.e. detector) background models. This can be accomplished by issuing 25 "untie" commands as previously noted, or in a single command line using the SPIuntie command:

```
XSPEC12> SPIuntie bkg 475 19 -1

untie bkg:52

 Chi-Squared =  1.2030200E+04 using 1615 PHA bins.
 Reduced chi-squared =  7.5852458E+00 for   1586 degrees of freedom
 Null hypothesis probability =  0.0000000E+00

 untie bkg:78

 Chi-Squared =  1.2030200E+04 using 1615 PHA bins.
 Reduced chi-squared =  7.5900314E+00 for   1585 degrees of freedom
 Null hypothesis probability =  0.0000000E+00

 untie bkg:104
 renorm: no renormalization necessary

 Chi-Squared =  1.2030200E+04 using 1615 PHA bins.
 Reduced chi-squared =  7.5948231E+00 for   1584 degrees of freedom
 Null hypothesis probability =  0.0000000E+00
 ...
```

One might then make a second pass at fitting the data, hopefully leading to improved statistics. Subsequently, additional background model parameters could be untied using the SPIuntie procedure as well. For example, to untie three additional parameters over the full data set, the command syntax is:

```
XSPEC12> SPIuntie bkg 475 19 1 3
...
```

This will untie the first 3 parameters of the background model identified by "bkg", i.e. equivalent to issuing $(475 - 1) \times 3$ individual untie commands. Note that you can always be reminded of the command-line argument definitions by typing "SPIuntie -h" at the XSPEC prompt.

Suppose now that you are satisfied with the relative background normalization terms, and wish to freeze them at their current values for subsequent fitting passes. This could be accomplished using the SPIfreeze command script:

```
XSPEC12> SPIfreeze bkg 475 -1
XSPEC12>SPIfreeze bkg 19 1 -1
freeze bkg:52 1

 Chi-Squared =  6.6232600E+05 using 1805 PHA bins.
 Reduced chi-squared =  3.7589444E+02 for   1762 degrees of freedom
 Null hypothesis probability =  0.0000000E+00

freeze bkg:78

 Chi-Squared =  6.5791894E+05 using 1805 PHA bins.
 Reduced chi-squared =  3.7318148E+02 for   1763 degrees of freedom
 Null hypothesis probability =  0.0000000E+00

 ...
```

As with the SPIuntie command script, typing "SPIfreeze -h" at the XSPEC prompt will scroll the command-line definitions to your screen.

Note that the current SPI background models, which are documented elsewhere, are designed so that the parameter list is hierarchically ordered in terms of decreasing criticality. Thus, freeing the first parameter is likely to have the most significant impact on the statistics, the second parameter, the next most significant, and so on.

Chapter 5

XSPEC Commands

5.1 Summary of Commands

The following is a list of the commands available in XSPEC, together with a brief description of the purpose of each. The commands have been categorized under six headings: Control, Data, Fit, Model, Plot, Script, and Setting. The Control commands contain the interface with the operating system: they cause commands to be executed, or user input written to disk, or control how much is output. The Data commands manipulate the data being analyzed, by reading data into the program or replacing spectra or their ancillary detector, background, correction, or efficiency (auxiliary response) arrays. Additionally data commands control the channels under analysis. The fit commands invoke the fitting routines, modify their behavior by interchanging fitting algorithms or statistics in use, fixing parameters, or perform statistical testing. The Model commands create or manipulate the model, adding or editing components, modifying parameters, or alternatively performing analytical calculations from a model. The Plot commands deal with all aspects of plotting. The scripts are auto-loaded Tcl scripts that can be used in the same ways as commands. Finally the Setting commands sets variables that affect theoretical models.

Command	Category	Description
abund	`SETTING`	Set the abundance table.
addcomp	`MODEL`	Add a component to the model.
addline	`MODEL`	Add lines to a model
arf	`DATA`	Read an auxiliary response file.

autosave	CONTROL	Periodically save the XSPEC status.
backgrnd	DATA	Reset the files to be used for background subtraction.
bayes	FIT	Set up for Bayesian inference.
chain	FIT	Run a Monte Carlo Markov Chain.
chatter	CONTROL	Control the verbosity of XSPEC.
corfile	DATA	Reset the files to be used for background correction.
cornorm	DATA	Reset the normalization to be used in correcting the background.
cosmo	SETTING	Set H_0, q_0, and Λ_0
cpd	PLOT	Alias for **setplot device**.
data	DATA	Input one or more PHA data files.
delcomp	MODEL	Delete a component from the model.
diagrsp	DATA	Diagonalize the current response for an ideal response.
dummyrsp	MODEL	Create a dummy response, covering a given energy range.
editmod	MODEL	Add, delete, or replace one component in the model.
energies	MODEL	Specify new energy binning for model fluxes.
eqwidth	MODEL	Calculate a model component˘2019s equivalent width.
error (**rerror**)	FIT	Determine a single parameter confidence region. rerror is for response parameters.
exec	CONTROL	Execute a shell command from within XSPEC.
exit	CONTROL	Wind up any hardcopy plots and exit from XSPEC.
extend	MODEL	This is now obsolete. See **energies** command.
fakeit	DATA	Produce simulated data files for sensitivity studies.
fit	FIT	Find the best fit model parameters.
flux	MODEL	Calculate the current model's flux over an energy range.
freeze (**rfreeze**)	FIT	Do not allow a model parameter to vary during the fit. rfreeze is for response parameters.
ftest	FIT	Calculate the F-statistic between two model fits
gain	MODEL	Perform a simple modification of the response gain.
goodness	FIT	Monte Carlo calculation of goodness-of-fit.

hardcopy	PLOT	Spool the current plot to the printer.
help	CONTROL	Obtain help on XSPEC commands.
identify	MODEL	List possible lines in the specified energy range.
ignore	DATA	Ignore a range of PHA channels in future fit operations.
initpackage	MODEL	Compile, build, and initialize a package of local models.
iplot	PLOT	As plot command but interactive using PLT.
lmod	MODEL	Load a package of local models.
log	CONTROL	Open the log file to save output.
lrt	SCRIPT	Likelihood ratio test between two models.
lumin	MODEL	Calculate the current model's luminosity over a given rest frame energy range and redshift.
margin	FIT	MCMC probability distribution.
mdefine	MODEL	Define a simple model using an arithmetic expression.
method	SETTING	Set the minimization method.
model (**rmodel**)	MODEL	Define the model to be used when fitting the data.
modid	MODEL	Guess line IDs in the model.
multifake	SCRIPT	Perform many iterations of fakeit and save the results in a FITS file.
newpar (**rnewpar**)	MODEL	Modify the model parameters. rnewpar is for response parameters
notice	DATA	Restore a range of PHA channels for future operations.
parallel	CONTROL	Enable parallel processing for particular tasks in XSPEC.
plot	PLOT	Plot various information on the current plot device.
query	CONTROL	Switch on/off prompt to continue fitting.
quit	CONTROL	An alias for **exit**
renorm	FIT	Adjust the model norms, and/or control automatic renorming.
rescalecov	SCRIPT	Rescale the covariance matrix used in the proposal chain command.

response	DATA	Reset the files used to determine the detector responses.
save	CONTROL	Save aspects of the current state to a command file.
script	CONTROL	Open the script file to save all commands input.
setplot	PLOT	Modify the plot device and other values used by the plot routines.
show	CONTROL	Display current file and model information.
simftest	SCRIPT	Generate simulated datasets to estimate the F-test probability for adding a model component.
source	CONTROL	Execute a script file.
statistic	SETTING	Change the fit statistic in use.
steppar	FIT	Step through a range of parameter values; perform a fit at each step.
syscall	CONTROL	Run a shell command.
systematic	MODEL	Set the model systematic error.
tclout	CONTROL	write xspec data to a tcl variable
tcloutr	CONTROL	tclout with return value
thaw (**rthaw**)	FIT	Allow a model parameter to vary during the fit. rthaw is for response parameters
time	CONTROL	Display elapsed time and other statistical information.
uncertain	FIT	Alias for error
untie (**runtie**)	MODEL	Untie linked parameters. runtie is for response parameters.
version	CONTROL	Print XSPEC version and build date/time
weight	FIT	Change the weighting function used for chi-squared fits.
writefits	SCRIPT	Write information about the current fit and errors to a FITS file.
xsect	SETTING	Change the photoelectric absorption cross-sections in use.
xset	SETTING	Modify a number of XSPEC internal switches

5.2 Description of Syntax

The individual commands are treated in alphabetical order in the following section. The novice would be well-served by reading the treatments of the **data**, **model**, **newpar**, and **fit** commands, in t hat order, then the other commands as needed. The write-up for each command includes a brief description of the purpose, an outline of the correct syntax, a more detailed discussion of the command assumptions and purpose, and a series of examples. Some commands have one or more subcommands that are similarly described following the command.

In the command description, the syntax uses the following conventions.

`<arg>`	an argument to the command
`<arg c> =::<arg a> <arg b>`	defines `<arg c>` as `<arg a>` followed by `<arg b>`
`<arg>...`	a repeated string of arguments of the same type
`[<arg>]`	is an optional argument
`<arg a> |<arg b>`	indicates a choice between an argument of `<arg a>` or `<arg b>`

Exceptional responses to the command prompt are :

empty line or line containing only spaces and tab characters	nothing performed, prompt repeated
/	any remaining arguments will have the values given on the last invocation of the command
<EOF> (Ctrl-D on Unix)	same as quit
/*	skip input and return to prompt. Defaults for prompted values will be used.
?, ?command, command ?	Print list of commands, or summary help for a single command

5.3 Control Commands

5.3.1 autosave

set frequency of saving commands

Set or disable autosave, which saves the XSPEC environment to a file periodically.

Syntax: **autosave** <`option`>

where <`option`> is either `off` or a non-zero positive integer N. If the option is `off`, then auto-saving is disabled. If the option is N, the XSPEC environment is saved every N commands. The saving of the environment is equivalent to the command

```
XSPEC12>save all xautosav.xcm .,
```

i.e. both the file and model information is saved to the file xautosav.xcm, placed in the directory /.xspec/cache. Thus in case of an unexpected crash, the state of XSPEC before the crash can be restored by running @xautosav.xcm. The default value for the auto-save option is 1.

5.3.2 chatter

set verboseness level

Control the verbosity of XSPEC.

Syntax: **chatter** <`chatter level`> <`log chatter`>

where <`chatter level`> and <`log chatter`> are integer values.The initial value for each argument is 10. Higher values will encourage XSPEC to tell the user more, lower values tell the user less, or make XSPEC "quieter" <`chatter level`> applies to the terminal output, while <`log chatter`> controls the verbosity in the log file. Currently, the maximum chattiness is 25.Values below five should be avoided, as they tend to make XSPEC far too obscure.Some commands may temporarily modify the chattiness, such as the **error** command. A chattiness of 25 will generate a lot of debug output.

Examples:

```
XSPEC12> chatter 10
// Set the terminal chattiness to 10, same as the initial value.
XSPEC12> chatter ,0
//Set the chattiness for the log file to very low.
//This setting essentially disables the log file output.
XSPEC12> chatter 5
//Make XSPEC very quiet.
XSPEC12> chatter 10 25
// Restore the terminal chattiness to the initial level,
// while in the log file XSPEC will tell all
// (particularly when new data files are read in)
```

5.3.3 exit, quit

exit program

The command to end the current XSPEC run.

Syntax: **exit**

After an **exit**, the current plot files are closed. An <EOF> will have an identical result.

5.3.4 help

display manual or help for a specific command/theoretical model component

Obtain help on the XSPEC commands, their syntax, and examples of their use.

Syntax: **help** `[<topic list>]`

On the first invocation of the help command, an instance of a pdf file reader (by default Adobe Acrobat Reader) is started (a shortly delay may ensue), or the XSPEC manual is accessed online. Please see the subsection "Customizing XSPEC" in the XSPEC Overview section for details on how to control this behavior. The Acrobat reader must be in the user's path. If this default is used, then subsequent calls to help will use this instance to display other help pages. help without arguments displays the XSPEC manual, with a bookmark index that allows random access to the help system, or in the online mode will open to the XSPEC manual homepage.

The design allows for users to add help files for local models and scripts to the help system if they are placed in the help search path.

Examples:

```
XSPEC12> help
//show the entire manual.
XSPEC12> help fit
//Go to the help text for the fit command.
XSPEC12> help model pow
//Go to the help text for the powerlaw model. (Entering just "XSPEC12> model" will proc
XSPEC12> help appendices
//show the manual appendices (which document the user interface, the Cash statistic, ho
XSPEC12>help appendix local
//show the appendix describing how to add local models
```

Help also displays the following information as scrolling text:

```
XSPEC12> help ?
//Show a list of all available commands.
XSPEC12> help ??
//Show a brief summary and usage syntax of all available commands.
XSPEC12> <command> ?
// Show brief summary and syntax of <command>.
```

5.3.5 log

log the session output

Open a log file.

Syntax: **log** `[STAMP] <log file>`

where `<log file>` is the name of the file to be opened (default extension is `.log`).If no arguments are on the line, then the default file name is `xspec.log`. If `<log file>` matches the string `none`, then the current log file is closed. If the string `STAMP` is given as an argument then the log filename will include a data and time stamp. If `<log file>` has no suffix then the stamp is appended to the name and a `.log` suffix added. To change the chattiness level for the log file (ie. the amount of information written to the log file) use the **chatter** command. The default chatter level for the log file is 10.

Examples:

```
XSPEC12> log
//Turn on the log file (default xspec.log).
XSPEC12> log none
//Close the log file.
XSPEC12> log mylog
//Open the log file (mylog.log)
XSPEC12> chatter ,, 12
//Set the log file chattiness to 12.
```

5.3.6 parallel

enable parallel processing for particular tasks in XSPEC.

Syntax: **parallel** <task> <max num of processes>

where <task> is currently limited to `leven`, `error`, `steppar`, `goodness`, or `walkers`. For best results, it is recommended that you set <max num of processes> to the number of CPU cores on your machine. Set <max num processes> back to 1 to turn parallel processing off for the particular task. To display current settings, type 'parallel' with no arguments.

The `leven` option will spawn up to <max num> processes during the Levenberg-Marquardt fitting, specifically to perform the N independent calculations of the parameter first-order partial derivatives (N being the number of variable fit parameters). [This will not apply if the USE_NUMERICAL_DIFFERENTIATION variable in the user's Xspec.init file is changed from the default 'false' to 'true'.]

The speed-up that one can expect is highly dependent upon the model in use. For simpler models with quick calculation times, you will probably see little to no speed gain with **parallel leven**. But with multi-core CPUs, gains should be quite noticeable when the model calculation consumes a large fraction of the overall fitting time. For example, with fits using the time-intensive **sedov** model on a 4-core machine, we've typically seen about a 40% reduction in fit time compared with the single processing case.

The `error` option is for running parallel computations within XSPEC's **error** command. This enables the error calculations for multiple parameters to be performed simultaneously. The speed-up here should simply be proportional to the number of cores available. However for cases where complications are reported (such as a new minimum found, or a non-monotonicity in the statistic space), it is recommended that you perform the error calculations in standard

single-process mode.

When the `steppar` option is set, XSPEC will divide the N-dimensional steppar grid into <`max num`> sections of equal size, and spawn a separate process for calculating each section.

If both **parallel** `leven` and `error` or `steppar` are in use, XSPEC will temporarily disable the lower-level`leven` parallelization when running the higher-level parallel `error` or `steppar` command calculations.

The `goodness` option is for parallelizing the calculations of the simulated spectra during a **goodness** command run. The `walkers` option may be used to speed up the calculation of sets of `walkers` needed for Monte Carlo Markov Chain runs using the Goodman-Weare algorithm (see **chain**).

Examples:

```
XSPEC12> model cflow
// Using a model with 5 variable fit parameters.

XSPEC12> parallel leven 4
XSPEC12> fit
// Calculations for the 5 parameters will be divided amongst
// 4 processes during the fit.

XSPEC12> parallel leven 1
// Restores single-process calculation to the
// Levenberg-Marquardt algorithm.

XSPEC12> parallel error 3
// Allow up to 3 simultaneous 'error' parameter calculations
// to be performed in parallel.

XSPEC12> error 2 3 6
// Perform error calculations on parameters 2, 3, and 6 in parallel.

XSPEC12> parallel steppar 4
      // The following 20x30 steppar grid will be split amongst
      // 4 parallel processes.
XSPEC12> steppar 1 10. 11. 20 2 .5 .8 30

// Display current settings:
XSPEC12> parallel
Maximum number of parallel processes:
   error: 3
```

```
goodness: 1
leven: 1
steppar: 4
walkers: 1
```

5.3.7 query

set a default answer for prompts in scripts

Switch on/off the continue fitting questions.

Syntax: **query** `<option>`

where `<option>` is `yes`, `no`, or `on`. If `on` then the continue fitting question in **fit**, **steppar**, and **error** will be asked when the number of trials is exceeded. Also, when the number of trials to find the error is exceeded a question will be asked. For either of the other two options the questions will not be asked but the answer will be assumed to be `yes` or `no` depending on the value set. To ensure that fitting continues without any questions being asked use the command:

```
XSPEC12> query yes
```

5.3.8 save

save the current session commands

Save aspects of the current state to a command file.

Syntax: **save** `<option>` `<filename>`

If no `<filename>` is given, then the file `savexspec.xcm` is created. If you don't give the extension to the file name the default is `.xcm`. The values of `<option>` allowed are `model`, `files`, and `all`. The `model` option writes out commands to recreate the current model and parameter values; the `files` option writes out commands to read-in the current spectra, and the `all` option does both of the above. The default option is `model`. To recover the saved context use the command:

```
XSPEC12>@filename
```

Examples:

```
XSPEC12> save model fname
// Write out model commands to the file fname.xcm
XSPEC12> save
// Same as above, but save into file savexspec.xcm.
XSPEC12> save files fname
// Write out data file commands.
```

5.3.9 script

write commands to a script file

Open a script file.

Syntax: **script** `<script file>`

where `<script file>` is the name of the file to be opened (default extension is .xcm). If no arguments are on the line, then the default file name is `xspec.xcm`. If `<script file>` matches the string `none`, then the current script file is closed. The script file saves all commands that are input. This command is useful for users who use the same set of commands repeatedly. Once a script file is written and saved, the user then can re-run the same set of commands on other data by:

```
XSPEC12> source <script file>
```

Examples:

```
XSPEC12> script
// Turn on the script file (default xspec.xcm)
XSPEC12> script none
// Close the script file.
XSPEC12> script myscript
// Open the script file (myscript.xcm)
```

5.3.10 show

output current program state

List selected information to the user's terminal (and the log file, if open).

Syntax: **show** [<selection>]

where <selection> is a key word to select the information to be printed. If omitted, it is the information last asked for. Initially, the default selection is all. (Note: to better integrate the usage of OGIP type-II files, much of the information given by **show files** in previous versions is now displayed by **show data**.)

Selections are:

abund Show current solar abundance table.

all All the information.

allfile All file information = files + noticed + rates.

control XSPEC control information.

data File names, associated coefficients, and net countrates, displayed in order of spectrum number. For higher chatter, also displays grouping map.

free Free parameters.

files Equivalent to **show data** but displayed in order of file name.

fit Fit information.

model The model specification.

noticed Channel ranges noticed for each file.

parameters All current parameter values (including gain parameters, if any). Adding <par range> shows the subset of all model parameters given by <par range>, e.g. **show parameters 1,3,5-8**.

pha Current data, error and model values for each channe.

plot Current plot settings from setplot command, include rebinning info.

rates Folded model, correction rates for each file.

response Show responses loaded.

rparameters All current gain (response) parameters. Adding <par range> shows the subset of all response parameters given by <par range>.

version Show the XSPEC version number.

xsect Show description of cross-section table

5.3.11 syscall

execute a shell command

Execute command in a shell.

Syntax: **syscall** [<shell command>]

This command executes its arguments by passing them to the users current shell for execution. Thus file name globbing (i.e. wildcard expansion) are performed on the command before execution. This is in contrast to the **exec** command, which executes commands directly, without first passing them on to a shell.

If no arguments are given, then the command will start an interactive subshell.

5.3.12 tclout

create tcl variables from current state

Write internal xspec data to a tcl variable. This facility allows the manipulation of xspec data by tcl scripts, so that one can, for example, extract data from xspec runs and store in output files, format xspec output data as desired, use independent plotting software, etc.

Syntax: **tclout** <option> [<par1>] [<par2>] [<par3>]

tclout creates the tcl variable `$xspec_tclout`, which can then of course be set to any named variable. The allowed values of <option> are:

?	Show the valid options. Does not set $xspec_tclout.
areascal n <s|b>	Writes a string of blank separated values giving the AREASCAL values for spectrum n. If no second argument is given or it is "s" then the values are from the source file, if "b" from the background file.
arf n	The auxiliary response filename(s) for spectrum n.
backgrnd n	Background filename for spectrum n.
backscal n <s|b>	Same as areascal option but for BACKSCAL value.

`chain best|last|proposal|stat`	The best option returns the parameter values corresponding to the smallest statistic value in the loaded chains. The last option returns the final set of parameter values in the loaded chains. The proposal option takes arguments distribution or matrix and returns the name or covariance matrix for the proposal distribution when using Metropolis-Hastings. The stat option returns the output of the last chain stat command.
`chatter`	Current xspec chatter level.
`compinfo [<mod>:]n [<group n>]`	Name, 1st parameter number and number of parameters of model component n, belonging to model w/ optional name <mod> and optional datagroup <group n>.
`cosmo`	Writes a blank separated string containing the Hubble constant (H0), the deceleration parameter (q0), and the cosmological constant (Lambda0). Note that if Lambda0 is non-zero the Universe is assumed to be flat and the value of q0 should be ignored.
`covariance [m,n]`	Element (m,n) from the covariance matrix of the most recent fit. If no indices are specified, then entire covariance matrix is retrieved.
datagrp [n]	Data group number for spectrum n. If no n is given, outputs the total number of data groups.
`datasets`	Number of datasets.
`dof`	Degrees of freedom in fit, and the number of channels.
`energies [n]`	Writes a string of blank separated values giving the energies for spectrum n on which the model is calculated. If n is not specified or is 0, it will output the energies of the default dummy response matrix.
`eqwidth n [errsims]`	Last equivalent width calculated for spectrum n. If `errsims` keyword is supplied, this will instead return the complete sorted array of values generated for the most recent eqwidth error simulation.

`error [<mod>:]n` (for gain parameters use: rerror [<sourceNum>:]n)	Writes last confidence region calculated for parameter n of model with optional name <mod>, and a string listing any errors that occurred during the calculation. The string comprises nine letters, the letter is T or F depending on whether or not an error occurred. The 9 possible errors are: 1. new minimum found 2. non-monotonicity detected 3. minimization may have run into problem 4. hit hard lower limit 5. hit hard upper limit 6. parameter was frozen 7. search failed in -ve direction 8. search failed in +ve direction 9. reduced chi-squared too high So for example an error string of "FFFFFFFFT" indicates the calculation failed because the reduced chi-squared was too high.
`expos n <s|b>`	Same as areascal option but for EXPOSURE value.
`filename n`	Filename corresponding to spectrum n.
`fileinfo <keyword> n`	Value of keyword in the SPECTRUM extension of the file corresponding to spectrum n.
`flux [n][errsims]`	Last model flux or luminosity calculated for spectrum n. Writes a string of 6 values: val errLow errHigh (in ergs/cm2) val errLow errHigh (in photons). Error values are .0 if flux was not run with `errsims` option. If the `errsims` keyword is supplied, this will instead return the completed sorted array of values generated during the most recent flux error calculation.
`ftest`	The result of the last ftest command.
`gain [<sourceNum>:]<specNum> slope|offset`	For gain fit parameters: value, delta, min, low, high, max for the slope or offset parameter belonging to the [<sourceNum>:]<specNum> response. For non-fit gain parameters, only the value is returned.

goodness [sims]	The percentage of realizations from the last goodness command with statistic value less than the best-fit statistic using the data. If optional `sims` keyword is specified, this will instead give the full array of simulation values from the last goodness command.
`idline e d`	Possible line IDs within the range [e-d, e+d].
`ignore [<n>]`	The range(s) of the ignored channels for spectrum n.
`lumin [n] [errsims]`	Last model luminosity calculated for spectrum n. Same output format as flux option, in units of $1.0 \times 10^{44} erg/s$.
margin probability|[<modName>:]<parNum>	The probability option returns the probability column respectively from the most recent margin command. Otherwise, the parameter column indicated by <parNum> is returned. Note that for multi-dimensional margin the returned parameter column will contain duplicate values, in the same order as they originally appeared on the screen during the margin run.
`model`	Description of current model(s).
`modcomp [<mod>]`	Number of components in model (with optional model name).
`modpar [<mod>]`	Number of model parameters(with optional model name).
`modval [<specNum> [<modName>]]`	Write to Tcl the last calculated model values for the specified spectrum and optional model name. Writes a string of blank separated numbers. Note that the output is in units of $photons/cm^2/s/bin$.
`nchan [<n>]`	Total number of channels in spectrum n (including ignored channels).
`noticed [<n>]`	Range (low,high) of noticed channels for spectrum n.
noticed energy [<n>]	The noticed energies for spectrum n.
`nullhyp`	When using chi-square for fits, this will retrieve the reported null hypothesis probability.
`param [<mod>:]n`	(value, delta, min, low, high, max) for model parameter n.
`peakrsid n [lo,hi]`	Energies and strengths of the peak residuals (+ve and -ve) for the spectrum n. Optional arguments lo, hi specify an energy range in which to search.

`pinfo [<mod>:]n`	Parameter name and unit for parameter n of model with optional name.
`plink [<mod>:]n`	Information on parameter linking for parameter n. This is in the form true/false (T or F) for linked/not linked, followed by the multiplicative factor and additive constants if linked.
`plot <option> <array> [<plot group n>]`	Write a string of blank separated values for the array. <option> is one of the valid arguments for the plot or iplot commands. <array> is one of x, xerr, y, yerr, or model. xerr and yerr output the 1-sigma error bars generated for plots with errors. The model array is for the convolved model in data and ldata plots. For contour plots this command just dumps the steppar results. The command does not work for genetic plot options.
`plotgrp`	Number of plot groups.
`query`	The setting of the query option.
`rate <n|all>`	Count rate, uncertainty and the model rate for the specified `spectrum n`, or for the sum over all spectra.
`rerror [<sourceNumber>:]n`	Writes last confidence region calculated for response parameter n of model with optional source number, and a string listing any errors that occurred during the calculation. See the help above on the error option for a description of the string.
`response n`	Response filname(s) for the spectrum n.
`sigma [<modelName>:]n`	The sigma uncertainty value for parameter n. If n is not a variable parameter or fit was unable to calculate sigma, -1.0 is returned.
`simpars`	Creates a list of parameter values by drawing from a multivariate Normal distribution based on the covariance matrix from the last fit. This is the same mechanism that is used to get the errors on fluxes and luminosities, and to run the goodness command.
`solab`	Solar abundance table values.
`stat [test]`	Value of statistic. If optional `test` argument is given, this outputs the test statistic rather than the fit statistic.

statmethod [test]	The name of the fit stat method currently in use. If optional `test` argument is given, this will give the name of the test stat method.
`steppar statistic` | `delstat` | `[<modName>:]<parNum>`	The `statistic` and `delstat` options return the statistic or delta-statistic column respectively from the most recent steppar run. Otherwise, the parameter column indicated by <parNum> is returned. Note that for multi-dimensional steppars the returned parameter column will contain duplicate values, in the same order as they originally appeared on the screen during the steppar run.
`varpar`	Number of variable fit parameters.
`version`	The XSPEC version string.
`weight`	Name of the current weighting function.
`xflt n`	XFLT#### keywords for spectrum n. The first number written is the number of keywords and the rest are the keyword values.

Examples:

```
XSPEC12>data file1
XSPEC12> model pha(po)
...
XSPEC12> fit
...
XSPEC12>tclout stat
XSPEC12>scan $xspec_tclout "%f" chistat
XSPEC12>tclout param 1
XSPEC12>scan $xspec_tclout "%f"par2
XSPEC12>tclout param 2
XSPEC12>scan $xspec_tclout "%f"par3
XSPEC12>tclout param 3
```

In this example, `scan` is a tcl command that does a formatted read of the variable $xspec_tclout. It reads the first floating point number into the variable given by the last argument on the line.This sequence creates a simple model, fits it, and then writes the χ^2statistic and the three parameters to tcl variables $chistat, $par1, $par2, and $par3. These can now be manipulated in any way permitted by tcl. Examples of using tclout and tcloutr can be found in the Xspec/src/scripts directory.

5.3.13 tcloutr

tclout with return value

Syntax: **tcloutr** `<option> [<par1>] [<par2>] [<par3>]`

tcloutr is identical to the **tclout** command except that it also provides what is stored in $xspec_tclout as a return value. Therefore it can be used in tcl scripts like this:

```
set var1 [tcloutr energies 1]
```

tcloutr may produce quite a lot of unwanted screen output, which can be avoided by using **tclout**.

5.3.14 time

print execution time

Get some information about the program run time.

Syntax: **time**

The time command prints out elapses CPU time attributed to the user and to the system. Two output lines are given, one for user/system time since the time command was last called, and one for time elapsed since the program started.

5.3.15 undo

undo the previous command

Undo the affects of the previously entered xspec command.

Syntax: **undo**

New for xspec version 12, the **undo** command will restore the state of the xspec session prior to the most recently entered command. The current implementation does not allow restoration to more than one command back, so calling **undo** repeatedly will have no effect. Also, a **plot** command cannot be undone.

5.3.16 version

print the version string

Syntax: **version**

version prints out the information about version number and build date and time (not current date, time) displayed when XSPEC is started.

5.4 Data Commands

5.4.1 arf

change the efficiency file for a given response

Read in one or more auxiliary response files (ARF). An ARF gives area versus energy and is used to modify the response matrix for a spectrum. The file must be in the OGIP standard format.

Syntax: **arf** `[<filespec>...]`

where `<filespec>` =:: [[`<source #:>`]`<spectrum num>`] `<filename>`[{ranges}]... and where `<spectrum num>` is the spectrum number for the first `<filename>` specified, `<spectrum num>` plus one is the spectrum number for the next file (or next entry in ranges specifier for Type II multi-ARF files), and so on. `<filename>` is the name of the auxiliary response file to be used with the associated spectrum. The optional source number defaults to 1, and for ARFs stored in OGIP Type II files, {ranges} specifies the row numbers of the desired ARF(s). See the **data** command for allowed range specification.

If no `<spectrum num>` is given in the first `<filespec>` it is assumed to be 1. If no file specifications are entered, then none of the spectrum responses are modified.An error message is printed if the spectrum number is greater than the current number of spectra (as determined from the last use of the **data** command). A file name `none` indicates that no auxiliary response is to be used for that spectrum. If a file is not found or cannot be opened for input, then the user is prompted for a replacement auxiliary response file. An <EOF> at this point is equivalent to `none`. See the **data** command for ways to completely remove the dataset from consideration.

Note: The **arf** command is currently not implemented for data formats which use multiple RMFs per spectrum, such as Integral/SPI data.

Examples:

It is assumed that there are currently three spectra:

```
XSPEC> arf a,b,c
// New files for the auxiliary response are given for all three spectra.
XSPEC> arf 2 none
// No auxiliary response will be used for the second spectrum.
XSPEC> arf ,d.fits
// d.fits becomes the auxiliary response for the second spectrum.
XSPEC> arf 2 e.fits{3-4}
// Rows 3 and 4 of multi-ARF file e.fits become the auxiliary responses for the second
XSPEC> arf 2:1 f.fits
// f.fits becomes the auxiliary response for the second source of spectrum 1.
```

5.4.2 backgrnd

change the background file for a given spectrum

Modify one or more of the files used in background subtraction.

Syntax: **backgrnd** [<filespec>...]
backgrnd <spectrum number> none

where <filespec> =:: [<spectrum num>] <filename>... and where <filename> is the name of the PHA file to be used for background subtraction. The numbering scheme is as described for the **data** command, except that the <spectrum num> must have previously been loaded.

An error message is printed if <spectrum num> is greater than the current number of spectra (as determined from the last use of the **data** command). **backgrnd** <n> none indicates that no background subtraction is to be performed for that spectrum. If a file is not found or cannot be opened for input, then the user is prompted for a replacement background file (an <EOF> at this point is equivalent to **backgrnd** <spectrum number> none). The current ignore status for channels is not affected by the **backgrnd** command. (See the **ignore** and **notice** commands). Finally, any grouping specification will be overridden by the grouping in the source spectral file so that the source and background are binned in the same way.

The format of the background file must match that of the spectrum file: for this purpose OGIP Type I and II are considered to be the same format.

For details of how to remove spectra see the **data** command documentation.

Examples:

Suppose there are currently three spectra. Then:

```
XSPEC12> backgrnd a,b,c
// New files for background subtraction are given for all
// three spectra.
XSPEC12> backgrnd 2 none
// No background subtraction will be done for the second spectrum.
XSPEC12> backgrnd ,d
// d.pha becomes the background for the second spectrum.
XSPEC12> backgrnd 2 e{4-5}
// Rows 4 and 5 of Type II file e.pha become the background for
// the second and third spectrum respectively.
```

5.4.3 corfile

change the correction file for a given spectrum

Reset the files used for background correction.

Syntax: **corfile** [<filespec>...]

where <filespec> is the same as for the **backgrnd** command. The correction file can be associated with a spectrum to further adjust the count rates. It is a PHA file whose count rate is multiplied by the current associated correction norm (see the **cornorm** command and **recorn** model) and then subtracted from the input uncorrected data. The correction norm is not changed by running the **corfile** command. Default values for the correction file and norm are included in the data PHA file. Unlike the background file, the correction data does NOT contribute to the measurement error. A file name of `none` is equivalent to no correction file used. If an input file can not be opened or found, an error message is printed and the user prompted for a replacement. As with the **backgrnd** command, the correction file is checked against the associated spectrum for number of channels, grouping status, and detector ID. The current ignore status for channels is not affected by the **corfile** command. Note that correction files have the same format as the PHA files used by the **data** command.

Examples:

It is assumed that there are currently three spectra:

```
XSPEC12> corfile a,b,c
// New correction files are used for all three spectra.>
XSPEC12> corfile 2 none
// No correction will be done for the second spectrum.}
XSPEC12> corfile ,d
// The 2nd file now uses d.pha as its correction.
XSPEC12> corfile 2 e{4-5}
// Rows 4 and 5 of Type II file e.pha becom the correction files for the second
// and third spectrum respectively.
```

5.4.4 cornorm

change the normalization of the correction file

Reset the normalization used in correcting the background.

Syntax: **cornorm** [[<spectrum range>...] [<cornorm>]]...

where <spectrum range> =:: <first spectrum no.> - <last spectrum no.> is a range of spectra to which the correction is to be applied and <cornorm> is the value to be used for the normalization. A decimal point (.) is used to distinguish a correction norm from a single spectrum <spectrum range>. If no correction norm is given, then the last value input is used (the initial value is one (1)). If no range is given, then the last single range input is modified. (See the **corfile** command.)

Examples:

Assume that there are four spectra, all with associated correction files already defined, either by default in their PHA file, or explicitly by using the **corfile** command.

```
XSPEC12> cornorm 1-4 1.
//The correction norm for all four is set to 1.0
XSPEC12> cornorm 0. 1-2 0.3
//The correction norm for the last input range (which was 1-4)
// is set to 0., then files 1 and 2 are reset to 0.3.
XSPEC12> cornorm 4
//file 4 has the correction also set to 0.3.
XSPEC12> cornorm 1 4 -.3
//files 1 and 4 are set to -.3.
XSPEC12> cornorm .7
//file 4 (as the last input single range) is set to 0.7.
```

5.4.5 data

read data, background, and responses

Input one or more spectra, together with their associated (background, response) files.

Syntax:	**data**	`<file spec1> [,...] [<file spec2>...] [/]`
	data	`none`
	data	`<spectrum #> none`

where the file specification is

`<filespec>` =:: [[`<data group #>`:] `<spectrum #>`] `<filename>` [{`ranges`}]

If a particular file is not found or cannot be opened for input for some reason, then the user is prompted for a replacement file name. An <EOF> at this point is equivalent to typing `none`. The default extension for all files is `.pha`, so all other extensions, (e.g. `.fak`) must be entered explicitly. The default directory is the current user directory when XSPEC is invoked. When a new file is input, by default all its PHA channels are considered good channels for fitting and plotting purposes (see the **ignore** and **notice** commands).

XSPEC's "native" data format is the OGIP standard. The standard specifies the representation of spectrum and all related datasets. XSPEC12 is explicitly designed to be able to work with other data formats as required: for example, the Integral/SPI spectral data format, although based on OGIP TypeII, deviates slightly. This was necessary because 3 response/arf pairs are required per spectrum. XSPEC12 has the ability to specify how response and other data are stored on disk, composed, and combined within the spectral fitting problem by adding new data modules at run-time. In XSPEC12, unlike XSPEC11, the channels that are ignored are a property of the spectrum, and therefore must be reset when the spectrum is replaced by another.

If the file contains multiple spectra, such as an OGIP Type II PHA file, then the desired spectrum can be specified by appending {`ranges`} to the end of the filename, where n is the row number of the spectrum in the file.

XSPEC12 allows any combination of multiple ranges in the parentheses delimited by commas. The wildcard characters * and ** may also be used. A ** on either side of a hyphen indicates either the first or last row in the file, based on whether it is to the left or right of the hyphen. (If a * is entered on the left or right side of a hyphen, it is substituted by the most recently entered left or right value respectively.) All rows in the file may be selected simply with a single * or ** between the brackets with no hyphen.

Examples:

```
XSPEC12> data pha2data{1,3,5-8,14-26,75-**}
//  In addition to the various specified rows between 1 and 26,
//  also load rows 75 through the end of the file.
XSPEC12> data pha2data{*}
//  Select all rows in the file.
```

For files with multiple spectra the data may either specify a header keyword specifying the response, auxiliary response, background and correction files, or these may be string-valued columns specifying a different filename per row.

Consult the http://heasarc.gsfc.nasa.gov/docs/software/ftools package documentation for details of how to modify the file.

The individual spectral data files are created outside of XSPEC by detector-specific software. They are organized as XSPEC data files, but often referred to as PHA files. Whatever its format, the PHA file contains such information as integration time, detector effective area, and a scaling factor (BACKSCAL in the OGIP standard) that estimates the expected size of the internal background. The data file also contains the names of the default files to be used for background subtraction and for the detector sensitivity versus incident photon energy (the response and arf files). A data file has the total observed counts for a number of channels and a factor for the size of any systematic error. Each channel is converted to a count rate per unit area (assumed cm^{-2}). The default background file is used for background subtraction. An error term is calculated using Poisson statistics and any systematic error indicated in the file.[1]

spectrum numbering

Multiple <`filespec`> clauses can be input on a single data command, or also on multiple data commands. Within XSPEC each set of data is referred to by its associated spectrum number,<`spectrum #`>, as determined by the following rules. For convenience, we denote the number of spectra that have been previously read in by data command as N_s.

Spectra in XSPEC are numbered sequentially from 1.

If no spectrum number is specified by the user, the spectrum in the first filename specified is assigned to 1. If spectra have already been loaded at this point, they will be replaced, deleted, or added to depending on the command. For example, if there are 3 spectra loaded ($N_s = 3$) and the user types:

[1] For OGIP files, any FITS NULL values will be converted to the value 1.E-32. This should have no practical effect because any channels with NULL values will presumably be marked as bad or otherwise ignored.

```
XSPEC12> data   multidatafile{1-2}
```

then spectra 1 and 2 will be replaced and 3 deleted. The command

```
XSPEC12> data   multidatafile{1-4}
```

will replace all three spectra and add the fourth.

If the user specifies a "load point", i.e. the first spectrum number to be created by the new command, i.e.

```
XSPEC12> data 3 multidatafile{1-4}
```

then that load point may not exceed $N_s + 1$. If it does, XSPEC will correct the number and issue a warning.

A skipped-over argument can be effected by a comma, for example

```
XSPEC12> data  3 spectrum1, , spectrum2
```

indicates that the spectrum for that position, as input in an earlier invocation of data, will continue to be used (in this example, spectrum 3 is replaced, 4 is left untouched, and 5 is either replaced or added. Any spectra with numbers greater than 5 are removed.

If the filename input is `none`, that spectrum is removed, and so are any higher-number spectra unless `none` is terminated with a / character. For example:

```
XSPEC12> data  3 none
```

removes all spectra numbered 3 or higher,

```
XSPEC12>data 3 none/
```

removes only spectrum 3 and renumbers the rest.

The data command determines the current total number N_T of spectra: either N_T spectra are implied by the command line, or the highest spectrum number

added (after XSPEC has made corrections as mentioned above) is N_T This is true UNLESS a / character terminates the data command.

If the line is terminated by a slash (/), then the current number of spectra is the previous total number of datsets N_s or the number as determined from the command line, whichever is greater.

The command

```
XSPEC12> data
```

by itself prints the one-line help summary, as does

```
XSPEC12> data ?
```

data groups

XSPEC allows the user to specify separate data groups for different spectra. Each data group has its own set of parameters for the defined model. These parameters can be either independent from data group to data group, or they can be linked across data groups using the standard XSPEC syntax (see the **newpar** command). This facility can be used for, say, analyzing extended sources.

Note that the data group number precedes the spectrum number: in the example

```
XSPEC12> data  2:3 spectrum4
```

which assumes that at least two spectra are already present, the data group number is 2 and the spectrum number is 3.

XSPEC will not allow the data group number to exceed the spectrum number: for example

```
XSPEC12> data  3:2 spectrum4
```

is invalid. XSPEC will correct this and issue a warning.

More Examples:

```
XSPEC12> data a
```

```
//The file a.pha is read in as the first (and only) spectrum.
XSPEC12> data ,b
//b.pha becomes the second spectrum, the first spectrum is
// unmodified (i.e. it is still a.pha)
XSPEC12> data c 3 d,e,f
//c.pha replaces a.pha as the first spectrum;d.pha, e.pha, and
// f.pha provide the, third, fourth, and fifth spectra.
XSPEC12> data g/
//g.pha replaces c.pha as the first spectrum; the slash (/)
// indicates that the 2nd through the 5th spectra remain as before.
XSPEC12> data 2 none/
//the string none indicates that the 2nd spectrum (b.pha) is to be
// totally removed. The current total number of datasets thus becomes
// one less (4).The current spectra are g.pha,d.pha, e.pha,
// and f.pha.
XSPEC12> data h,,
//The current total number of spectra becomes 2, the current data
// sets are from h.pha and d.pha.
XSPEC12> data
//There is no change in the data status.
XSPEC12> data 1
//The number of spectra is set explicitly to one, that being from
// h.pha.
XSPEC12> data 1:1 a 2:2 b 3:3 c
//Read a.PHA into data group 1, b.pha into data group 2, and c.pha
//into data group 3
XSPEC12> data 1:1 a 1:2 b 2:3 c
//Read a.pha and b.pha into data group 1, and c.pha into data group 2
XSPEC12> data a{3}
// Read the third spectrum in the file a.pha.
```

5.4.6 diagrsp

set a 'perfect' response for a spectrum

Diagonalize the current response matrix for ideal response.

Syntax: diagrsp

This command diagonalizes the current response matrix. The response matrix is set so that the channel values are mapped directly into the corresponding energy ranges, based on the channel energies and energy response range of the current response matrix.

This command is very similar to running **dummyrsp** in mode 1, with two important differences. Unlike **dummyrsp**, usage of this command requires that an actual response is currently loaded. It takes its energy range and channel binning information from this currently loaded response rather than user input parameters. Secondly, this does not change the effeciency (ie. effective area) as a function of energy stored for the current detector. Invoking this command will simulate a detector with perfect spectral resolution. If you wish to simulate a detector with perfect resolution AND perfect efficiency, use the **dummyrsp** command.

The previous response matrices can be reimplemented with the **response** command, with no arguments. Any use of the **data** and **notice** commands will replace the dummy diagonal response with the correct set of matrices.

5.4.7 fakeit

simulate observations of theoretical models

Produce spectra with simulated data.

Syntax: **fakeit** `[nowrite] [<file spec>...]`

where `<file spec>` =:: [`<file number>`] `<file name>`[{`ranges`}]... is similar to the syntax used in the **backgrnd**, **corfile**, and **response** command. The **fakeit** command is used to create a number of spectrum files, where the current model is multiplied by the response curves and then added to a realization of any background. Statistical fluctuations can be included. The integration time and correction norm are requested for each file. The file names input as command line arguments are used as background. The number of faked spectra produced is the maximum of the number of spectra currently loaded and the number of file specifications in the command line arguments. The special case **fakeit none** makes one fake spectrum for each spectrum loaded (or one fake spectrum if there are none loaded). See the examples below for a clearer description.

If **fakeit** is immediately followed by the `nowrite` specifier, no actual output files will be generated. In this case the fake spectra will exist just for the duration of the Xspec session (or until they are unloaded).

If a faked spectrum is based on a currently loaded spectrum, then by default the background, response, correction file, and numerical information are taken from the currently-defined data, unless a background file is specified on the command line in which case it becomes the background. The **fakeit none** case prompts for the rmf and arf filenames and sets the default numerical data to 1.0, except

the correction norm, which is set to zero. If the output file is type II then the exposure time and correction scale factor will be the same for all spectra in the file.

For each output file, the user will be prompted for an output file name. If a background file is in use then **fakeit** will also simulate a new background for each spectrum. Background files are given the same names as output spectrum files but with `_bkg` appended to the end of the stem.

The simulated spectra automatically become the current data files. The ignore status is completely reset.

Statistical Issues:

The statistical fluctuations used to create the simulated spectra will depend on whether the current spectra have Poisson or Gaussian errors. If a spectrum file has a STAT_ERR column and the POISSERR keyword is set to false then xspec assumes Gaussian errors with sigma from the values in the column. Otherwise, errors are assumed to be Poisson based on the number of counts. Note that it is possible for the spectrum and background files to have different error types. For fakeit cases when there is no current file to use, Poisson errors are assumed.

Type I vs. Type II Output:

Fakeit determines whether to place its fake spectra and background data into type I or type II files based on the following rules.

If fake spectra are based on currently loaded spectra then the output files will have the same format as those loaded. For example: Assume 3 spectra are currently loaded, spectrum 1 from file `typeIdata.pha` and spectra 2 and 3 from file `typeIIdata.pha`. Then,

```
XSPEC12> fakeit
```

will produce 3 fake spectra in 2 output files with names prompted from the user. The first file will be type I, the second type II containing 2 spectra. The same is true for any background files produced.

If the user asks for more fake spectra to be created than the number of spectra currently loaded, for example by typing the following when the same 3 spectra above described are loaded:

```
XSPEC12> fakeit 5
```

then fake spectra 1-3 will be placed in the two files as before. For the additional fake spectra (4 and 5), fakeit uses the following rule: If any of the originally loaded spectra were in a type II file, then all of the additional fake spectra will be placed in 1 type II file. Otherwise, they will each be placed in a separate type I file. In this example, since a type II file was originally loaded (`typeIIdata.pha`) when fakeit was called, spectra 4 and 5 will be placed together in a type II output file, in addition to the type I and type II files for the first 3 fake spectra.

If there are no currently loaded spectra all output files will be type I unless either of the following situations exist: 1. Any of the background files entered on the command line are type II, as indicated by row specifiers in brackets. 2. The first response file used clearly belongs to a format associated with type II data, such as SPI/Integral with its multiple RMF format (see section on SPI/Integral usage).

Overall, though the method of determining output format for additional spectra may seem quite complicated, it can be easily summed up: Fakeit will place all additional spectra and backgrounds (ie. those not based on already loaded data) in type I output files, unless it detects any evidence of type II file usage amongst the command line input, in which case it will produce type II output.

Note on grouped spectra:

If an input spectrum has grouping information (ie a GROUPING column telling XSPEC how to bin up the data) then fakeit will simulate the number of counts in each of the grouped bins. However, the spectrum that is written out must have the ungrouped number of channels (and a copy of the GROUPING column from the original spectrum). The solution that XSPEC adopts is to place all the counts from a grouped bin in the first channel which goes to make up that bin. This is of no consequence for future uses of the simulated spectrum provided that the GROUPING column is not changed. So, in this case grppha or similar tools cannot be run on the simulated spectrum.

Note For SPI/Integral Format:

Since the SPI/Integral format builds its responses from a combination of multiple RMFs and ARFs, it must use a different scheme than the OGIP type I and II formats for storing RMF and ARF file location information. This information is stored in a FITS extension, named "RESPFILE_DB" ,added to the PHA file. Therefore, when fakeit prompts the user for the location of the response file, simply enter the name of a FITS file which contains a RESPFILE_DB extension pointing to the RMFs and ARFs to be applied. When prompted for an ARF name, enter nothing.

The prompts will only appear for the first spectrum in the data set, and the ARFs will be assigned row by row 1 to 1 with the spectra. For example, if no

data is currently loaded, to create 3 fake SPI spectra from the RMFs and ARFs named in the RESPFILE_DB extension of the file `realSpiData.pha`:

```
XSPEC12> fakeit 3
// ...(various prompts will follow)...
For fake spectrum #1 response file is needed:  realSpiData.pha
// ...and ancillary file:  <Ret>
// ...(more fakeit prompts)...
```

This will create 3 fake spectra, each making use of the same RMFs/ARFs, spectrum 1 using the first row of the ARFs, spectrum 2 using the second etc.

*** CAUTION - SPI/Integral ***

As currently implemented, the RESPFILE_DB method of storing ARF locations does not retain specific row information. The assumption is that the rows in the ARF correspond 1 to 1 with the rows in the spectral data extension. Therefore, much confusion can arise when the row numbers of the loaded spectra do not match that of the fake spectra. For example:

```
XSPEC12> data my_spi_data.pha{3-4}
// my_spi_data.pha contains a RESPFILE_DB table pointing to arf1.fits,
// arf2.fit, arf3.fits.
// ...(fit to some model(s))...
XSPEC12> fakeit
```

This will produce 2 fake spectra generated from the model*response operation, where the model has parameters based on a fit to the original spectra in rows 3 and 4 of `my_spi_data.pha`, which used ROWS 3 AND 4 of the 3 arf files for their own responses. However, the responses used above to generate the 2 fake spectra will use ROWS 1 AND 2 of the 3 arf files. This is necessary since the fake spectra will be placed in rows 1 and 2 of their fakeit output file.

Examples:

Type I files:

For each of these examples, assume 3 spectra are currently loaded, each in its own type I file, and that the second spectrum has a background file.

```
XSPEC12> fakeit
```

This will produce 3 fake spectra each in its own type I output file, and the user will be prompted for the file names. The response file information will come from each of the original spectra. If any response information is invalid, the user will then be prompted. A fake background file will be produced for the second spectrum.

```
XSPEC12> fakeit 4
```

Produces 4 fake spectra, the first 3 created as in the previous example. The fourth will be created with no background spectrum, and this user is prompted for response information.

```
XSPEC12> fakeit backa,,none 4
```

Produces 4 fake spectra. For the first spectrum, a fake background file will be generated from the file `backa`. The second uses its own background file as before. The third fake spectra will no longer use the response information from loaded spectrum 3, the user will be prompted instead, and its default numerical data will be reset to 1. The fourth spectrum will be created as in the previous example.

If no data is currently loaded:

```
XSPEC12> fakeit 2
```

Produces 2 fake spectra in separate type I files, unless the first user entered response file belongs to a format that is explicitly type II (ie. SPI/Integral).

Type II files:

Assume four spectra with no backgrounds have been loaded from one type II file:

```
XSPEC12> data original_type2_data.pha{5-8}
```

Then, after model(s) have been entered and a fit:

```
XSPEC12> fakeit
```

This will produce 4 fake spectra in rows 1 to 4 of one type II output file, with responses and arfs taken from the columns of `original_type2_data.pha`.

```
XSPEC12> fakeit ,,backb{1-3}
```

This produces 5 fake spectra in two type II output files, and 3 fake background spectra also placed in two type II output files:

The first 4 fake spectra are placed in one output file since that is how the 4 spectra they were based on were originally organized. The default numerical data for this file are taken from the original spectra. Fake spectra 3 and 4 now have backgrounds, based on `backb{1}` and `backb{2}` respectively. These will generate 2 fake background spectra, placed in rows 3 and 4 of the first output fake background file. Rows 1 and 2 of this file will just consist of zeros since the first 2 spectra have no backgrounds.

The fifth fake spectrum will be placed in the second type II PHA file. Response and numerical data will not be based on the existing loaded spectra. A fake background will be generated from `backb{3}` and placed in row 1 of the second type II fake background file.

Now assume no data is currently loaded:

```
XSPEC12> fakeit 2 backb{1}
```

2 fake spectra in one type II output file are produced, as is a corresponding fake background file with 2 rows. The fact that the user has entered a type II background file on the command line tells fakeit to produce type II output. The first fake spectrum will have no associated background, so row 1 in the fake background file will be all zeros. Row 2 will consist of the fake background generated from `backb{1}`.

5.4.8 ignore

ignore detector channels

Ignore data channels. (See also **notice**.)

Syntax: **ignore** `<range1> [<range2>] ... [<rangeN>]`
ignore `bad`

where `<rangeI>` =:: `<spectrum range>:<channel range>` | `<channel range>`.

If no `<spectrum range>` is given, then the previous range is used (the initial default range is file one (1) only). The form of `<spectrum range>` is

<spectrum range> =:: <initial spectrum>-<last spectrum> | <spectrum>

where <init spectrum>, <last spectrum>, and <spectrum> are spectrum numbers, in the order that they were input with the **data** command. The form of <channel range> is

<channel range> =:: <initial channel>-<last channel> | <channel>

If integers are given for the channel ranges then channels will be used while if reals are given then energies (or wavelengths if **setplot wave** has been specified). Energy and wavelength units are determined by the **setplot energy** and **setplot wave** settings. If only the last channel is indicated, then a default value of one (1) is used for the initial channel. Channels remain ignored until they are explicitly noticed with the **notice** command, or if a spectrum is replaced.

Examples:

Assume that 4 spectra have been read in, the first 2 with 100 channels and the last 2 with 50 channels.

```
XSPEC12> ignore **:1-10
//The first 10 channels of all 4 spectra are ignored
XSPEC12> ignore 80-**
//An attempt will be made to ignore channels  in all four data
// sets (as that was the last spectrum range specified). As a result,
// only channels 80-100 will be ignored for spectra 1 and 2.
// No change will occur for spectra 3 and 4, as they have no
// channels greater than 50.
XSPEC12> ign 4:1-20 3:30-40 45-**
//Channels 11-20 for spectrum 4 are ignored (1-10 were ignored already)
// while channels 30-40 and 45-50 of spectrum 3 are ignored.
XSPEC12> ignore 1:1-5
//No channels are ignored, as these were ignored at the beginning.
XSPEC12> ignore 2:1.-5.
//Ignore all channels between 1 and 5 keV in the second dataset
```

5.4.9 notice

notice data channels

(See also **ignore**.)

Syntax: **notice** `<range1> [<range2>] ... [<rangeN>]`
notice `all`

where `<rangeI>` =:: `<spectrum range>:<channel range>` | `<channel range>`.

If no `<spectrum range>` is given, then the previous range is used (the initial default range is file one (1) only). The form of `<spectrum range>` is

`<spectrum range>` =:: `<initial spectrum>-<last spectrum>` | `<spectrum>`

where `<init spectrum>`, `<last spectrum>`, and `<spectrum>` are spectrum numbers, in the order that they were input with the **data** command. The form of `<channel range>` is

`<channel range>` =:: `<initial channel>-<last channel>` | `<channel>`

If `<channel range>` are integers then channels will be used or if reals then energies (or wavelengths if **setplot wave** has been specified). Energy and wavelength units are determined by the **setplot energy** and **setplot wave** settings. If only the last channel is indicated, then a default value of 1 is used for the initial channel. Channels remain noticed until they are explicitly ignored with the **ignore** command. When a spectrum is replaced by another spectrum, all input channels automatically are noticed.

```
XSPEC12> notice all
```

resets all the channels to 'noticed'.

Examples:

Assume that 4 spectra have been read in, the first 2 having 100 channels and the last 2 having 50 channels. Assume also that channels 1-10 of all four spectra are ignored and that channels 80-100 of spectra 1 and 2 are ignored.

In XSPEC12, **notice** does not force the detector response to be reread (see **response** description).

```
XSPEC12> notice **:1-10
//The first 10 channels of all 4 spectra are noticed.
XSPEC12> notice 80-**
//an attempt will be made to notice channels  in all 4 spectra
// (as that was the last spectrum range specified) but the result is that
// only channels 80-100 will be noticed for spectra 1 and 2, with no
// change for spectra 3 and 4 as  they have no channels greater than 50.
```

```
XSPEC12> notice 1:1-5
//No channels are noticed, as these channels were noticed
//in the beginning.
```

5.4.10 response

change the detector response for a spectrum

Modify one or more of the matrices used to describe the response(s) of the associated spectrum to incident X-rays.

Syntax: **response** [<filespec>...]
response [<source num>:]<spectrum num> none

where <filespec> =:: [[<source num>:]<spectrum num>] <file name>..., and <file name> is the name of the response file to be used for the response of the associated spectrum. If <file name> ends in a {n} specifier then the nth response will be read from the file. <spectrum num> is the spectrum number for the first file name in the specification, and follows similar rules as described in the **data** command description. An important difference however is that the **response** command may only be used to modify the response of a previously loaded spectrum: an error message is printed if the <spectrum num> is greater than the current number of spectra (as determined from the last use of the **data** command).

An optional <source num> may be specified to attach additional responses to a spectrum, and should be paired with <spectrum num> separated by a ':'. This allows the user to assign multiple models, each with their own response file, to a particular spectrum. See the **model** command for more information. If no <source num> is specified, it always defaults to 1. Source numbers do not need to be assigned consecutively to a spectrum, and gaps in numbering are allowed. The additional response may be removed with a **response** <source num>:<spectrum num> none command. Both the **show data** and **show response** commands will display current information regarding the response(s) to spectrum assignments.

A file name none indicates that no response is to be used for that spectrum. This situation means that any incident spectrum will produce no counts for those particular channels. If a file is not found or cannot be opened for input, then the user is prompted for a replacement response file. An <EOF> at this point is equivalent to using none as the response. See the **data** command for ways to totally remove the spectrum from consideration. The user is also prompted for a replacement if the response file has a different number of PHA channels than the associated spectrum. A warning will be printed out if the

response detector ID is different from the associated spectrum's. The current ignore status for channels is not affected by the command. (See the **ignore** and **notice** commands).

Examples:

It is assumed that there are currently three spectra:

Single source usage:

```
XSPEC12> response a,b,c
// New files for the response are given for all three files.
XSPEC12> response 2 none
// No response will be used for the second file.
XSPEC12> response ,d{2}
// The second response in d becomes the response for
//the second file.
```

Multiple source usage:

```
XSPEC12> response 2:1 e
// A second source with response e.rsp is now added to
// the first spectrum.  A second model can be assigned
// to this source.
XSPEC12> response 2:2 f  3:2 g
// A second and third source is assigned to spectrum 2.
XSPEC12> response 2:2 none
// The second source is now removed from spectrum 2.
```

5.5 Fit Commands

5.5.1 bayes

set up for Bayesian inference

Syntax:	**bayes**	`<option>`
	bayes	`<mod par #>` `<prior type>` `[<hyperparameters>]`

where `<option>` =:: `[off |on |cons]`. If a parameter number is given as the first argument then this command sets up the prior for the specified model

parameter but does not turn Bayesian inference on. If the first argument to the **bayes** command is not a parameter number then one of the options `off`, `on`, or `cons` is used. The first two turn Bayesian inference off or on, while `cons` turns Bayesian inference on and gives all parameters a constant prior. The options for prior types are as follows:

Prior type	Log(prior)
cons	0
exp	$-par/hpar1 - log(hpar1)$
jeffreys	$-log(par)$
gauss	$-0.5log(2\pi hpar2) - 0.5(hpar1 - par)^2/hpar2^2$

Where *par* is the parameter value and *hpar#* the hyperparameter values. jeff is an abbreviation for the Jeffreys prior, which is 1/x for an assumed Gaussian distribution of the parameter.

5.5.2 chain

run a Monte Carlo Markov Chain

Syntax: chain `[burn <length>] [clear] [filetype fits|ascii] [info] [length <length>] [load <filename>] [proposal [<distr> <source>] |[<user-defined>]] [rand on|off] [run [>]<filename>] [stat <par num>] [temperature <value>] [type mh|gw] [unload <range>] [walkers <value>]`

If the proposal source is set to use the fit correlation matrix (the default), you must perform a fit before running any chains.

When chains are loaded (and their parameters correspond to the currently loaded model), they will be used by the various XSPEC commands that require distributions of parameter values, such as **eqwidth** or **flux** when calculating error estimations. The **error** command itself will also use the loaded chains, determining the error range from a central percentage of the sorted chain values. This is likely to be faster than the **error** command's standard algorithm when not using chains.

`burn <length>` Specifies that the first `<length>` steps should be thrown away prior to storing the chain.

`clear` Does a reset and removes all chains from the list.

`filetype fits|ascii` Chooses the format of the output chain file. `fits` (the default) writes the chain to a binary table in a FITS file. `ascii` writes the chain to a simple text file. Either format is readable when using the **chain load** command.

`info` Prints out information on the current chains.

`length <length>` Sets the length for new chains.

`load <filename>` Loads a chain which has been run earlier, stored in file given by `<filename>`.

`proposal <distr> <source>` Selects the proposal distribution and source of covariance information to be used when running new chains. The default is `proposal gaussian fit`. Currently implemented `<distr>` options are: `gaussian` and `cauchy`. `<source>` options are:

`chain` Covariance is taken from the currently loaded chains.

`diagonal <values>` The values of a diagonal covariance matrix are entered directly on the command line, separated by commas and/or spaces: `C_11 C_22 ... C_nn`.

`<filename>` Covariance is read in from a user-specified text file. The file must contain the values of an NxN matrix where N is the current number of freely varying parameters. The values of each matrix row should be entered on one line with whitespace separation. Since this matrix is always symmetrical, values above the diagonal may be omitted. For example a 2x2 matrix could be entered as:

```
0.98
0.15 0.96
```

`fit` Covariance is taken from the correlation information produced by the current fit.

`matrix <values>` The lower half and diagonal of a symmetrical square covariance matrix are entered directly on the command line, separated by commas and/or spaces: `C_11 C_21 C_22 C_31 C_32 C_33 ... C_nn`

chain proposal with no other arguments will show a list of all available proposal options. For an alternative to XSPEC's `<distr> <source>` proposal options, the user may instead want to provide their own custom randomization algorithm. This can be done by writing their own C++ class(es) derived from an XSPEC randomizer base class. The custom class is added at runtime using the same **initpackage** or **lmod** command sequence as for local models, and is specified by `proposal <name>` where `<name>` is the unique name attribute the user provides for their class. Please see Appendix G for more information on writing a custom randomizing class, and **initpackage** for building and loading it.

`rand on|off` Specifies whether the chain start point will be randomized, or taken from the current parameters.

`recalc` A deprecated option that performs the equivalent of `proposal gaussian chain`.

`rescale <range>` Rescale the covariance matrix used in the proposal distribution by the factor given.

`run [>] <filename>` Runs a new chain written to the specified file, or append to an already loaded file if the '>' character preceeds the filename. The chain is written to the file as it runs so its performance can be monitored by examining the file. For high-chatter settings, additional information is printed to the screen. A long run may be interrupted with Ctrl-C, in which case the chain file will still exist but will not be automatically loaded. If appending to a file, the current `filetype` setting must match the format of the file or XSPEC will prevent it.

`stat [<modName>:]<parIdx>` Writes out statistical information on a particular parameter of the chain, specified by the parameter index number (with optional model name). The information displayed is:

line1: The mean of the parameter over each chain file.

line2: The parameter mean over all chain files and the variance between chain means.

line3: The variance within the chains.

line4: The Rubin-Gelman convergence criterion.

line5: The fraction of repeats, defined as the number of lines in the chain file for which all parameter values are identical to the previous line, divided by the number of lines in the file.

`temperature <value>` Sets the temperature parameter used in the Metropolis-Hastings algorithm for the proposal acceptance or rejection. The default value is 1.0 and zero or negative values are forbidden. By using the run append option, it is possible for different sections of the chain file to use different temperatures. The temperatures and the line numbers to which they apply are stored in the header of the FITS format chain files, or in the metadata section at the top of the ASCII text format files.

`type mh|gw` Determines the algorithm used to generate the chain. Choices are `mh` (Metropolis-Hastings) or `gw` (Goodman-Weare, the default). If using Goodman-Weare, must also set the `walkers` parameter. Also note for Goodman-Weare that the calculation of `walkers` sets may be sped up by

adjusting the **parallel** command's `walkers` option to perform the calculations over multiple processes. The walkers are initialized in two different ways depending on whether a fit has previously been run. If a fit has been run and a covariance matrix is available then walkers are initialized by drawing from a multi-Normal distribution whose variance matrix is based on the covariance matrix. If a covariance matrix is not available then parameters in each walker are initialized by drawing from a uniform distribution centered on the current fit parameters with width ten times that of the parameter delta.

`unload` <`range`> Removes the chains specified by <`range`> from the list in xspec. Note that this does NOT delete the chain files.

`walkers` <`value`> Sets the `walkers` parameter for the Goodman-Weare chain algorithm (see **chain type**). This must be an even integer, and both the chain length and burn length should be divisible by it (XSPEC will adjust the lengths to make them so if necessary).

All loaded chains must contain the same fit parameters. xspec will prevent the loading of a chain with a different number of parameters from the currently loaded chains.

Examples:

```
XSPEC12>chain length 100
//Sets length of chains produced by the run command to 100.
XSPEC12>chain run chain_file1.out
//Runs a chain based on current valid fit parameters, output to
//chain_file1.out
XSPEC12>chain run >chain_file1.out
//Appends another run of length 100 to the end of chain_file1.out
XSPEC12>chain load chain_old.out
//Loads a pre-existing chain file, the result of an earlier run
//command.  Warning is issued if not the same length as
//chain_file1.out
XSPEC12>chain stat 3
//Prints statistical information on the 3rd parameter of the chain.
XSPEC12>chain proposal gaussian myfile.txt
//New chain proposals will be a normal distribution using
//covariance values stored in myfile.txt rather than fit
//correlation matrix.
XSPEC12>chain prop gauss diag .1 .001 .0001
// New chain proposals will be a normal distribution using a 3x3
// diagonal covariance matrix with the values from the
// command line.
```

```
XSPEC12>chain temperature .8
// Sets the Metropolis-Hastings temperature value to .8 for
// future chain runs, replacing the default 1.0.
XSPEC12>chain clear
//Removes the 2 loaded chains from xspec's chain list.
```

5.5.3 error (and rerror)

determine confidence intervals of a fit

Determine the confidence region for a model parameter.

Syntax: error [[stopat <ntrial> <toler>] [maximum <redchi>] [nonew] [<delta fit statistic>] [<model param range>...]]

where <model param range> =:: [<modelName>:]<first param> - <last param>

determines the ranges of parameters to be examined, and <delta fit statistic> (distinguished from the model parameter indices by the inclusion of a decimal point), is the change in fit statistic used.

For response parameters (see **gain** command), use **rerror** with identical syntax except:

<response param range> =:: [<sourceNum>:]<first param> - <last param>

The **error** command uses one of two algorithms. If Monte Carlo Markov Chains are loaded (see **chain** command) the error range is determined by sorting the chain values, and then taking a central percentage of the values corresponding to the confidence level as indicated by <delta fit statistic>. This is likely to be the faster of the two algorithms.

When chains are **not** loaded, **error**'s algorithm is as follows:

Each indicated parameter is varied, within its allowed hard limits, until the value of the fit statistic, minimized by allowing all the other non-frozen parameters to vary, is equal to the last value of fit statistic determined by the **fit** command plus the indicated <delta fit statistic>, to within an absolute (not fractional) tolerance of <toler>. Note that before the **error** command is executed, the data must be fitted. The initial default values are the range 1-1 and the <delta fit statistic> of 2.706, equivalent to the 90% confidence region for a single interesting parameter. The number of trials and the tolerance for determining when the critical fit statistic is reached can be modified by preceeding them

with the `stopat` keyword. Initially, the values are 20 trials with a tolerance of 0.01 in fit statistic.

If a new minimum is found in the course of finding the error, the default behavior is to abort the calculation and then automatically rerun it using the new best fit parameters. If you prefer not to automatically rerun the **error** calculation, then enter `nonew` at the start of the command string. The maximum keyword ensures that **error** will not be run if the reduced chi-squared of the best fit exceeds <`redchi`>. The default value for <`redchi`> is 2.0.

Since there are very many scenarios which may cause an **error** calculation to fail, it is highly recommended that you check the results by viewing the 9-letter error string, which is part of the output from the **tclout error** command (see **tclout** for a description of the error string). If everything went well, the error string should be "FFFFFFFFF".

Examples:

```
//Assume that the current model has four model parameters.

XSPEC12> error 1-4
//Estimate the 90% confidence ranges for each parameter.
XSPEC12> error 9.0
//Estimate the confidence range for parameters 1-4 with delta fit
// statistic = 9.0, equivalent to the 3 sigma range.
XSPEC12> error 2.706 1 3 1. 2
//Estimate the 90% ranges for parameters 1 and 3, and the 1. sigma
// range for parameter 2.
XSPEC12> error 4
//Estimate the 1. sigma range for parameter 4.
XSPEC12> error nonew 4
//Same as before, but calculation will NOT automatically restart if a new minimum is found.
XSPEC12> error stop 20,,3
//Estimate the 1-sigma range for parameter 3 after resetting the number
// of trials to 20.Note that the tolerance field had to be included
//(or at least skipped over).
```

5.5.4 fit

fit data

Find the best fit model parameters for the current data by minimizing the

current statistic.

Syntax: **fit** <fit method parameters>

The arguments to fit depend on the fitting method currently in use. See the **method** command for details (and for the usage of the USE_NUMERICAL_DIFFERENTIATION option in the user's startup Xspec.init file). Output from the **fit** command also depends on the fitting method currently in use.

Using the Levenberg-Marquardt algorithm, the parameters accepted are the maximum <number of iterations> before the user is prompted, the <critical delta>, which is the (absolute, not fractional) change in the statistic between iterations less than which the fit is deemed to have converged, and <critical beta>.

The <critical beta> provides an optional second stopping criterion, and it refers to the |beta|/N value reported during a Levenberg-Marquardt fit. This is the norm of the vector derivatives of the statistic with respect to the parameters divided by the number of parameters. At the best fit this should be zero, and so provides another measure of how well the fit is converging. <critical beta> is set to a negative value by default, which renders it inactive.

Including the string `delay` as an argument to fit turns on delayed gratification. It is turned off by `nodelay`. Delayed gratification modifies the way the damping parameter is set and has been shown in many cases to speed up convergence. The default is `nodelay`.

If <number of iterations>, <critical delta>, <critical beta>, `delay`, or `nodelay` is entered through the **fit** command, it also becomes the future default value for the currently loaded fit **method** (ie. Levenberg-Marquardt).

Examples:

```
XSPEC12> fit
// Fit with the default number of iterations and critical delta
// chi-squared.
XSPEC12> fit 60
// Fit with 60 as the number of iterations.
XSPEC12> fit  50 1.e-3
// Fit with 1.e-3 as the critical delta.
XSPEC12> fit 50 1.e-3  20.
// Same fit, but will now use |beta|/N = 20.0 as another stopping criterion in additio
// to that of the critical delta.
XSPEC12> fit delay
// Same fit, but will now use delayed gratification.
```

5.5.5 freeze (and rfreeze)

set parameters as fixed

Do not allow indicated model parameters to vary. (See also **thaw**)

Syntax: **freeze** [<param range>...]

where <param range> =:: [<modelName>:]<param#>|<param#> - <param#>.

For response parameters (see **gain** command):

Syntax: **rfreeze** [<param range>...]

where <param range> =:: [<source number>:]<param#>|<param#> - <param#>.

The indicated model parameter or range of model parameters will be marked so they cannot be varied by the **fit** command. By default,the range will be the last range input by either a **freeze** or **thaw** command.

Examples:

Currently there are six parameters, initially all unfrozen.

```
XSPEC12> freeze 2
//Parameter 2 is frozen
XSPEC12> freeze 4-6
//Parameters 4, 5, and 6 are frozen.
XSPEC12> thaw 2 3-5
//Parameters 2, 4, and 5 are thawed, parameter 3 is unaffected.
XSPEC12> freeze
//Parameters 3,4,5 are frozen (the last range input by a freeze
//or thaw command).
XSPEC12> rfreeze 4-6
//Response parameters 4, 5, and 6 are frozen.
```

5.5.6 ftest

calculate the F-statistic from two chi-square values

Calculate the F-statistic and its probability given new and old values of $\chi^2$and number of degrees of freedom (DOF).

Syntax: **ftest** `chisq2 dof2 chisq1 dof1`

The new $\chi^2$and DOF, chisq2 and dof2, should come from adding an extra model component to (or thawing a frozen parameter of) the model which gave chisq1 and dof1. If the F-test probability is low then it is reasonable to add the extra model component. WARNING : it is not correct to use the F-test statistic to test for the presence of a line (see Protassov et al 2002, ApJ 571, 545). WARNING: this command can only be used if the extra model component is additive, this does not give the correct result if the component is multiplicative (see Orlandini et al. 2012, ApJ 748, 86).

5.5.7 goodness

perform a goodness of fit Monte-Carlo simulation

Perform a Monte Carlo calculation of the goodness-of-fit.

Syntax: **goodness** `[<# of realizations>] [sim|nosim]`

This command simulates `<# of realizations>` spectra based on the model and writes out the percentage of these simulations with the fit statistic less than that for the data. If the observed spectrum was produced by the model then this number should be around 50%. This command only works if the sole source of variance in the data is counting statistics. The `sim|nosim` switch determines whether each simulation will use parameter values drawn from a Gaussian distribution centered on the best fit with sigma from the covariance matrix. The `sim` switch turns on this option, `nosim` turns it off in which case all simulations are drawn from the best-fit model. The default starting setting is `nosim`. The goodness calculation may be sped up by setting the **parallel** command's `goodness` option to run multiple processes.

5.5.8 margin

MCMC probability distribution

Use the currently loaded MCMC chains to calculate a multi-dimensional probability distribution.

Syntax: **margin** `<step spec.> [<step spec.> ...]`

where `<step spec.>` ::= `[log|nolog] [<model name>:]<fit param index> <low value> <high value> <no. steps>`.

The indicated fit parameter is stepped from <`low value`> to <`high value`> in <`no. steps`>+1 trials. The stepping is either linear or log. Initially, the stepping is linear but this can be changed by the optional string `log` before the fit parameter index. `nolog` will force the stepping to be returned to the linear form. The number of steps is set initially to ten. The results of the most recently run margin command may be examined with **plot margin** (for 1-D and 2-D distributions only). This command does not require that spectral data files are loaded, or that a valid fit must exist.

Examples:

Assuming chain(s) are loaded consisting of 4 parameters.

```
XSPEC12>margin 1 10.0 12.0 20 log 3 1.0 10.0 5
//Calculate a 2-D probability distribution of parameter 1 from 10.0-12.0 in 20 linear bins, and p
XSPEC12>margin 2 10.0 100.0 10 nolog 4 20. 30. 10
//Now calculate for parameter 2 in 10 log bins and parameter 4 in 10 linear bins.
```

5.5.9 renorm

renormalize model to minimize statistic with current parameters

Renormalize model, or change renorm conditions.

Syntax: **renorm** `[auto|none|prefit]`

The **renorm** command will adjust the normalizations of the model by a single multiplication factor, which is chosen to minimize the fit statistic. Such a renorm will be performed explicitly whenever the command is used without a key-word, or during certain XSPEC commands, as determined by the following key-words:

`auto`	Renormalize after a **model** or **newpar** command, and at the beginning of a fit.
`prefit`	Renormalize only at the beginning of a fit.
`none`	Perform no automatic renormalizations, i.e., only perform them when a **renorm** command is given explicitly.

This command works by doing an analytic calculation to change the normalizations of all additive components in the model to minimize the fit statistic. The relative normalizations of the components will not be changed. This analytic calculation cannot be performed for all the fit statistics, in particular it will not

work for cstat if there is a background file. If the analytic calculation is not possible this command will have no effect.

5.5.10 steppar

generate the statistic "surface" for 1 or more parameters

Perform a fit while stepping the value of a parameter through a given range. Useful for determining confidence ranges in situations where greater control is needed than given with the **error** command.

Syntax: **steppar** `[current|best] <step spec> [<step spec>...]`

where `<step spec>` ::= `[log|nolog] [<modelName>:]<param index> <low value> <high value> <# steps>`

or `<step spec>` ::= `[log|nolog] [<modelName>:]<param index> delta <step size> <# steps>`

In the first case the parameter is stepped from `<low value>` to `<high value>` in `<# steps>` plus one trials. In the second case the parameter is stepped from `<best fit value>`-`<step size>`*`<# steps>` to `<best fit value>>` +`<step size>`*`<# steps>`, ie a total of 2`<# steps>`+1 trials. The stepping is either linear or log. Initially, the stepping is linear but it can be changed by the optional string `log` before the parameter index. `nolog` will force the stepping to be returned to the linear form. If more than one parameter is entered, then `<# steps>` must be entered for each one except the last. Note that every variable parameter whose `<param index>` is NOT entered in the command will still be allowed to vary freely during each steppar iteration.

To perform a **steppar** run on **gain** (or response) parameters, the optional `[<modelName>:]` specifier is replaced by an optional `[<sourceNumber>:]` specifier, and the letter 'r' needs to be attached as a prefix to the `<parameter index>`. For example:

```
steppar 2:r3 1.5 2. 10
```

will step the third response parameter belonging to source number 2.

The number of steps is set initially to 10. At each value, the parameter is frozen, a fit performed, and the resulting value of chi-squared given. If `best` is given

as an argument then the non-stepped parameters are reset to the best-fit values at each grid point. Alternatively, if `current` is given as an argument then the non-stepped parameters are started at their values after the last grid point (the default).

If multiple <`step spec`> are given for different parameters, then a raster scan of the parameter ranges is performed. At the end of the set, the parameters and chi-squared are restored to the values they had initially.

If the model is in a best-fit state when a **steppar** run is started and a new best fit is found during the run, the user will be prompted at the end of the run to determine if they wish to accept the new best-fit values for their parameters. This prompting can be disabled by the setting of the **query** flag.

Depending on the machine, a **steppar** run may be sped up significantly by assigning it to multiple processes. See the **parallel** command with the **steppar** option for more details.

Examples:

Assume that the current model has four parameters:

```
XSPEC12> steppar 3 1.5 2.5
//Step parameter 3 from 1.5 to 2.5 in steps of .1.
XSPEC12> steppar log
//Repeat the above, only use multiplicative steps of 1.0524.
XSPEC12> step nolog 2 -.2 .2 20
//Step parameter 2 linearly from -.2 to .2 in steps of 0.02.
XSPEC12> step 2 delta 0.02 5
      //Step parameter 2 linearly from the best-fit value-0.1 to
      //the best-fit value+0.1 in a total of 11 steps.
```

5.5.11 thaw (and rthaw)

allow fixed parameters to vary

Allow indicated parameters to vary. (See also **freeze**)

Syntax: **thaw** [<`param range`>...]

where <`param range`> =:: [<`modelName`>:]<`param#`>|<`param#`> - <`param#`>.

For response parameters (see **gain** command):

Syntax: **rthaw** [<param range>...]

where <param range> =:: [<source number>:]<param#>|<param#> - <param#>.

The indicated parameter, or range of parameters, will be marked as variable by the fitting commands and treated as a fitting parameter in subsequent fits. By default, the range will be the last range input by either a **freeze** or **thaw** command. See the **freeze** examples for an example of the use of the **thaw** command.

5.5.12 weight

change weighting used in computing statistic

Change the weighting function used in the calculation of chi-squared.

Syntax: **weight** [standard | gehrels | churazov | model]

standard weighting uses $\sqrt{N}$ or the statistical error given in the input spectrum. gehrels weighting uses $1+\sqrt{N+0.75}$, a better approximation when N is small (Gehrels, N. 1986, ApJ 303, 336). churazov weighting uses the suggestion of Churazov et al. (1996, ApJ 471, 673) to estimate the weight for a given channel by averaging the counts in surrounding channels. model weighting uses the value of the model, not the data, to estimate the weight.

5.6 Model Commands

5.6.1 addcomp

add component to a model

Add a component to the model.

Syntax: **addcomp** [<modeName>:]<n> <comp>

where <n> is the component number *before* which the new component is to be inserted, and <comp> is the name of the new component. Components are numbered in sequence in order of appearance in the expression entered. The new component is regarded as an operator on the component added if it is not

additive.

The optional <`modelName`> qualifier allows the user to address a named model.

The user is prompted for parameter values for the component. If there are *m* components in the current model, then acceptable values for the component number added are *1* to *m+1*.

XSPEC detects the type of the model (additive, multiplicative etc), checks the correctness of the syntax of the output model, and adds the component if the resulting models obeys the syntax rules documented in the **model** command.

Thus,

```
XSPEC12> mo  wa(po)
```

Followed by

```
XSPEC12> addcomp 2 bb
```

Yields the model achieved by

```
XSPEC12> mo  wa(bb + po)
```

See also **delcomp** (delete component by number).

Other examples will serve to clarify **addcomp**'s behavior. Suppose that the current model specification is

```
ga+po
```

which using the **show** command would yield the description

```
model  = gaussian[1] + powerlaw[2]
```

The comments give the model expression following the entry of **addcomp** and **delcomp** commands:

```
XSPEC12> addcomp 2 wab
```

```
//gaussian[1]+wabs[2](powerlaw[3])
XSPEC12> addcomp 4 pha
//(gaussian[1]+wabs[2](powerlaw[3]))phabs[4]}
XSPEC12> delcomp 1
//(wabs[1](powerlaw[2]))phabs[3]}
XSPEC12> addcomp 2 zg
//(wabs[1](zgauss[2]+powerlaw[3]))phabs[4]}
XSPEC12> delcomp 3
//(wabs[1](zgauss[2]))phabs[3]

XSPEC12> mo wa(po)
XSPEC12> addcomp 1 ga
// gauss[1] + wabs[2]*powerlaw[3]
XSPEC12> delcomp 1
XSPEC12> addcomp 1 pha
// phabs[1]*wabs[2]*powerlaw[3]

XSPEC12>mo wabs(po)
XSPEC12> addcomp 3 bb
// wabs[1]*powerlaw[2] + bbody[3]
XSPEC12> delcomp 1
XSPEC12> addcomp 3 pha
// wabs[1]*powerlaw[2]*pha[3]
XSPEC12> addcomp 3 po
// ERROR: po (additive) is interpreted as being added to the multiplicative
// model pha[3], which is a context error.

For multiply nested models?
XSPEC12> mo wa(po + pha(bb + ga))
XSPEC12> addcomp 6 po
// wabs[1](powerlaw[2] + phabs[3](bbody[4] + ga[5]) + powerlaw[6])
XSPEC12> addcomp 5 peg
// wabs[1](powerlaw[2] + phabs[3](bbody[4] + pegpwlw[5] ga[6]) + powerlaw[7])
XSPEC12> addcomp 7 wa
// wabs[1](powerlaw[2] + phabs[3](bbody[4] + pegpwlw[5] ga[6]) + wabs[7]*powerlaw[8])
```

5.6.2 addline

add spectral lines to a model

Tcl script to add one or more lines to the current model in an optimum fashion.

Syntax: **addline** [<nlines>] [<modeltype>] [fit|nofit]

<nlines> additional lines are added one at a time. Line energies are set to that of the largest residual between the data and the model. For each line a fit is performed with the line width and normalization as the only free parameters. The default option is one gaussian line. The other <modeltype> that can be used is lorentz. If no third argument is given then the sigma and normalization of each line are fit. If `nofit` is specified then the fit is not performed but if `fit` is specified then all free parameters are fit.

addline currently will only work with the default model (i.e. not for named models).

5.6.3 delcomp

delete a model component

Delete one or more components from the current model.

Syntax: **delcomp** [<modelName>:]<comp num range>

where <comp num range> is a range of positions in the model specification of the components to be deleted.

Examples:

Suppose that the current model specification is

```
wa(po+ga+ga)
```

Then

```
XSPEC12> delcomp 3-4
//Changes the model to wa(po)
XSPEC12> delcomp 1
//Changes the model to po
```

5.6.4 dummyrsp

create and assign dummy response

Create a "dummy" response, covering a given energy range.

Syntax: dummyrsp[<low energy> [<high energy> [<# of ranges> [log|linear [<channel offset> [<channel width> [<sourceNun>:<specNum>]]]]]]]

This command creates a dummy response matrix based on the given command line arguments, which will either temporarily supersede the current response matrix, or create a response matrix if one is not currently present. There are two main uses for this command: to do a "quick and dirty" analysis of uncalibrated data (mode 1), and to examine the behaviour of the current model outside the range of the data's energy response (mode 2). Note that mode 2 usage has now been rendered redundant by the more flexible **energies** command.

All parameters are optional. The initial default values for the arguments are 0.01 keV, 100 keV, 200 logarithmic energy steps, 0.0 channel offset, and 0.0 channel width. The default values of the first 5 parameters will be modified each time the parameter is explicitly entered. The channel width parameter however always defaults to 0.0 which indicates mode 2 operation, described below.

In addition to the 6 optional parameters allowed for versions 11.x and earlier, a seventh optional parameter has been added allowing the user to apply the dummy response to just one particular source of a spectrum. It consists of two integers for (1-based) source number and spectrum number, separated by a colon. Either both integers should be entered, or they should be left out entirely. ie. A dummy response is either made for EVERY source in every spectrum, or just 1 source in 1 spectrum. This parameter always defaults to all sources and all spectra.

For mode 1 usage, simply enter a non-zero value for the channel width. In this instance, one has a spectrum for which typically no response matrix is currently available. This command will create a diagonal response matrix with perfect efficiency, allowing for the differences in binning between the photon energies and the detector channel energies (see example below). The response matrix will range in energy from <low energy> to <high energy>, using <# of ranges> as the number of steps into which the range is logarithmically or linearly divided. The detector channels are assigned to have widths of energy <channel width> (specified in keV), the lower bound of the first channel starting at an energy of <channel offset>. Then the data can be fit to models, etc., under conditions that assume a perfect detector response.

For mode 2 usage (channel width = 0.0), one can use this command to examine the current model outside the range of the energy response of the detector. When examining several aspects of the current model, such as plotting it or determining flux, XSPEC uses the current evaluation array. This, in turn, is defined by the current response files being used, which depend on the various detectors. For example, low energy datasets (such as those from the EXOSAT LEs) may have responses covering 0.05 to 2 keV, while non-imaging proportional

counters can span the range from 1 to 30 keV. If the user wishes to examine the behavior of the model outside of the current range, then he or she temporarily must create a dummy response file that will cause the model to be evaluated from <`low energy`> to <`high energy`>, using <`# of ranges`> as the number of steps into which the range is logarithmically or linearly divided. If one wishes only to set the energy response range, than the <`channel width`> argument may be omitted. In this case, or in the case where no data file has been read in, all entries of the dummy response matrix are set to zero. Under these circumstances the dummyrsp has no physically correct way of mapping the model into the data PHA channels, so the user should not try to fit-or plot-the data while the dummyrsp is active in this mode. Also, data need not even be loaded when calling this command in mode 2.

The previous response matrices can be reimplemented with the **response** command, with no arguments. Any use of the **data** and **notice** commands will replace the dummy response with a correct set of matrices, or with no response matrix if none was originally present.

Examples:

```
XSPEC12> dummyrsp
//Create the dummy response for all spectra and sources with the
//default limits, initially .01, 100, and 200 bins.
XSPEC12> dummyrsp .001 1
//Create a dummy response with 200 bins that cover the range from
//0.001 to 1 keV.
XSPEC12> dummyrsp ,,,500
//The same range, but now with 500 bins.
XSPEC12> dummyrsp ,,,,lin
//The same range, but now with linearly spaced bins.
XSPEC12> dummyrsp ,,,,,0.1
//The same range, but now create a diagonal response matrix, with
//channel widths of 0.1 keV.
XSPEC12> response
//Restore any previous correct responses.
```

Example dummy response matrix:

Assume a spectrum with 4 channels, then

```
XSPEC12> dummyrsp .0 30.0 3 lin 5.0 8.0
```

will produce the following response:

	Detector channel energies			
Energies	5.0-13.0	13.0-21.0	21.0-29.0	29.0-37.0
0.0-10.0	0.5	0	0	0
10.0-20.0	0.3	0.7	0	0
20.0-30.0	0	0.1	0.8	0.1

5.6.5 editmod

edit a model component

Add, delete, or replace one component in the current model.

Syntax: editmod [<delimiter>] <component1> <delimiter> <component2> <delimiter> ... <componentN> [<delimiter>]

where <delimiter> is some combination of (,+,*,and), and <componentJ> is one of the models known to XSPEC.

The arguments for this command should specify a new model, with the same syntax as the previous model, except for one component which may be either added, deleted, or changed to a different component type. XSPEC then compares the entered model with the current model, determines which component is to be modified (prompting the user if necessary to resolve ambiguities) and then modifies the model, prompting the user for any new parameter values which may be needed.

Examples:

```
XSPEC12> mo wabs(po)
XSPEC12> ed wabs(po+ga)
//This command will add the component gauss to model
// in the specified place and prompt the user for its initial
// parameters.
XSPEC12> mo wabs(po+zg)
XSPEC12> ed po+zg
//This command will delete the component wabs from the
//model, leaving the other components and their current
//parameter values unchanged
XSPEC12> mo wabs(po+po)
XSPEC12> ed wabs(po)
```

```
//Here an ambiguity exists as to which component to delete.
//In this case XSPEC will print out  the current model,
//showing the component number for each component, and then
//prompt the user for which component he wants deleted.
XSPEC12> mo wabs(po+ga)
XSPEC12> ed wabs(po+zg)
//The component gauss will be replaced by the component zgauss,
//and the user will be prompted for parameter values for the new
// component
```

5.6.6 energies

specify new energy binning for model fluxes

Supply an energy-binning array to be used in model evaluations in place of their associated response energies. The calculated model spectra are then interpolated onto the response energy arrays before multiplying by the response matrix. This command replaces and enhances the **extend** command from earlier versions.

Syntax:
energies `<range specifier> [<additional range specifiers>...]`
energies `<input ascii file>`
energies `extend <extension specifier>`
energies `reset`

where the first `<range specifier>` ::= `<low E> <high E> <nBins> log|lin`
`<additional range specifiers>` ::= `<high E> <nBins> log|lin`
`<extension specifier>` ::= `low|high <energy> <nBins> log|lin`

All energies are in keV. Multiple ranges may be specified to allow for varied binning in different segments of the array, but note that no gaps are allowed in the overall array. Therefore only the first range specifier accepts a `<low E>` parameter. Additional ranges will automatically begin at the `<high E>` value of the previous range.

The `extend` option provides the same behavior as the old **extend** command. Models will use associated response energy arrays, with an additional low and/or high array extension. `<energy>` is the value to which the array is extended, using `<nBins>` additional log or linear bins.

With the `<input ascii file>` option, the user can instead supply a customized energy array from a text file. The format requirements are simply that the bin values must appear 1 to a line and in ascending sorted order. Blank lines are allowed and so are comments, which must be preceded by a '#'. A

simple example:

```
# myEnergyBinning.txt
.1
1.0
10.    # now some linear bins
15.
20.
25.
```

which would actually produce the same energy array as:

```
energies .1 10. 2 log  25. 3 lin
```

Once an energy array is specified, it will apply to all models, and will be used in place of any response energy array (from actual or dummy responses) for calculating and binning the model flux. It will also apply to any models that are created after it is specified. To turn off this behavior and return all models back to using their response energies, simply type **energies reset**.

Similarly, an array extension created by the `extend` option will continue to be applied to all models until it is either overwritten by another extension, replaced by a new energies array, or removed with the `reset` option. This allows both low and high extensions to exist together.

When a custom-energy binned model flux array needs to be multiplied by a response matrix, xspec will temporarily rebin the flux array to match up with the response energy binning. This is done by simply scaling the flux by the fractional overlap between the custom and response bins. If there is no overlap between the custom and response energies, then the response will be multiplied by zero.

The **energies** command saves the most recently entered range and extension specifiers to be used as default values the next time it is called. The initial default range specifier is 1 range with <`low E`> = .1, <`high E`> = 10., <`nBins`> = 1000, and `lin`. The initial default extension specifier is `high` with <`energy`> = 100., <`nBins`> = 200, and `log`.

Examples:

```
XSPEC12> energies ,50,,log
// Creates an array from .1 to 50. of 1000 logarithmic bins.
```

```
XSPEC12> energies ,,,,100. 5 lin
// Modifies previous array by adding 5 linear bins from 50. to 100.
XSPEC12> energies ,,,,200.
// The 2nd range is now 50. to 200. in 5 linear bins.
XSPEC12> energies 1.,,100
// Array is now just 1 range, 1. to 50. in 100 logarithmic bins.
XSPEC12> energies myFile.txt
// Array is replaced with values stored in myFile.txt
XSPEC12> energies extend ,75.,,lin
// Models will go back to using response energies, but with an
// extension of the high end to 75. keV in 100 additional linear bins.
XSPEC12> energies extend low .01
// Add a low-end extension to .01 keV with 100 new linear bins.
XSPEC12> energies reset
// All models will go back to using the original energy arrays
// from responses.
```

5.6.7 eqwidth

determine equivalent width

Determine the equivalent width of a model component.

Syntax: eqwidth `[[range <frac range>] [<model name>:]<model component number>] [err <number> <level>|noerr]`

The command calculates the integrated photon flux produced by an additive model component (combined with its multiplicative and/or convolution prefactors) (FLUX), the location of the peak of the photon spectrum (E), and the flux (photons per keV) at that energy of the continuum (CONTIN). The equivalent width is then defined as {EW = FLUX / CONTIN} in units of keV. New for XSPEC12: the continuum is defined to be the contribution from all other components of the model.

There are certain models with a lot of structure where, were they the continuum, it might be inappropriate to estimate the continuum flux at a single energy. The continuum model is integrated (from $E(1- < fracrange >)$ to $E(1+ < fracrange >)$. The initial value of `<frac range>` is 0.05 and it can changed using the `range` keyword.

The `err/noerr` switch sets whether errors will be estimated on the equivalent width. The error algorithm is to draw parameter values from the distribution and calculate an equivalent width. `<number>` of sets of parameter values will be drawn. The resulting equivalent widths are ordered and the central `<level>`

percent selected to give the error range. You can get the full array of simulated equivalent width values by calling **tclout eqwidth** with the `errsims` option (see **tclout** command).

When Monte Carlo Markov Chains are loaded (see **chain** command), they will provide the distribution of parameter values for the error estimate. Otherwise the parameter values distribution is assumed to be a multivariate Gaussian centered on the best-fit parameters with sigmas from the covariance matrix. This is only an approximation in the case that fit statistic space is not quadratic.

Examples:

The current model is assumed to be $M_1(A_1 + A_2 + A_3 + A_4 + M_2(A_5))$, where the M_x models are multiplicative and the A_x models are additive.

```
XSPEC12> eqwidth 3
// Calculate the total flux of component M1A2 (the third
// component of the model with its multiplicative pre-factor)
// and find its peak energy (E). The continuum flux is
// found by the integral flux of M1(A1+A3+A4+M2(A5)), using the
// range of 0.95E to 1.05E to estimate the flux.
XSPEC12> eqwidth range .1 3
// As before, but now the continuum is estimated from
// its behavior over the range 0.9E to 1.1E.
XSPEC12> eqwidth range 0 3
// Now the continuum at the single energy range (E)
// will be used.
XSPEC12> eqwidth range .05 2
// Now the component M1A1 is used as the feature, and
// M1(A2+A3+A4+M2(A5)) are used for the continuum.  The range
// has been reset to the original value.
XSPEC12> eqwidth 1
// Illegal, as M1 is not an additive component.
```

5.6.8 flux

calculate fluxes

Calculate the flux of the current model between certain limits.

Syntax: flux `[<lowEnergy> [<hiEnergy>]] [err <number> <level>|noerr]`

where `<lowEnergy>` and `<hiEnergy>` are the values over which the flux is

calculated. Initial default values are 2 to 10 keV.

The flux is given in units of photons $cm^{-2}s^{-1}$ and ergs $cm^{-2}s^{-1}$. The energy range must be contained by the range covered by the current spectra (which determine the range over which the model is evaluated). Values outside this range will be reset automatically to the extremes. Note that the energy values are two separate arguments, and are NOT connected by a dash. (see parameter ranges in the **freeze** command).

The flux will be calculated for all loaded spectra. If no spectra are loaded (or none of the loaded spectra have a response), the model is evaluated over the energy range determined by its dummy response. (In XSPEC12, models are automatically assigned default dummy responses when there is no data, so the dummyrsp command need not be given.) If more than 1 model has been loaded, whichever model the user has specified to be the active one for a given source is the one used for the flux calculation.

The results of a **flux** command may be retrieved by the **tclout flux** <`n`> command where n is the particular spectrum of interest. If the flux was calculated for the case of no loaded spectra, the results can be retrieved by **tclout flux** with the <`n`> argument omitted.

The `err/noerr` switch sets whether errors will be estimated on the flux. The error algorithm is to draw parameter values from the distribution and calculate a flux. <`number`> of sets of parameter values will be drawn. The resulting fluxes are ordered and the central <`level`> percent selected to give the error range. You can get the full array of simulated flux values by calling **tclout flux** with the `errsims` option (see **tclout** command).

When Monte Carlo Markov Chains are loaded (see **chain** command), they will provide the distribution of parameter values for the error estimate. Otherwise the parameter values distribution is assumed to be a multivariate Gaussian centered on the best-fit parameters with sigmas from the covariance matrix. This is only an approximation in the case that fit statistic space is not quadratic.

There is also a model component **cflux** which can be used to estimate fluxes and errors for part of the model. For instance, defining the model as `wabs(pow + cflux(ga))` provides a fit parameter which gives the flux in the gaussian line.

Examples:

The current data have significant responses to data within 1.5 to 18 keV.

```
XSPEC12> flux
//Calculate the current model flux over the default range.
```

```
XSPEC12> flux 6.4 7.0
//Calculate the current flux over 6.4 to 7 keV
XSPEC12> flux 1 10
//The flux is calculated from 1.5 keV (the lower limit of the
//current response's sensitivity) to 10 keV.
```

5.6.9 gain

modify a response file gain

Modify a response file gain, in a particularly simple way. ***CAUTION*** This command is to be used with extreme care for investigation of the response properties. To properly fit data, the response matrix should be recalculated explicitly (outside of XSPEC) using any modified gain information derived.

The **gain** command shifts the energies on which the response matrix is defined and shifts the effective area curve to match. The effective area curve stored by XSPEC is either the ARF, if one was in use, or is calculated from the RSP file as the total area in each energy range. This means that if there are sharp features in the response then these will only be handled correctly by the gain command if they are in the ARF or if no ARF is input. The new energy is calculated by

E' = E/<slope> - <intercept>

where <intercept> is in units of keV.

Syntax:	**gain**	[<sourceNum>:]<specNum> <slope> <intercept>
	gain	fit [[<sourceNum>:]<specNum>]
	gain	nofit {[[<sourceNum>:]<specNum>]|all}
	gain	off

The first variant of the gain command shown above will apply the gain shift specified by the <slope> and <intercept> parameters to the response belonging to spectrum <specNum>, and optionally specified <sourceNum> if the data is analyzed with multiple models. The initial default <specNum> is 1; later, the default is the number of the spectrum last modified. Initially, all responses are assumed to have nominal gains, determined implicitly by the data in the response files. This is equivalent to a <slope> of 1 and an <intercept> of zero. All responses can be reset back to this original state by entering **gain off**. Note that in this mode of usage, the slope and intercept values do NOT become variable fit parameters. They are simply fixed values used to modify the response.

The **gain fit** mode is used when the user wishes to have the slope and intercept

parameters determined by the results of a fit. The <`specNum`> and optional <`sourceNum`> parameters specify to which response the fit gain values are to be applied. These may be omitted only if a single spectrum is loaded, with a single model source. Otherwise at least a spectrum number is required. The user will then be prompted for slope and intercept parameter information in the same way as model parameters are normally entered. These values are then immediately applied to the response, and will be adjusted the next time a fit is run.

Gain fit parameters belong to the more general category of **response parameters** in XSPEC, and may be modified using an equivalent set of commands to those used for regular model parameters. The command names are the same except prefixed by the letter **'r'**:

XSPEC commands for editing/viewing model parameters	Equivalent commands for gain (or response) parameters)
newpar	rnewpar
freeze	rfreeze
thaw	rthaw
untie	runtie
error	rerror
model	rmodel
show par	show par, show rpar

For example after assigning gain fit parameters to source 1 of spectrum 1 (with `gain fit 1`):

```
XSPEC12> rfreeze 1
XSPEC12> rnewpar 2 .05
XSPEC12> show rpar

Response parameters defined:
========================================================================
Source No.: 1
Rpar Spectrum Rmodel   Rpar_name  Unit   Value

   1     1    gain     slope             1.00000      frozen
   2     1    gain     offset            5.00000E-02  +/-  0.0
------------------------------------------------------------------------
```

Rnewpar can also link gain parameters to one another and can adjust the hard and soft parameter limits, as newpar does for model parameters. The default gain parameter hard limits are hardcoded in XSPEC, but these can be overrid-

den by setting GSLOP_MIN, GSLOP_MAX, GOFFS_MIN, and GOFFS_MAX keywords in the matrix extension of your response file.

The gain operation itself belongs to the category of **response functions**, which in future versions of XSPEC may be defined with **rmodel** just as regular XSPEC model functions are defined with **model**. Though **gain** is currently the only available response function, the following command will work:

```
// Apply gain to the response belonging to source 2 of spectrum 1
XSPEC12>rmodel 2:1 gain
```

which is equivalent to:

```
XSPEC12>gain fit 2:1
```

The `nofit` argument switches off the fitting and leaves the gain at the current values of the parameters. Unless the argument `all` is given, it is applied to a single response specified by <`specNum`> and optional <`sourceNum`>. As with **gain fit**, both arguments may be omitted if only a single spectrum with 1 source is loaded. When `all` is specified, fitting is switched off for the gain parameters of all responses. **gain off** will switch off fitting for all gain parameters, and will reset all of them to their nominal value.

Whenever a new response file is defined for a spectrum, the response will return to the `nofit` state with nominal value. The **ignore** and **notice** commands however will not affect the current gain of the response. THE GAIN COMMAND IS NOT CURRENTLY IMPLEMENTED FOR DUMMY RESPONSES.

Examples:

```
XSPEC12>gain 1 0.98
// The response belonging to spectrum 1 is adjusted with a slope of 0.98.
// The 1 may be omitted if only 1 spectrum (with 1 source) is loaded.
XSPEC12>gain 1,,.03
// The offset also is moved now by 0.03 keV.
XSPEC12>gain 2:4 1.1 0.1
// The response belonging to source number 2, spectrum 4, is adjusted with slope 1.1
// and offset 0.1 keV.
XSPEC12>gain off
// The above 2 responses, and any others that have been adjusted, are reset to slope
// 1.0, offset 0.0.
XSPEC12> gain fit 3
```

```
// Variable fit parameters are created for spectrum 3 response.  User will be prompted
// for starting fit parameter values of slope and offset.
XSPEC12> fit
// Best fit gain values will now be determined for and applied to spectrum 3 response.
XSPEC12> gain nofit 3
// Spectrum 3 response will retain its current gain values, but values will not be
// adjusted during future fits.
```

NOTE: Current gain information may be easily viewed with the **show response** command. Gain fit parameters may also be viewed with the **show par** or **show rpar** commands.

Historical Notes:

The gain command has been slightly revised for XSPEC12. Previously when a user entered a gain command, it was generally interpreted to apply to an entire **model**. This new implementation clearly defines an applied gain as belonging to a particular **response**. It also offers less ambiguity for dealing with XSPEC12's multiple models scheme. So for example if 2 spectra are loaded, each in its own data group, and the user enters a **gain fit** command, under the old system they would be prompted for 2 sets of parameters since the model is applied to 2 data groups. With the new system, the user specifies which particular response (belonging to either spectrum 1 or 2) they wish to apply the gain fit to, and are then prompted for just the 1 set of gain parameters for that response. This is more clearly demonstrated with the examples above. The new command options **gain nofit all** and **gain off** are also described above.

***** NOTE: Backwards incompatible syntax change *****

Beginning with XSPEC 12.5.1, **gain** parameters must be specified as [<sourceNum>:]<specNum> and NOT <specNum>[:<sourceNum>]. This reversal was made so that the gain command conforms to the [<sourceNum>:]<specNum> usage in other XSPEC commands, such as **response** and **arf**.

5.6.10 identify

identify spectral lines

List possible lines in the specified energy range.

Syntax: **identify** `<energy> <delta_energy> <redshift> <line_list>`

The energy range searched is <energy>±Δ<energy> (keV) in the rest frame of the source. If working in wavelength mode, as set by the setplot command, then the <energy> and <delta energy> parameters should be entered as wavelengths (in Angstroms). <line list> specifies the list of lines to be searched. The options are bearden, which searches the Bearden compilation of fluorescence lines (Bearden, J.A., 1967, Rev.Mod.Phys. 39, 78), mekal, which uses the lines from the mekal model (q.v.) and apec, which uses the APEC http://cxc.harvard.edu/atomdb line list. The apec option takes an additional two arguments: the temperature of the plasma (keV) and a minimum emissivity of lines to be shown. If the command **xset** has been used to set APECROOT then **identify** uses the APECROOT value to define the new atomic physics data files. See the help on the **apec** model for details.

5.6.11 initpackage

initialize a package of local models

The **initpackage** command initializes a package of local models from their source code and from a model component description file in model.dat format which defines the component's name, type, function call, and its parameter names and initial settings. Further details of the file format, function and parameter specifications are given in Appendix C, Adding Local Models To XSPEC [Note: **initpackage** is now also supported on Cygwin. The former Cygwin-only static_initpackage command has been removed.]

Syntax: initpackage <name> <description file> [<directory>] [-udmget |-udmget64]

The <name> argument names the package. For internal reasons package names must be lowercase: the **initpackage** command will force lower case and warn the user if the argument contains uppercase letters. Also there should be no numerals in the package name.

The <description file> argument specifies the model component description file. The third argument <directory> is optional and specifies the location of the source code. If it is not given, the value of the setting LOCAL_MODEL_DIRECTORY given in the user's Xspec.init file will be used. Finally, the <description file>, if not specified as an absolute pathname, will be read from the same directory as the source code.

Another optional argument is -udmget, for local model libraries containing Fortran code which makes use of XSPEC's now-obsolete udmget function for dynamic memory allocation. None of the functions in XSPEC's built-in models library use udmget anymore, and the necessary xsudmget.cxx file no longer

resides there. If a user still requires this code for their own local models, they should add `-udmget` at the end of the comamnd line. **initpackage** will then copy the files `xsudmget.cxx` and `xspec.h` into the user's local model directory.

initpackage performs the following tasks:

- reads the model description file
- writes code that will load the new component calculation functions
- writes a makefile that will drive the compilation and installation of the new code
- invokes the compiler and builds the library.

A separate command, **lmod**, actually loads the library. This two step process makes it easier to determine where the user is during the process if compilation failures arise. Further, if the model is complete and working correctly, only the **lmod** command need be invoked.

initpackage can also be run as a stand-alone program outside of XSPEC. When used like this however, after **initpackage** has finished the user must manually run hmake to build their library. XSPEC performs this part automatically using a script file.

5.6.12 lmod

load a package of local models

The **lmod** command loads a user model package Further details are given in Appendix C. [Note: This command is now also supported on Cygwin.]

Syntax: **lmod** `<name> [<directory>]`

As with **initpackage**, the `<name>` argument is the name of the model package being loaded, and the `<directory>` is the its location, defaulting to the setting of LOCAL_MODEL_DIRECTORY given in the user's `Xspec.init` file.

lmod performs the following tasks:

- loads the library corresponding to the package named `<name>`
- reads the model description file supplied by the **initpackage** command for the library

- adds the new model components to the list of models recognized by the model command

Note that **lmod** requires that the user has write-access to <`directory`> (please see Appendix C for details).

5.6.13 lumin

calculate luminosities

Calculate the luminosity of the current model for a given redshift and rest frame energy range.

Syntax: lumin [<`lowEnergy`>] [<`hiEnergy`>] [<`redshift`>] [`err` <`number`> <`level`>|`noerr`]

where <`lowEnergy`> and <`hiEnergy`> are the rest frame energies over which the luminosity is calculated and <`redshift`> is the source redshift. Initial default values are 2 to 10 keV for 0 redshift. The luminosity is given in units of ergs/s. The energy range redshifted to the observed range must be contained by the range covered by the current spectra (which determine the range over which the model is evaluated). Values outside this range will be automatically reset to the extremes. Note that the energy values are two separate arguments and are NOT connected by a dash (see parameter ranges in the freeze command description).

The **lumin** will be calculated for all loaded spectra. If no spectra are loaded (or none of the loaded spectra have a response), the model is evaluated over the energy range determined by its dummy response. (In XSPEC12, models are automatically assigned default dummy responses when there is no data, so the **dummyrsp** command need not be given.) If more than 1 model has been loaded, whichever model the user has specified to be the active one for a given source is the one used for the **lumin** calculation.

The results of a **lumin** command may be retrieved by the **tclout lumin** <`n`> command where <`n`> is the particular spectrum of interest. If **lumin** was calculated for the case of no loaded spectra, the results can be retrieved by **tclout lumin** with the <`n`> argument omitted.

The `err/noerr` switch sets whether errors will be estimated on the luminosity. The error algorithm is to draw parameter values from the distribution and calculate a luminosity. <`number`> of sets of parameter values will be drawn. The resulting luminosities are ordered and the central <`level`> percent selected

to give the error range. You can get the full array of simulated lumin values by calling **tclout lumin** with the `errsims` option (see **tclout** command).

The parameter values distribution is assumed to be a multivariate Gaussian centered on the best-fit parameters with sigmas from the covariance matrix. This is only an approximation in the case that fit statistic space is not quadratic.

There is also a model component **clumin** which can be used to estimate luminosities and errors for part of the model. For instance, defining the model as `wabs(pow + clumin(ga))` provides a fit parameter which gives the luminosity in the gaussian line.

Examples:

The current data have significant response to data within 1 to 18 keV.

```
XSPEC> lumin,,,0.5
//Calculate the current model luminosity over the default range for z=0.5
XSPEC> lumin 6.4 7.0
//Calculate the current luminosity over 6.4 to 7 keV.
```

5.6.14 mdefine

define a simple model using an arithmetic expression

Syntax: mdefine`[<name> [<expression> [: [<type>] [<emin> <emax>]]]`

where `<name>` = the name of the model. If `<name>` is a previously defined model with **mdefine**, the current definition will overwrite the old one, and the user is warned; if it is a built-in model, however, the user will be asked to use a different name.

`<expression>` = a string of arithmetic expression. Simple rules for expression:

1. The energy term, must be 'e' or 'E' in the expression. Other words, which are not numerical constants nor internal functions, are assumed to be model parameters.
2. If a convolution model varies with the location on the spectrum to be convolved, the special variable '.e' or '.E' may be used to refer to the convolution point.
3. The expression may contain spaces for better readability.

<type> = user may optionally specify the type of the model, the valid types are `add, mul, con`. (Mix models are not yet implemented as of v12.5.0) Please note that the character ':' must be used to separate the options from the <expression>. If <type> is not given default is `add`.

<emin> <emax> = user may also specify the minimum and maximum energy values for the model, the default values are 1.e-20 and 1.e+20, respectively.

Note that **mdefine** can also be used to display and delete previously defined models:

1. To display the name, type and expression of all previously defined models:

   ```
   XSPEC12>mdefine
   ```

2. To display the name, type and expression of a previously defined model by the name, MNAME:

   ```
   XSPEC12> mdefine MNAME
   ```

3. To delete a previously defined model by the name, MNAME:

   ```
   XSPEC12> mdefine MNAME :
   ```

Operators:

The following operators are recognized in an expression:

+	=	plus operator
-	=	minus operator
*	=	multiplying operator
/	=	dividing operator
**	=	exponentiation operator
^	=	exponentiation operator

Functions:

The following internal functions are supported:

unary functions

EXP	(expr)	=	exp of a vector expression
SIN	(expr)	=	sine of vector expression in rad
SIND	(expr)	=	sine of a vector expression in degree
COS	(expr)	=	cosine of a vector expression in rad
COSD	(expr)	=	cosine of a vector expression in degree
TAN	(expr)	=	tangent of a vector expression in rad
TAND	(expr)	=	tangent of a vector expression in degree
LOG	(expr)	=	base 10 log of a vector expression
LN	(expr)	=	natural log of a vector expression
SQRT	(expr)	=	sqrt of a vector expression
ABS	(expr)	=	absolute value of a vector expression
INT	(expr)	=	integer part of a vector expression
ASIN	(expr)	=	sinˆ-1 of a vector expression in rad
ACOS	(expr)	=	cosˆ-1 of a vector expression in rad
MEAN	(expr)	=	mean value of a vector expression
DIM	(expr)	=	dimension of a vector expression
SMIN	(expr)	=	minimum value of a vector expression
SMAX	(expr)	=	maximum value of a vector expression

binary functions

MAX	(expr1, expr2)	=	maximum of the two vector expressions
MIN	(expr1, expr2)	=	minimum of the two vector expressions

Examples:

```
//  define a model named "dplaw" with 3 parameters, p1, p2, f
XSPEC12> mdef dplaw E**p1 + f*E**p2

// define a model named "junk" with 2 parameters (a, b)
XSPEC12> mdef junk a*e+b*log(e)/sin(e)

// define a model named "junk2" with 1 parameter, a; the option
//   following ":" says that it will be a multiplicative model.
XSPEC12> mdef junk2  exp(-a*e) : mul

// define a model named "junk3" with 1 parameter, B, options
//   following ":" says that this will be a multiplicative model
XSPEC12> mdef junk3  0.2+B*e : mul

// try to define a blackbody model with name "bb", you get warning:
XSPEC12> mdef bb E**2/T**4/(exp(E/T)-1)
```

```
        ***Warning: bb is a pre-defined model
        Please use a different name for your model.

// this defines a Gaussian convolution model with sigma varying with
//   square root of energy.
XSPEC12> mdef sg  exp(-E^2/(2*A*.E)) / sqrt(6.283*A*sqrt(.E))  :  con

// delete junk2
XSPEC12> mdef junk2 :

// display all user-defined models
XSPEC12> mdef

-- Name ---- Type ------ Expression -----
dplaw           add       E**p1+f*E**p2
junk            add       a*E+b*LOG(E)/SIN(E)
junk3           mul       a+b*E
sg              con       EXP(-E^2/(2*A*.E))/SQRT(6.283*A*SQRT(.E))
-----------------------------------------
```

5.6.15 model (and rmodel)

define a theoretical model

Define the form of the theoretical model to be fit to the data.

Syntax: model[<source num>:<name>] [<delimiter>] <component1> <delimiter> <component2> <delimiter>...<componentN> [<delimiter>]

model [?]
model [<name>|unnamed] none
model clear
model <name>|unnamed active|inactive
rmodel [<source num>:]<spec num> <response function>|none

where <delimiter> is some combination of (, +, ∗,), and <componentJ> is one of the model components known to XSPEC. The optional name must be preceded by a source number followed by a colon. To specifically refer to the default model use the string unnamed. Descriptions of these models may be accessed by typing help model <componentJ> at the prompt.

The source argument and name, if present, assign that model to be used with one of the sources found to be in the spectrum during the data pipelining. These 2 parameters allow one to simultaneously analyze multiple models, each assigned

to their own responses. The model will be referred to the channel space using a response corresponding to that source number. To create a model for a source number higher than 1, a detector response must first exist for that number. See the examples below and the **response** command for more information about using multiple sources. This ability to assign multiple models both generalizes and replaces the XSPEC11 method of using '/b' to specify background models.

After the model is loaded, if there are data present the model is attached through the instrumental response to the spectra to be fitted, as in XSPEC11. Unlike XSPEC11, however, if there are no data loaded the model will be attached to a default diagonal dummy response. The parameters of that dummy response (energy range, number of flux points, linear/logarithmic intervals) can be set by the user in the `Xspec.init` file using the DUMMY setting. Thus any model can be plotted in energy or wavelength space as soon as it has been defined.

Component Types

The model components are of various types depending on what they represent and how they combine with other models additive, multiplicative, convolution, pile-up, and mixing models. Each component may have one or more parameters that can be varied during the fit (see the **newpar** command writeup).

Additive model components are those directly associated with sources, such as power laws, thermal models, emission lines, etc. The net effect of two independent additive models is just the sum of their individual emissivities.

Multiplicative model components do not directly produce photons, but instead modify (by an energy-dependent multiplicative parameter) the spectrum produced by one or more additive components. Examples of multiplicative models are photoelectric absorption models, edges, absorption lines, etc.

Convolution models components modify the spectrum as a whole, acting like operators rather than simply applying bin by bin multiplication factors. An example of a convolution model is a gaussian smoothing with energy dependent width. Thus, when using convolution models, the ordering of components is in general significant (see below under **syntax rules**).

The **pile-up** model is similar to the operation of the convolution models. The only difference is that the flux is multiplied by the effective area on input and divided by the same factors on output.

Mixing model components implement two-dimensional transformations of model spectra. The data are divided into regions by assigning them to 2 or more datagroups, and the transformation "mixes" the flux among the regions. An example is the **projct** (projection) model, which assumes that the regions are 3-dimensional ellipsoidal shells in space, and projects the flux computed from

the other components onto 2-dimensional elliptical annuli.

A list of all the currently installed models is given in response to the command

```
XSPEC12> model ?
\\ The '?' is not actually required.
\\ This will leave the current model in use.
```

The new command variants have the following uses:

model [<name>] none
: removes the model of name <`name`> if given. Without the <`name`> argument, the command removes the unnamed "default" model, which is of course the XSPEC11 behavior.

model clear
: removes all models

model <name>|unnamed active|inactive
: makes the model named <`name`> active (fit to data) or inactive. Inactive models are tied to a dummy (unit diagonal) response. Making a model assigned to a given source active makes any previous model assigned to that source inactive. Note that to make the default unnamed model active or inactive refer to it by the string unnamed.

  See the commands **delcomp**, **addcomp** and **editmod** for details on how to modify the current model without having to enter a completely new model.

rmodel [<source num>:]<spec num> <response function>|none
: assigns or removes a response function to the response belonging to <`source num`> of spectrum <`spec num`>. Currently the only available <`response function`> in XSPEC is **gain**, which makes **rmodel** redundant with the **gain** command usage:

  `gain fit [<source num>:]<spec num>`

  The **rmodel none** option removes the response function and restores the response to its initial state.

Syntax Rules

Model components are combined in the obvious algebraic way, with + separating additive models, ∗ separating multiplicative models, and parentheses to show which additive models the multiplicative models act on. The ∗ need not be included next to parentheses, where it is redundant. Also, if only one additive

model is being modified by one or more multiplicative models, the required brackets may be replaced by a $*$. In this case the additive model must be the last component in the grouping. Thus

M1*(A1+A2) + M2*M3(A3) + M4*A4 + A5

is a valid model, where the M's signify multiplicative models and the A's additive models.

The old style syntax for entering models (versions 9.02 and earlier) is not supported in version 12 and will return a syntax error.

XSPEC12's recursive lexical analyzer and expression parser allows, in principle, infinite nesting depth. It has been tested to 3 levels of parentheses, although it should be said that this new behavior is a by-product of the design rather than fulfilling an important need. Thus, expressions such as

M1*(A1 + A2*(A3 + M2*M3*(A4 + A5))) + C1*(A6 + M4*(A7 + M5*A8))

are supported.

The model expression is analyzed on entry and syntax errors, or undefined models, will return control to the prompt with an error message. XSPEC12's model definition algorithm treats expressions delimited by '+' signs that are not within parentheses as separate "Component Groups". The Component Group comprises a list of components of the different types, and these are in turn calculated and then combined to produce an internal "Sum Component". These Sum Components from each such component group are then added to produce the output model (note that if there is an overall component - for example, a convolution or mixing component - then all of the model will be contained inside one Component Group).

The syntax rules that are checked for are as follows:

- A '*' must be preceded and followed by words or a brace (redundant braces are removed).
- A standalone component must be additive. A standalone component is defined as a single component model or a single component at the beginning (end) of the expression followed (preceded) by a '+', or in the middle of the expression delimited by 2 '+' signs.
- A convolution or mixing component must not appear at the end, or followed by a closing brace.

When using convolution or mixing components, the order in which they are applied is in general significant.For example, the two models

C1*M1(A1+A2) and M1* C1 (A1+A2)

are not necessarily equivalent (here the C's represent convolution models). The way XSPEC handles the ordering of components is by first computing the spectrum for the additive components of a given additive group (A1+A2 in the above example). It then applies all multiplicative or convolution components in the additive group from right to left in the order they appear in the model formula.

N.B. Beginning with v12.5.0, convolutions no longer have to precede the source. Parentheses may also be used to specify convolution precedence, so the following two examples are not equivalent:

C1*M1(A1+A2) and (C1*M1) (A1+A2)

Examples:

Note that po (= **powerlaw**) and ga (= **gauss**) are additive models, and that **wabs** and **phabs** (different photoelectric absorption screens) are multiplicative models.

```
XSPEC12> model po
// The single component po (powerlaw) is the model.
XSPEC12> model po+ga
XSPEC12> model (po+ga)wabs
XSPEC12> model phabs(po+ga)
XSPEC12> model wa(phabs(po)+ga)
XSPEC12> model wa po phabs ga //error: old syntax
XSPEC12> model wa*phabs*po
XSPEC12> model (po+po)phabs
//Note that though the first and second components are the same
// form, their parameters are varied separately.
XSPEC12> model phabs*wa(po)

A complex (and almost certainly unphysical) example is the following:

XSPEC12>model wa(po+pha(peg+edge(disk+bbod)))const + pla(pos+hr*step) + not*gau

Applying multiple models:
Assume 3 spectra are loaded, each with a single response (source 1 by default).
```

```
XSPEC12> model wa(po)
// The unnamed model wa(po) will apply to all 3 spectra, accordingly
// multiplied by each spectrum's response.
XSPEC12> response 2:2 new_resp.pha 2:3 another_new_resp.pha
// Additional responses assigned to source number 2 for spectra 2 and 3.
XSPEC12> model 2:second_mod ga
// The model "second_mod" will now apply to source 2, and is therefore
// multiplied by new_resp.pha and another_new_resp.pha for spectra 2
// and 3 respectively.
XSPEC12> model second_mod inactive
// "second_mod" will no longer apply to spectra 2 and 3, though they
// retain responses for source 2.
OR
XSPEC12> response 2:2 none
XSPEC12> response 2:3 none
// No responses exist for source number 2, second_mod is
// rendered inactive.
```

5.6.16 modid

write out possible IDs for lines in the model

Tcl script to write out possible IDs for gaussian or lorentzian lines in the current model.

Syntax: **modid** [<delta> |conf]

This script runs the **identify** command for every gaussian or lorentzian line included in the current model. If a number is given as an argument then that is used as the delta energy for identify. If the string `conf` is given as the argument then the last calculated confidence regions are searched for possible line IDs. If no argument is given then `conf` is assumed.

5.6.17 newpar (and rnewpar)

change parameter values

Adjust one or more of the model parameters.

Syntax: **newpar** [<modelName>:]<index range> [<param spec list>]
Syntax: **newpar** [<modelName>:]<index> = <coupling expression>
Syntax: **newpar** 0

where:

```
<param spec list> =:: <param value> <delta> <param range spec>
<param range spec> =:: <hard min> <soft min> <soft max> <hard max>
```

For **response parameters** (created with the **gain** or **rmodel** command):

```
Syntax:    rnewpar [<sourceNum>:]<idx range> [<param spec list>]
Syntax:    rnewpar [<sourceNum>:]<index> = <coupling expression>
```

The model parameters are accessed through their model parameter indices. For example, the first parameter of the first model component generally is model parameter 1, etc. The first command line argument, <index range>, gives the indices' parameters to be modified by the **newpar** command. The default value is the range from the previous invocation of **newpar**. The remaining arguments can be used to update the parameter specification. If the parameter specification is omitted from the command line, then the user is explicitly prompted for it. The first two arguments of the parameter specification are:

`<param value>`	The trial value of the parameter used initially in the fit
`<delta>`	The step size used in the numerical determination of the derivatives used during the fitting process. When delta is set to zero, the parameter is not adjustable during the fit. This value may be overriden for all parameters by the **xset delta** command option, which will apply a proportional rather than a fixed delta.

The four arguments of the range specification determine the range of acceptable values for the parameter. The soft limits should include the range of expected parameter behavior. Between the hard and soft limits, the parameter is made stiffer to adjustment by the minimization routine invoked by the **fit** command. The parameter is never allowed to have a value at or outside the hard limits.

A slash (/) will set all the six parameter specification values (value, delta, range specification) to the previous value (default for a new model, current value if the parameter has previously been set or fit).

The sequence /* leaves all parameters unchanged (in the case of a new model, to be set to the default).

newpar 0 prints the current parameter settings.

Parameter Links

Coupling of parameters allows parameters in a model to always have the same value or to be related by an expression. The expression is a function of the other parameters (XSPEC will reject attempts to link parameters to themselves!). Scale parameters (i.e. never variable during a fit), and s witch parameters (i.e. that change the mode in which a component is calculated) can only be linked to other scale and switch parameters, respectively. Details of parameter types are explained in more detail in Appendix C.

The syntax for linking parameters is

```
XSPEC12>newpar <par> = f(<par>)
```

where `f` is a function in the (other) parameters. Parameters can be specified either by the character 'p' followed by the parameter number (preferred) or by the parameter number. Integers appearing in `f` that are within the range of existing parameter numbers will be interpreted as parameters: to avoid confusion, if a real number is intended it should include a decimal point. Integers larger than the last parameter number will be interpreted as integers. Parameters of named models must be prefixed by `<modelName:>`.

The following operators and functions can be used in `f`:

operators

+	=	plus operator
-	=	minus operator
*	=	multiplying operator
/	=	dividing operator
**	=	exponentiation operator
^	=	exponentiation operator

unary functions

EXP	(expr)	=	exponential
SIN	(expr)	=	sine in radians
SIND	(expr)	=	sine in degrees
COS	(expr)	=	cosine in radians
COSD	(expr)	=	cosine in degrees
TAN	(expr)	=	tangent in radians
TAND	(expr)	=	tangent in degrees
LOG	(expr)	=	base 10 log
LN	(expr)	=	natural log
SQRT	(expr)	=	square root
ABS	(expr)	=	absolute value
INT	(expr)	=	integer part
ASIN	(expr)	=	inverse sine in radians
ACOS	(expr)	=	inverse cosine in radians
MEAN	(expr)	=	mean value

binary functions

MAX	(expr1, expr2)	=	maximum of the two expressions
MIN	(expr1, expr2)	=	minimum of the two expressions

If there are multiple data groups present, then the parameters of models associated with datagroups greater than 1 ("secondary models") are coupled by default to their "primary" counterparts. For example, if there are 5 parameters in the model and 3 datagroups present, then the model command will prompt for 15 parameters. If the user types

```
XSPEC12> model <expression>
XSPEC12>/*
```

Then parameters 1-5 will be set to their values specified in the initialization (`model.dat`) file. Parameters 6-15 will be linked to their counterparts, i.e. as if the user had typed

```
XSPEC12> newpar 6 = p1
XSPEC12> newpar 7 = p2
...
XSPEC12> newpar 11 = p1
```

And so on.

Examples:

The total number of model parameters for the example is four.

```
XSPEC12> newpar 2 0.1
// The value of the second parameter is set to 0.1.
XSPEC12> newpar 3-4
// The program will prompt for a specification for the 3rd
//   parameter (comp gives the name of the corresponding model
//   component)
comp:param3>0.001, 0
// which has its value set to 0.001 and its delta set to zero, fixing
//   it in later fits.The program now prompts for a specification for
//   the 4th parameter
comp:param4>21
// which is set to 21.As there is no 5th parameter, the program
//   displays a summary and returns to command level.
XSPEC12> newpar ,,.001
// The value of the delta of the 3rd parameter (which is the default
//   index as it was the first parameter modified in the previous
//   newpar invocation) is set to 0.001, allowing it to be adjusted
//   during any fits.
```

The total number of parameters for this example is eight.

```
XSPEC12> newpar 4 = 1
// The value of parameter 4 is set to the value of parameter 1.
//   This has the consequence of model parameter 4 being frozen at the
//   value of parameter 1 during subsequent fitting procedures.
//   If model parameter 1 is a free parameter, then both parameters
//   1 and 4 change their values simultaneously in the fit procedure.
XSPEC12> newpar 4 = p3/5 + 6.7
// The value of parameter 4 is set to the value of
//   (parameter 3/ parameter 5) plus 6.7
XSPEC12> newpar 6 = p3 * 0.1 - 9.5
// The value of parameter 6 is set to 0.1 times the
//   value of parameter 3 minus 9.5
XSPEC12> newpar 5 = 2 + 5.
// The value of parameter 5 is set to the value
//   of parameter 2 plus 5.
XSPEC12> newpar 8 = p1 / 4.6
// parameter 8 is set to parameter 1 divided by 4.6
XSPEC12> newpar 8 = abs(p1^3) / 2.0
// parameter 8 is set to the absolute value of the cube of
//   parameter 1 divided by 2.0
XSPEC12> newpar 5 = cos(p1) + sin(p3)
```

```
// parameter 5 is set to the cosine of parameter 1 plus the sine
//   of parameter 3
XSPEC12> newpar 3 = log(mymodel:p1)
// parameter 3 is set to the log (base 10) of parameter 1 in the
//   mymodel model
XSPEC12> untie 6
// Makes parameter 6 independent of parameter 3 and a free
//  parameter.
```

5.6.18 systematic

add a model-dependent systematic term to the variance

Syntax: systematic[<model systematic error>]

Set a systematic error term on the model to be added in quadrature to that on the data when evaluating chi-squared. The default value is zero.

5.6.19 untie (and runtie)

unlink previously linked parameters

Untie the specified parameter from any links to other parameters.

Syntax: **untie** <param range>

where <param range> is of the form

<param range> =:: [<modelName>:]<param #>

For **response parameters** (see **gain** command):

runtie <param range>

where <param range> is of the form

<param range> =:: [<sourceNum>:]<param #>

Parameters previously linked together with commands such as

XSPEC12>newpar <param spec>

are unlinked. The parameter will retain its current value for the next fit.

5.7 Plot Commands

5.7.1 cpd

set current plotting device

Syntax:	**cpd**	`<plot device>`
	cpd	`<filename>`
	cpd	`<filename>/{ps,cps,vps,vcps}`
	cpd	`none`

Set current plot device. The same can be achieved with the **setplot device** command, which takes the same options. In XSPEC12 as in previous versions, the plot device options are those allowed by the PGPLOT library.

When plotting to the screen, the most commonly used devices are `/xs` (`/xserve`) and `/xw` (`/xwindow`). If you select `/xs`, the plot window is persistent: it remains visible and in the selected position even after the XSPEC session is finished. With `/xw` the plot window closes at the end of the XSPEC session. Also note that on some platforms, when using `/xs` in multiple desktops, you might not see the window appear in a second desktop if it is still open in the first.

If the second argument does not start with a '/' character, which indicates that the string represents a PGPLOT device, it is taken to be a filename for Postscript output, and the default postscript driver will be used. The default postscript driver produces a monochrome plot in landscape orientation.

The filename argument can be followed by a '/' that specifies a particular postscript driver variant. Allowable variants are: `cps` (color postscript), `vps` (monochrome portrait orientation), and `vcps` (color portrait orientation), as well as the default, `ps`.

PGPLOT devices

A number of plot device types are supported in XSPEC. PGPLOT devices available on Unix machines are:

/GIF	Graphics Interchange Format file, landscape orientation
/VGIF	Graphics Interchange Format file, portrait orientation
/NULL	Null device, no output
/PPM	Portable Pixel Map file, landscape orientation
/VPPM	Portable Pixel Map file, portrait orientation
/PS	PostScript file, landscape orientation
/VPS	PostScript file, portrait orientation
/CPS	Colour PostScript file, landscape orientation
/VCPS	Colour PostScript file, portrait orientation
/TEK4010	Tektronix 4010 terminal
/GF	GraphOn Tek terminal emulator
/RETRO	Retrographics VT640 Tek emulator
/GTERM	Color gterm terminal emulator
/XTERM	XTERM Tek terminal emulator
/ZSTEM	ZSTEM Tek terminal emulator
/V603	Visual 603 terminal
/KRM3	Kermit 3 IBM-PC terminal emulator
/TK4100	Tektronix 4100 terminals
/VT125DEC	VT125 and other REGIS terminals
/XDISP	pgdisp or figdisp server
/XWINDOW	X window window@node:display.screen/xw
/XSERVE	An /XWINDOW window that persists for re-use

cpd none closes the device. For Postscript output, it flushes the write buffer into the file and closes the file. Note that in XSPEC12, each plot command produces a separate page in the postscript file, unlike previously where each plot overwrote the previous plot.

Examples:

Produce a set of color postscript plots in landscape orientation

```
// ... commands to produce a plot.
XSPEC12> cpd   dataplot.ps/cps
XSPEC12> plot data chi
```

```
XSPEC12> plot ufspec
XSPEC12> plot efficiency
XSPEC12> cpd none
```

Will produce 3 plots in the file `dataplot.ps`

Note, in contrast, that the **hardcopy** command will print only the plot that is currently in a graphics frame.

5.7.2 hardcopy

print plot

Spool the current plot to the printer.

Syntax: **hardcopy** [<filename>] [mono|color]

This command takes whatever is the current display in you plot window, writes it to a postscript file, and then sends it to a printer using the unix `lpr` command. It will thus be printed on whatever printer `lpr` uses as your default printer. If a filename is specified, the postscript file will be saved (e.g. `hardcopy dataplot.ps color` will produce a color plot saved in the file `dataplot.ps`). If `mono` or `color` is not given, the hardcopy will be monochrome.

5.7.3 iplot

make a plot, and leave XSPEC in interactive plotting mode

Interactive plotting on the current plot device.

Syntax: **iplot** <plot type>

This command works like the **plot** command (see the **plot** command description), but allows the user to change the plot and to add text to the plot interactively using the PLT package. See the Overview of PLT in the Appendices for more information.

5.7.4 plot

make a plot

Make one or more plots to the current plot device (see **cpd** or **setplot device**).

Syntax: **plot** `<plot type> [<plot type>] [<plot type>] ...`

`<plot type>` is a keyword describing the various plots allowed. Up to six plot panes can be put on a single page by combining multiple `<plot type>` options. For example:

```
XSPEC12> plot data resid ratio model
```

will produce a 4-pane plot. However contour plots may not be combined with other plots in this manner. When a certain plot type takes additional arguments (eg. `chain`, `model`), simply list them in order prior to specifying the next plot type:

```
XSPEC12> plot chain 3 4 data ufspec
```

In multi-pane plots, XSPEC will determine if two consecutive plot types may share a common X-axis (e.g. `plot data delchi`, or `plot counts ratio`). If so, the first pane will be stacked directly on top of the second. (Note that the small subset of multi-pane plots that were allowed in earlier versions of XSPEC all belonged in this category.)

For changing plot units, see **setplot energy** and **setplot wave**. Also see **iplot** for performing interactive plots.

background

Plot only the background spectra (with folded model, if defined). To plot both the data and background spectra, use `plot data` with the `setplot background` option.

chain

Plot a Monte Carlo Markov chain.

`plot chain [thin <n>] <par1> [<par2>]`

Chains must be currently loaded (see **chain** command), and `<par1>` and `<par2>` are parameter identifiers of the form `[<model name>:]<n>` where `<n>` is an integer, specifying the parameter columns in the chain file to serve as the X and Y axes respectively. To select the fit-statistic column, enter '0' for the `<par>` value. If `<par2>` is omitted, `<par1>` is simply plotted against row number.

Use the `thin <n>` option to display only 1 out of every `<n>` chain points. Example:

```
# plot one in five chain points,
# using parameters 1 and 4 for (X,Y)
plot chain thin 5 1 4
```

The `thin` value will be retained for future chain plots until it is reset. Enter `thin 1` to remove thinning.

chisq

Plot contributions to chisq. The contribution is plotted +ve or -ve depending on whether the residual is +ve or -ve.

contour

Plot the results of the last **steppar** run. If this was over one parameter then a plot of statistic versus parameter value is produced while a **steppar** over two parameters results in a fit-statistic contour plot.

`plot contour [<min fit stat> [<# levels> [<levels>]]]`

where `<min fit stat>` is the minimum fit statistic relative to which the delta fit statistic is calculated, `<# levels>` is the number of contour levels to use and `<levels>` := `<level1>` ... `<levelN>` are the contour levels in the delta fit statistic. `contour` will plot the fit statistic grid calculated by the last **steppar** command (which should have gridded on two parameters). A small plus sign '+' will be drawn on the plot at the parameter values corresponding to the minimum found by the most recent fit.

The fit statistic confidence contours are often drawn based on a relatively small grid (i.e., 5x5). To understand fully what these plots are telling you, it is useful to know a couple of points concerning how the software chooses the location of the contour lines. The contour plot is drawn based only on the information contained in the sample grid. For example, if the minimum fit statistic occurs when parameter 1 equals 2.25 and you use `steppar 1 1.0 5.0 4`, then the grid values closest to the minimum are 2.0 and 3.0. This could mean that there are no grid points where delta-fit statistic is less than your lowest level (which defaults to 1.0). As a result, the lowest contour will not be drawn. This effect can be minimized by always selecting a **steppar** range that causes XSPEC to step very close to the true minima.

For the above example, using `steppar 1 1.25 5.25 4`, would have been a better selection. The location of a contour line between grid points is designated using a linear interpolation. Since the fit statistic surface is often quadratic, a linear interpolation will result in the lines being drawn inside the true location of the contour. The combination of this and the previous effect sometimes will result in the minimum found by the **fit** command lying outside the region enclosed by the lowest contour level.

A grey-scale image of the data being contoured is also plotted. This can be removed by using the PLT command `image off`.

Examples:

```
XSPEC12> steppar 2 0.5 1. 4 3 1. 2. 4
   // create a  grid for parameters 2 and 3
XSPEC12> plot contour
   // Plot out a grid with three contours with
   //  delta fit statistic of 2.3, 4.61 and 9.21
XSPEC12> plot cont,,4,1.,2.3,4.61,9.21
   // same as above, but with a delta fit statistic = 1 contour.
```

counts
Plot the data (with the folded model, if defined) with the y-axis being numbers of counts in each bin.

data
Plot the data (with the folded model, if defined).

delchi
Plot the residuals in terms of sigmas with error bars of size one. In the case of the `cstat` and related statistics this plots (data-model)/error where error is calculated as the square root of the model predicted number of counts. Note that in this case this is not the same as contributions to the statistic.

dem
Plot a histogram of the relative contributions of plasma at different temperatures for multi-temperature models. This is not very clever at the moment and only plots the last model calculated.

eemodel
See model.

eeufspec
See ufspec.

efficien
Plot the total response efficiency versus incident photon energy.

emodel
See model.

eqw
Plot the probability density of the most recently run **eqwidth** calculation with error estimate.

eufspec
See ufspec.

goodness
Plot a histogram of the statistics calculated for each simulation of the most recent **goodness** command run.

icounts
Integrated counts and folded model. The integrated counts are normalized to unity.

insensitv
Plot the insensitivity of the current spectrum to changes in the incident spectra (experimental).

integprob
Plot the integrated probability distribution from the results of the most recently run **margin** command (must be a 1-D or 2-D distribution). The integrated probability is calculated by summing bins in decreasing order of probability. A grey-scale image of the data being contoured is also plotted. This can be removed by using the PLT command `image off`.

lcounts
Plot the data (with the folded model, if defined) with a logarithmic y-axis indicating the count spectrum

ldata
Plot the data (with the folded model, if defined) with a logarithmic y-axis.

margin
Plot the probability distribution from the results of the most recently run **margin** command (must be a 1-D or 2-D distribution). A grey-scale image of the data being contoured is also plotted. This can be removed by using the PLT command `image off`.

model, emodel, eemodel
Plot the current incident model spectrum (Note: This is NOT the same as an unfolded spectrum.) The contributions of the various additive components are also plotted. If using a named model, the model name should be given as an additional argument. `emodel` plots $Ef(E)$ or, if plotting wavelength, $\lambda f(\lambda)$. `eemodel` plots $E^2 f(E)$, or if plotting wavelength, $\lambda^2 f(\lambda)$. The E (or λ) used in the multiplicative factor is taken to be the geometric mean of the lower and upper energies of the plot bin.

ratio
Plot the data divided by the folded model.

residuals
Plot the data minus the folded model.

sensitvity
Plot the sensitivity of the current spectrum to changes in the incident spectra (experimental).

sum A pretty plot of the data and residuals against both channels and energy.

ufspec, eufspec, eeufspec
Plot the unfolded spectrum and the model. The contributions to the model of the various additive components also are plotted. WARNING ! This plot is not model-independent and your unfolded model points will move if the model is changed. The data points plotted are calculated by D*(unfolded model)/(folded model), where D is the observed data, (unfolded model) is the theoretical model integrated over the plot bin, and (folded model) is the model times the response as seen in the standard `plot data`. `eufspec` plots the unfolded spectrum and model in $Ef(E)$, or if plotting wavelength, $\lambda f(\lambda)$. `eeufspec` plots the unfolded spectrum and model in $E^2 f(E)$, or if plotting wavelength, $\lambda^2 f(\lambda)$. The E (or λ) used in the multiplicative factor is taken to be the geometric mean of the lower and upper energies of the plot bin.

5.7.5 setplot

modify plotting parameters

Set one of the various plot options.

Syntax: **setplot** <`subcommand string`>

where <`subcommand string`> is a keyword followed in some cases by arguments. Current settings of all **setplot** items can be viewed with **show plot**.

add, noadd
Switch on (add) and off (noadd) individual additive model components on data plots.

area, noarea
After `setplot area` is entered, `plot data` and `plot ldata` will show the data divided by the response effective area for each particular channel. `plot residuals` will necessarily also be affected by this. Usual plotting is restored by `setplot noarea`. If data is associated with more than 1 response, the response effective area is calculated by simply summing the contributions from each response.

background, nobackground
When running `plot data` or `plot ldata`, also show associated background spectra (if any).

channel
Change the x-axis on data and residual plots to channels.

command

Add a PLT command to the command list.

`setplot command <PLT command>`

where `<PLT command>` is any valid PLT command. Every time you use **setplot command**, that command is added to the list that is passed to PLT when you use **plot** or **iplot**. The most common use of **setplot command** is to add a common label to all plots produced. You should be careful when using this command, because XSPEC does not check to see if you have entered a valid PLT command.These commands are appended to the list that XSPEC creates to generate the plot and so **setplot command** will override these values (this can either be a bug or a feature, depending on what you have done!) See also **setplot delete** and **setplot list**.

Example:

```
XSPEC12> setp co LA OT Crab  #Add the label "Crab" to future plots.
```

contimage, nocontimage

Switch on (contimage) and off (nocontimage) the background image on contour plots.

delete

Delete a PLT command from the command list.

`setplot delete [all|<command #>-<command #>|<command #>]`

where `<command #>` is the number of a PLT command that had been entered previously using `setplot command`. This command is used to delete commands from the list passed to PLT when you use the XSPEC **plot** or **iplot** commands.

device

Set current plot device.

```
XSPEC12>setplot device  <plot device>
XSPEC12>setplot device <filename>
XSPEC12>setplot device <filename>/{ps,cps,vps,vcps}
XSPEC12>setplot device  none
```

If the second argument does not start with a '/' character, which indicates that the string represents a PGPLOT device, it is taken to be a filename for Postscript output, and the default postscript driver will be used. The default postscript driver produces a monochrome plot in landscape orientation.

The filename argument can be followed by a '/' that specifies a particular postscript driver variant. Allowable variants are: cps (color postscript),

vps (monochrome portrait orientation), and vcps (color portrait orientation), as well as the default, ps.

A number of plot device types are supported in XSPEC. PGPLOT devices available on Unix machines are :

/GIF	Graphics Interchange Format file, landscape orientation
/VGIF	Graphics Interchange Format file, portrait orientation
/NULL	Null device, no output
/PPM	Portable Pixel Map file, landscape orientation
/VPPM	Portable Pixel Map file, portrait orientation
/PS	PostScript file, landscape orientation
/VPS	PostScript file, portrait orientation
/CPS	Colour PostScript file, landscape orientation
/VCPS	Colour PostScript file, portrait orientation
/TEK4010	Tektronix 4010 terminal
/GF	GraphOn Tek terminal emulator
/RETRO	Retrographics VT640 Tek emulator
/GTERM	Color gterm terminal emulator
/XTERM	XTERM Tek terminal emulator
/ZSTEM	ZSTEM Tek terminal emulator
/V603	Visual 603 terminal
/KRM3	Kermit 3 IBM-PC terminal emulator
/TK4100	Tektronix 4100 terminals
/VT125DEC	VT125 and other REGIS terminals
/XDISP	pgdisp or figdisp server
/XWINDOW	X window window@node:display.screen/xw
/XSERVE	An /XWINDOW window that persists for re-use

Examples:

```
XSPEC12> setplot device /xt
   // sets the device to the xterm.
XSPEC12> setplot device none
   // closes the plot file.
```

energy

Change the X-axis on plots to energies, and optionally change the units.

`setplot energy [<units>]`

where `<units>` is an optional string for modifying X-axis energy units. Valid choices currently are: `keV, MeV, GeV`, and `Hz`, which are case-insensitive and can be abbreviated. Energy units initially default to `keV`. The selection made here also determines the units in the **ignore** and **notice** energy range specifiers.

Where applicable, Y-axis units will be modified to match the X-axis selection. The exception is for the choice of Hz when emodel/eufspec is in Jy and eemodel/eeufspec in ergs/cm^2/s.

group

Define a range of spectra to be in the same group for plotting purposes only.

`setplot group <spectrum range>...`

where `<spectrum range>` is a range of contiguous spectra to be treated as a single spectrum for plotting purposes. The spectra still are fit individually. If multiple ranges are given, each range becomes a single group. Initially, all spectra read in are treated as single spectra. (See also `ungroup`.)

Examples:

Assume that there are five spectra currently read in, all of them ungrouped initially.

```
XSPEC12> setplot group 1-4
   //The first four spectra are treated as one group, with the fifth
   //  spectra on its own. Thus all plots will appear to have two spectra.
XSPEC12> setplot group 1 2 3 4
   //The spectra are reset to each be in their own group.
XSPEC12> setplot group 2-3 4-5
   //Now there are three plot groups, being spectrum 1, by itself, and
   //  spectra 2-3 and 4-5 as groups.
XSPEC12> setplot group 1-**
   //All the spectra are placed in a single plot group.
```

id, noid

Switch on (id) and off (noid) plotting of line IDs.

`setplot id <temperature> <emissivity limit> <redshift> <lowEng> <highEng>`

The IDs are taken from the APEC line list for the temperature given by the first argument. The plot only shows those lines with emissivities above the limit set and the lines are redshifted by the amount specified. Only lines between the low and high energy values will be shown. If both the low and high energy values are zero then energy range used is that plotted. When plotting with a wavelength x-axis the low and high values given are assumed to be in wavelength units. The APEC version is the current default unless `xset apecroot` has been used to reset the APEC files then `setplot id` uses a filename based on the value of `apecroot` as described in the documentation for the **apec** model.

list

List all the PLT commands in the command list.

`setplot list`

See `setplot delete` for an example of use.

rebin

Define characteristics used in rebinning the data (for plotting purposes ONLY).

`setplot rebin <min significance> <max # bins> <plot group> <error type>`

In plotting the data from a spectrum (or group of spectra, see `setplot group`), adjacent bins are combined until they have a significant detection at least as large as `<min significance>` (in σ).However, no more than `<max # bins>` may be so combined. Initial values are 0. and 1, respectively. This argument effects only the presentation of the data in plots. It does not change the fitting, in particular the number of degrees of freedom. The values given are applied to all the plotted data in the plot group specified as the final argument. To change the rebinning simultaneously for all the plot groups give a negative value of the plot group.

The `<error type>` argument specifies how to calculate the error bars on the new bins. The default is `quad` which sums in quadrature the errors on the original bins. `sqrt` uses $\sqrt{N}$ where N is the number of counts in the new bin, `poiss-1` uses $1 + \sqrt{N + 0.75}$, `poiss-2` uses $\sqrt{N - 0.25}$, and `poiss-3` is the arithmetic mean of `poiss-1` and `poiss-2`. If background is present its error is calculated by the same method then added in quadrature to the source error.

Examples:

```
XSPEC12> setplot rebin 3 5 1
   //Bins in plot group 1 are plotted that have at least 3 sigma,
   // or are grouped in sets of 5 bins.
XSPEC12> setplot rebin 5 5
   //The significance is increased to 5 sigma.
XSPEC12> setplot rebin,,10,-1
   //All plotted bins can be grouped into up to 10 bins in reaching the
   // 5 sigma significance criterion.
XSPEC12> setplot rebin ,,,sqrt}
   //Uses sqrt(N) to calculate error bars.
```

redshift

Apply a redshift to the X-axis energy and wavelength values.

`setplot redshift <z>`

This will multiply X-axis energies by a factor of (1+z) to allow for viewing in the source frame. Y-axis values will be equally affected in plots which are normalized by energy or wavelength. Note that this is not connected in any way to redshift parameters in the model (or the `setplot id` redshift parameter) and should only be used for illustrative purposes.

splashpage (on|off)

When set to off, the usual XSPEC version and build date information will not be printed to the screen when the first plot window is initially opened. This is intended primarily for the HERA installation of XSPEC.

ungroup

Remove previous grouping set up by `setplot group`, resetting all spectra to be in a distinct plot group.

wave

Change the x-axis on plots to wavelength, and optionally change the units.

```
setplot wave [<units>]
setplot wave perhz [off]
```

where `<units>` is an optional string for modifying X-axis wavelength units. Valid choices currently are: `angstom, cm, micron`, and `nm`, which are case-insensitive and can be abbreviated. Wavelength units initially default to `angstrom`.

Where applicable, Y-axis units will be modified to match the X-axis selection. However this behavior can be changed by the command `setplot wave perhz`, which will cause Y-axis units to be in 1/Hz. This feature is turned off by `setplot wave perhz off`, and its initial setting is determined by the WAVE_PLOT_UNITS setting in the user's `~/.xspec/Xspec.init` file. Also note that when `perhz` is selected, emodel/eufspec and eemodel/eeufspec will have the same Y-axis units as for `setplot energy hz`.

This command makes **ignore** and **notice** operate in terms of wavelength rather than energies. The units setting here also determines the units in the **ignore** and **notice** range specifiers.

xlog (on|off)

Set the x-axis to logarithmic or linear respectively for energy or wavelength plots. `xlog` has no effect on plots in channel space (recall that the default for energy plots is logarithmic: `xlog` allows the user to override this setting). `xlog` and `ylog` will not work for model-related plots (eg. model, ufspec, and their variants) as their axes are always set to log scale.

ylog (on|off)

Set the y-axis to logarithmic or linear respectively for energy or wavelength plots. For plot instructions that are explicitly logarithmic (`plot ldata`, `plot lcounts`) the state of the `ylog` setting is ignored. `xlog` and `ylog` will not work for model-related plots (eg. model, ufspec, and their variants) as their axes are always set to log scale.

5.8 Setting Commands

5.8.1 abund

set the solar abundances

Set the abundance table used in the plasma emission and photoelectric absorption models.

Syntax: **abund** <`option`>

where <`option`> is:

`angr`	from Anders E. & Grevesse N. (1989, Geochimica et Cosmochimica Acta 53, 197)
`aspl`	from Asplund M., Grevesse N., Sauval A.J. & Scott P. (2009, ARAA, 47, 481)
`feld`	from Feldman U.(1992, Physica Scripta 46, 202 except for elements not listed which are given grsa abundances)
`aneb`	from Anders E. & Ebihara (1982, Geochimica et Cosmochimica Acta 46, 2363)
`grsa`	from Grevesse, N. & Sauval, A.J. (1998, Space Science Reviews 85, 161)
`wilm`	from Wilms, Allen & McCray (2000, ApJ 542, 914 except for elements not listed which are given zero abundance)
`lodd`	from Lodders, K (2003, ApJ 591, 1220)
`file filename`	where `filename` is an ASCII file containing 30 lines with one number on each line. All abundances are number relative to H.

The tables are:

Element	angr	aspl	feld	aneb	grsa	wilm	lodd
H	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00
He	9.77e-02	8.51e-02	9.77e-02	8.01e-02	8.51e-02	9.77e-02	7.92e-02
Li	1.45e-11	1.12e-11	1.26e-11	2.19e-09	1.26e-11	0.00	1.90e-09
Be	1.41e-11	2.40e-11	2.51e-11	2.87e-11	2.51e-11	0.00	2.57e-11
B	3.98e-10	5.01e-10	3.55e-10	8.82e-10	3.55e-10	0.00	6.03e-10
C	3.63e-04	2.69e-04	3.98e-04	4.45e-04	3.31e-04	2.40e-04	2.45e-04
N	1.12e-04	6.76e-05	1.00e-04	9.12e-05	8.32e-05	7.59e-05	6.76e-05
O	8.51e-04	4.90e-04	8.51e-04	7.39e-04	6.76e-04	4.90e-04	4.90e-04
F	3.63e-08	3.63e-08	3.63e-08	3.10e-08	3.63e-08	0.00	2.88e-08
Ne	1.23e-04	8.51e-05	1.29e-04	1.38e-04	1.20e-04	8.71e-05	7.41e-05

Na	2.14e-06	1.74e-06	2.14e-06	2.10e-06	2.14e-06	1.45e-06	1.99e-06
Mg	3.80e-05	3.98e-05	3.80e-05	3.95e-05	3.80e-05	2.51e-05	3.55e-05
Al	2.95e-06	2.82e-06	2.95e-06	3.12e-06	2.95e-06	2.14e-06	2.88e-06
Si	3.55e-05	3.24e-05	3.55e-05	3.68e-05	3.55e-05	1.86e-05	3.47e-05
P	2.82e-07	2.57e-07	2.82e-07	3.82e-07	2.82e-07	2.63e-07	2.88e-07
S	1.62e-05	1.32e-05	1.62e-05	1.89e-05	2.14e-05	1.23e-05	1.55e-05
Cl	3.16e-07	3.16e-07	3.16e-07	1.93e-07	3.16e-07	1.32e-07	1.82e-07
Ar	3.63e-06	2.51e-06	4.47e-06	3.82e-06	2.51e-06	2.57e-06	3.55e-06
K	1.32e-07	1.07e-07	1.32e-07	1.39e-07	1.32e-07	0.00	1.29e-07
Ca	2.29e-06	2.19e-06	2.29e-06	2.25e-06	2.29e-06	1.58e-06	2.19e-06
Sc	1.26e-09	1.41e-09	1.48e-09	1.24e-09	1.48e-09	0.00	1.17e-09
Ti	9.77e-08	8.91e-08	1.05e-07	8.82e-08	1.05e-07	6.46e-08	8.32e-08
V	1.00e-08	8.51e-09	1.00e-08	1.08e-08	1.00e-08	0.00	1.00e-08
Cr	4.68e-07	4.37e-07	4.68e-07	4.93e-07	4.68e-07	3.24e-07	4.47e-07
Mn	2.45e-07	2.69e-07	2.45e-07	3.50e-07	2.45e-07	2.19e-07	3.16e-07
Fe	4.68e-05	3.16e-05	3.24e-05	3.31e-05	3.16e-05	2.69e-05	2.95e-05
Co	8.32e-08	9.77e-08	8.32e-08	8.27e-08	8.32e-08	8.32e-08	8.13e-08
Ni	1.78e-06	1.66e-06	1.78e-06	1.81e-06	1.78e-06	1.12e-06	1.66e-06
Cu	1.62e-08	1.55e-08	1.62e-08	1.89e-08	1.62e-08	0.00	1.82e-08
Zn	3.98e-08	3.63e-08	3.98e-08	4.63e-08	3.98e-08	0.00	4.27e-08

Examples:

5.8.2 cosmo

set the cosmology

Set the cosmology used (i.e., H_0, q_0, and Λ_0).

Syntax: **cosmo** $<H_0>$ $<q_0>$ $<\Lambda_0>$

where $<H_0>$ is the Hubble constant in km/s/Mpc, $<q_0>$ is the deceleration parameter, and $<\Lambda_0>$ is the cosmological constant. If the cosmological constant is non-zero then at present XSPEC requires that the universe is flat. In this case the value of $<q_0>$ will be ignored and XSPEC will assume that $\Omega_{matter} \geq 1-\Lambda_0$. The default values are $<H_0> = 70$, $<q_0> = 0.0$, and $<\Lambda_0> = 0.73$

Examples:

```
XSPEC12>cosmo 100
// Set <H0> = 100 km/s/Mpc
XSPEC12>cosmo ,0
// Set <q0> = 0
XSPEC12>cosmo ,,0.7
// Set a flat universe with <Λ0> = 0.7
```

5.8.3 method

change the fitting method

Set the minimization method.

Syntax: method`<algorithm> [<# of trials/evaluations> [<critical delta>] [<method-specific options>]]`

where `<algorithm>` is the method in use and the other arguments are control values for the minimization. Their meanings are explained under the individual methods. The `migrad` and `simplex` methods are taken from the CERN Minuit2 package, with documentation located at http://seal.web.cern.ch/seal/MathLibs/Minuit2/html/index.html. If either of these are used, then the **error** command will use the Minuit2 minos method to find the confidence regions.

leven

method `leven [<# of eval> [<crit delta>] [<crit beta>]] [delay |nodelay]`

The default XSPEC minimization method using the modified Levenberg-Marquardt algorithm based on the CURFIT routine from Bevington. `<# of eval>` is the

number of trial vectors before the user is prompted to say whether they want to continue fitting. <`crit delta`> is the convergence criterion, which is the (absolute, not fractional) difference in fit statistic between successive iterations, less than which the fit is determined to have converged.

<`crit beta`> refers to the |beta|/N value reported during a fit. This is the norm of the vector of derivatives of the statistic with respect to the parameters divided by the number of parameters. At the best fit this should be zero, and so provides another measure of how well the fit is converging. When this is set to a positive value, it will provide another fit stopping criterion in addition to that of the <`crit delta`> setting.

Including the string `delay` as an argument turns on delayed gratification. It is turned off by `nodelay`. Delayed gratification modifies the way the damping parameter is set and has been shown in many cases to speed up convergence. The default is `nodelay`.

<`# of eval`>, <`crit delta`>, <`crit beta`>, `delay`, and `nodelay` may also be set through the **fit** command.

This method requires an estimate of the second derivative of the statistic with respect to the parameters. By default, XSPEC calculates these using an analytic expression which assumes that partial 2nd derivatives of the model with respect to its parameters may be ignored. This may be changed by setting the USE_NUMERICAL_DIFFERENTIATION flag to "true" in the user's startup `Xspec.init` initialization file. XSPEC will then calculate all second derivatives numerically, which can be noticeably slower.

migrad

method `migrad [<# of eval>]`

The Minuit2 migrad method. <`# of eval`> is the number of function evaluations to perform before giving up. Migrad uses an internal convergence criterion.

The current version of Minuit2 included is that from ROOT v5.34. Documentation on Minuit2 can be found at http://seal.web.cern.ch/seal/MathLibs/Minuit2/html/.

If migrad is not working well try experimenting with different hard and soft limits on parameters.

simplex

method `simplex [<# of evaluations>]`

The Minuit2 simplex method. <# of evaluations> is the number of function evaluations to perform before giving up. Simplex uses an internal convergence criterion. This method is included for historical interest and is almost always outperformed by migrad.

5.8.4 statistic

change the objective function (statistic) for the fit

Change the fit or test statistic in use, for one or more spectra.

Syntax:	**statistic**	`[chi |cstat |lstat |pgstat |pstat |whittle[#]] [<spectrum range>]`
	statistic test	`[ad |chi |cvm |ks |pchi |runs] [<spectrum range>]`

The fit statistic options are chi-squared (`chi`), C statistic (`cstat`), Loredo statistic (`lstat`), a statistic for Poisson data with assumed known background (`pstat`), a statistic for Poisson data with Gaussian background (`pgstat`), and the Whittle statistic (`whittle`) for power density functions. If the statistic is given as `whittle` with a number appended (e.g. whittle5) then the statistic is appropriate for that number of power density functions averaged together.

The **test** statistic options are Anderson-Darling (`ad`), chi-squared (`chi`), Cramer-von Mises (`cvm`), Kolmogorov-Smirnov (`ks`), Pearson chi-square (`pchi`) and Runs (`runs`). These statistics are described in the appendix on Statistics in XSPEC.

If a spectrum number or spectrum range is given, the chosen statistic will only apply to those spectra. It is therefore possible for a multi-spectrum fit to use more than one fit or test statistic. If no spectrum number or range is given, the chosen statistic will apply to all loaded spectra and will be the default statistic for any future loaded spectra.

Note that if the chosen statistic is not compatible with the currently used **weight** method, the **weight** method will be changed to standard weighting until the conflict is removed.

Examples:

Assume 3 spectra are currently loaded, all using the chi-squared statistic, and that chi-squared is the default statistic.

```
XSPEC12>statistic cstat 2-3
   // Spectrum 1 continues to use chi-sq, 2 and 3 use cstat.
```

```
XSPEC12>data 4 spec4.pha
   // New spectrum 4 will use chi-sq.
XSPEC12>statistic cstat
   // All 4 spectra now use cstat, cstat is the new default.
XSPEC12>data 5 spec5.pha
   // New spectrum 5 will use cstat.
XSPEC12>statistic test ks
   // All 4 spectra now use ks as the test statistic.
```

5.8.5 xsect

set the photoionization cross-sections

Change the photoelectric absorption cross-sections in use.

Syntax: **xsect** [bcmc|obcm|vern]

The three options are: `bcmc`, from Balucinska-Church & McCammon (1992; Ap.J.400, 699) with a new He cross-section based on (1998; Ap.J. 496, 1044); `obcm`, as `bcmc` but with the old He cross-section, and, `vern`, from Verneret. al. (1996 Ap.J.). This changes the cross-sections in use for all absorption models with the exception of **wabs** and **tbabs**.

5.8.6 xset

set variables for XSPEC models

Modify a number of XSPEC internal switches.

Syntax: **xset** [abund |cosmo |delta |mdatadir |method |seed |statistic |weight |xsect |<string_name>] [<options> |<string_value>]

The arguments `abund, cosmo, method, statistic, weight`, and `xsect` just run the appropriate XSPEC commands. `mdatadir` changes the directory in which XSPEC searches for model data files. You probably don't want to change this. The `seed` option requires an integer argument, which will then be used to immediately re-seed and re-initialize XSPEC's random-number generator.

The `delta` option is for setting fit delta values (see the **newpar** command) which are proportional to the current parameter value rather than fixed. For example,

```
XSPEC12> xset delta .15
```

will set each parameter fit delta to .15 * parVal. To turn proportional deltas off and restore the original fixed deltas, set `delta` to a negative value or 0.0. The current proportional delta setting can be seen with **show control**.

The <`string_name`> option can be used to pass string values to models. XSPEC maintains a database of <`string_name`>, <`string_value`> pairs created using this command. Individual model functions can then access this database. Note that **xset** does no checking on whether the <`string_name`> is used by any model so spelling errors will not be trapped.

To access the <`string_name`>, <`string_value`> database from within a model function use the fortran function `fgmstr`. This is defined as `character*128` and takes a single argument, the string name as a `character*128`. If the <`string_name`> has not been set then a blank string will be returned.

The current <`string_name`> options, models to which they apply and brief descriptions are given in the following table :

APECROOT	apec, vapec, bapec, bvapec, equil, vequil, npshock, vnpshock, pshock, vpshock, sedov, vsedov, c6mekl, c6vmekl, c6pmekl, c6pvmekl, cemkl, cevmkl, mekal, vmekal, mkcflow, vmclow	Switch from default APEC input files.
APECTHERMAL	apec, vapec, bapec, bvapec, equil, vequil, npshock, vnpshock, pshock, vpshock, sedov, vsedov, c6mekl, c6vmekl, c6pmekl, c6pvmekl, cemkl, cevmkl, mekal, vmekal, mkcflow, vmclow	Thermally broaden emission lines in APEC input files.
APECVELOCITY	apec, vapec, bapec, bvapec, equil, vequil, npshock, vnpshock, pshock, vpshock, sedov, vsedov, c6mekl, c6vmekl, c6pmekl, c6pvmekl, cemkl, cevmkl, mekal, vmekal, mkcflow, vmclow	Velocity broaden emission lines in APEC input files.
NEIAPECROOT	gnei, nei, vgnei, nvei, equil, vequil, npshock, vnpshock, pshock, vpshock, sedov, vsedov	Switch from default NEIAPEC input files.
POW_EMIN, POW_EMAX	powerlaw, bknpower, bkn2pow, cutoffpl	Switch to normalize to a flux calculated over an energy range.

NEIVERS	gnei, nei, vgnei, vnei, equil, vequil, npshock, vnpshock, pshock, vpshock, sedov, vsedov	Switch NEIAPEC version number.
CFLOW_VERSION	mkcflow, vmclow	Switch CFLOW version number.
CFLOW_NTEMPS	mkcflow, vmclow	Switch number of temperature bins used in CFLOW model.
SUZPSF-IMAGE	suzpsf	Set image file to be used for surface brightness.
SUZPSF-RA	suzpsf	Set RA for center surface brightness map which is taken from the WMAP.
SUZPSF-DEC	suzpsf	Set Dec for center surface brightness map which is taken from the WMAP.
SUZPSF-MIXFACT-IFILE#	suzpsf	Set filename to read mixing factors.
SUZSF-MIXFACT-OFILE#	suzpsf	Set filename to write mixing factors.
XMMPSF-IMAGE	xmmpsf	Set image file to be used for surface brightness.
XMMPSF-RA	xmmpsf	Set RA for center surface brightness map which is taken from the WMAP.
XMMPSF-DEC	xmmpsf	Set Dec for center surface brightness map which is taken from the WMAP.
XMMPSF-MIXFACT-IFILE#	xmmpsf	Set filename to read mixing factors.
XMMPSF-MIXFACT-OFILE#	xmmpsf	Set filename to write mixing factors.
NSA_FILE	nsa	Change filename used for model data.
NSAGRAV_DIR	nsagrav	Change directory used for model data files.
NSMAX_DIR	nsmax	Change directory used for model data files.

ZXIPCF_DIR	zxipcf	Change directory used for model data files.

Examples:

```
XSPEC12> xset neivers 2.0
   // Set the NEIVERS variable to 2.0
XSPEC12> xset
   // List the current string variables
XSPEC12> xset apecroot /foo/bar/apec_v1.01
   // Set the APECROOT variable
XSPEC12> xset seed 1515151
   // Re-initialize the pseudo random-number generator
   // with the seed value 1515151
```

5.9 Tcl Scripts

5.9.1 lrt

likelihood ratio test between two models

Tcl script to perform a likelihood ratio test between two models.

Syntax: lrt `<niter> <model0_name> <model1_name> [<filename>]`

Runs `<niter>` simulations of datasets based on `<model0_name>`, calculates the likelihood ratio for `<model1_name>` relative to `<model0_name>` (calculated by the statistic for `<model0_name>` minus the statistic for `<model1_name>`), and outputs the fraction of iterations with the likelihood ratio smaller than that for the data. If the optional filename is given then the simulation results are written to the file. The first line of the file contains the results for the data, the other lines the simulations. Each line comprises the statistic values for `<model0_name>`, the statistic value for `<model1_name>`, and the difference.

Before running this procedure you must have created command files called `<model0_name>`.xcm and `<model1_name>`.xcm which define the two models. A good way to do this is to set up the model then use save model to make the command file.

5.9.2 multifake

perform multiple fakeit iterations and save to file

Tcl script to perform many iterations of fakeit and save the results in a FITS file.

Syntax: **multifake** <time> <niter> <outfile>

This script runs <niter> iterations of fakeit with an exposure of <time> and writes the results to <outfile>. Before running this procedure you have read in one (and only one) dataset along with its response and optional background and arf files. You must also have defined the model.

The output file is a FITS binary table with the columns being the value fit for each parameter in each iteration. The final column is the statistic value for that iteration.

Note that if an error occurs during the fit of a faked spectrum then -999 is written for all parameters and the statistic value for that iteration.

5.9.3 rescalecov

rescale the covariance matrix

Tcl script to rescale the entire covariance matrix used in the proposal chain command.

Syntax: **rescalecov** <scale>

Rescales the chain proposal distribution covariance matrix by the factor input as <scale>.

5.9.4 simftest

estimate the F-test probability for adding a component

Tcl script to generate simulated datasets and use these to estimate the F-test probability for adding a model component.

Syntax: **simftest** <model_comp> <niter> [<filename>]

This script runs <`niter`> sets of simulated datasets to estimate the F-test probability for adding the additional model component number <`model_comp`>. If <`filename`> is specified then passes this to `lrt.tcl` to save likelihood ratio simulation information. The first line of the file written contains the results for the data, the other lines for the simulations. Each line comprises the statistic value for the model without <`model_comp`>, that for the model with <`model_comp`>, and the difference.

Before running this script the model should be set up including the additional component to be tested. The script will create temporary files `model_with_comp.xcm` and `model_without_comp.xcm`.

5.9.5 writefits

write information about the current fit and errors to a FITS file

Tcl script to dump a lot of useful information to a FITS file.

Syntax: **writefits** <`FITS filename`>

This script writes filenames, free parameter values and errors to one row of a FITS file. The error command should have been run on all the free parameters before running this script. If the FITS file already exists then a new row is appended.

Chapter 6

XSPEC Models

6.1 Alphabetical Summary of Models

Table 6.1: Summary of Models

Model	Description
absori	Ionized absorber.
acisabs	Extra absorption due to contamination on the ACIS filters.
agauss, zagauss	Gaussian line profile in wavelength space.
ascac	ASCA PSF mixing model.
apec, **vapec**, **vvapec**	APEC thermal plasma model.
atable	Additive table model.
bapec, **bvapec**, **bvvapec**	Velocity broadened APEC thermal plasma model.
bbody, **zbbody**	Blackbody spectrum, with redshift variant
bbodyrad	Blackbody spectrum with norm proportional to surface area.
bexrav	E-folded broken power-law reflected from neutral matter
bexriv	E-folded broken power-law reflected from ionized matter
bknpower	Broken powerlaw.
bkn2pow	Three-segment broken powerlaw.
bmc	Comptonization by relativistically moving matter.
bremss, **vbremss**, **zbremss**	Thermal bremsstrahlung, with redshift variant.
btapec, **bvtapec**, **bvvtapec**	Broadened APEC emission spectrum with separate continuum and line temperatures.
c6mekl, **c6pmekl**, **c6vmekl**, **c6pvmkl**	6th-order Chebyshev polynomial DEM using **mekal** and variants
cabs	Compton scattering (non-relativistic)

carbatm	Nonmagnetic carbon atmosphere of a neutron star
cemekl, **cevmkl**	Multi-temperature **mekal**.
cflow	Cooling flow model.
cflux	Calculate flux of other model components.
clumin	Calculate luminosityof other model components.
compbb	Comptonized blackbody spectrum after Nishimura et al. 1986.
compLS	Comptonization spectrum after Lamb and Sanford 1979.
compmag	Thermal and bulk Comptonization for cylindrical accretion onto the polar cap of a magnetized neutron star.
compPS	Comptonization spectrum after Poutanen and Svenson 1986.
compST	Comptonization spectrum after Sunyaev and Titarchuk 1980.
comptb	Thermal and bulk Comptonization of a seed blackbody-like spectrum.
compTT	Comptonization spectrum after Titarchuk 1994.
constant	Energy-independent multiplicative factor.
cpflux	Convolution model to calculate photon flux.
cplinear	Non-physical model for low count background spectra.
cutoffpl	Powerlaw with high energy exponential rolloff.
cyclabs	Cyclotron absorption line.
disk	Disk model.
diskbb	Multiple blackbody disk model.
diskir	Irradiated inner and outer disk.
diskline	Line emission from relativistic accretion disk.
diskm	Disk model with gas pressure viscosity.
disko	Modified blackbody disk model.
diskpbb	Accretion disk with power-law T(r)
diskpn	Accretion disk around a black hole.
dust	Dust scattering out of the beam.
edge, **zedge**	Absorption edge.
eplogpar	Log-parabolic blazar model with vFv normalization.
eqpair, **eqtherm**, **compth**	Paolo Coppi's hybrid hot plasma emission models.
equil, **vequil**	Equilibrium ionization collisional plasma model from Borkowski.
etable	Table model for exponential of -1 times the input.
expabs	Low-energy exponential rolloff.
expdec	Exponential decay
expfac	Exponential factor.
ezdiskbb	Multiple blackbody disk model with zero-torque inner boundary.
gabs	Gaussian absorption line.
gadem, **vgadem**	Plasma emission, multi-temperature with gaussian distribution of emission measure.
gauss, **zgauss**	Simple gaussian line profile.
gnei, **vgnei**	Generalized single ionization NEI plasma model.
grad	GR accretion disk around a black hole.
grbm	Gamma-ray burst model.
gsmooth	Gaussian smoothing with an energy dependent sigma.

hatm	Nonmagnetic hydrogen atmosphere of a neutron star.
heilin	Voigt absorption profiles for He I series.
highecut, **zhighect**	High energy cutoff.
hrefl	Simple reflection model good up to 15 keV.
ireflect	Reflection from ionized material.
ismabs	High resolution ISM absorption model.
kdblur	Convolve with the Laor model shape.
kdblur2	Convolve with the Laor2 model shape.
kerrbb	Multi-temperature blackbody model for thin accretion disk around a Kerr black hole.
kerrconv	Accretion disk line shape with BH spin as free parameter.
kerrd	Optically thick accretion disk around a Kerr black hole.
kerrdisk	Accretion disk line emission with BH spin as free parameter.
laor	Line from accretion disk around a black hole.
laor2	Line from accretion disk with broken power-law emissivity around a black hole.
logpar	Log-parabolic blazar model.
lorentz	Lorentzian line profile.
lsmooth	Lorentzian smoothing with an energy dependent sigma.
lyman	Voigt absorption profiles for H I or He II Lyman series.
meka, **vmeka**	Mewe-Gronenschild-Kaastra thermal plasma (1992).
mekal, **vmekal**	Mewe-Kaastra-Liedahl thermal plasma (1995).
mkcflow, **vmcflow**	Cooling flow model based on mekal.
mtable	Multiplicative table model.
nei, **vnei**	Simple nonequilibrium ionization plasma model.
nlapec	Continuum-only APEC emission spectrum.
notch	Notch line absorption.
npshock, **vnpshock**	Plane-parallel shock with ion and electron temperatures.
nsa	Neutron star with hydrogen atmosphere
nsagrav	Neutron star with hydrogen atmosphere for different g.
nsatmos	Neutron star H atmosphere with e- conduction and self-irradiation
nsmax	Neutron star magnetic atmosphere.
nsmaxg	Neutron star with a magnetic atmosphere.
nsx	Neutron star with a non-magnetic atmosphere.
nteea	Pair plasma model.
nthcomp	Thermally comptonized continuum.
optxagnf, **optxagn**	Colour temperature corrected disc and energetically coupled Comptonisation model for AGN.
partcov	Convert absorption model into a partial covering absorption.
pcfabs, **zpcfabs**	Partial covering fraction absorption.
pegpwrlw	Powerlaw with pegged normalization.
pexmon	Neutral Compton reflection with self-consistent Fe and Ni lines.
pexrav	Exponentially cut-off power-law reflected from neutral matter.
pexriv	Exponentially cut-off power-law reflected from ionized matter.
phabs, **vphabs**, **zphabs**, **zvphabs**	Photo-electric absorption

pileup	CCD pile-up model for Chandra
plabs	Absorption model with power-law dependence on energy.
plcabs	Cut-off powerlaw observed through dense, cold matter.
posm	Positronium continuum.
powerlaw, **zpowerlw**	Simple photon power law.
projct	3-D to 2-D projection mixing model.
pshock, **vpshock**	Constant temperature, plane-parallel shock plasma model.
pwab	Power-law distribution of neutral absorbers.
raymond, **vraymond**	Raymond-Smith thermal plasma.
rdblur	Convolve with the diskline model shape.
recorn	Change correction norm for a spectrum (replaces old recornrm command).
redden	IR/optical/UV extinction from Cardelli et al. (1989)
redge	Recombination edge.
reflect	reflection from neutral matter
refsch	E-folded power-law reflected from an ionized relativistic disk.
rfxconv	angle-dependent reflection from an ionized disk
rnei, **vrnei**, **vvrnei**	Non-equilibrium recombining collisional plasma.
sedov, **vsedov**	Sedov model with electron and ion temperatures.
sirf	Multi-blackbody self-irradiated funnel model.
simpl	Comptonization of a seed spectrum.
slimbh	Stationary slim accretion disk.
smaug	Model for an optically-thin, spherically-symmetric thermal plasma.
smedge	Smoothed absorption edge.
snapec	Galaxy cluster spectrum using SN yields.
spexpcut	Super-exponential cutoff absorption.
spline	Spline multiplicative factor.
srcut	Synchrotron radiation from cut-off electron distribution.
sresc	Synchrotron radiation from escape-limited electron distribution.
SSSice	Einstein Observatory SSS ice absorption.
step	Step function convolved with gaussian.
suzpsf	Suzaku PSF mixing model.
swind1	Absorption by partially ionized material with large velocity shear.
tapec, **vtapec**, **vvtapec**	APEC emission spectrum with separate continuum and line temperatures.
tbabs, **ztbabs**, **tbgrain**, **tbvarabs**	Absorption due to the ISM including molecules and grains.
uvred	UV reddening.
varabs, **zvarabs**	Photoelectric absorption with variable abundances.
vashift	Velocity shifts an additive model.
vmshift	Velocity shifts a multiplicative model.
voigt	A simple Voigt line profile.
wabs, **zwabs**	Photoelectric absorption (Morrison & McCammon).
wndabs, **zwndabs**	Photoelectric absorption with low energy window.
xion	The reflected spectrum from a photo-ionized accretion disk.
xilconv	angle-dependent reflection from an ionized disk

xmmpsf	XMM PSF model
xscat	dust scattering
zashift	Redshift an additive model.
zbabs	EUV ISM attenuation.
zdust	Extinction by dust grains (Pei, 1992).
zigm	UV/Optical attenuation by the intergalactic medium.
zmshift	Redshift a multiplicative model.
zredden	Redshifted IR/optical/UV extinction from Cardelli et al. (1989)
zsmdust	Extinction by dust grains in starburst galaxies.
zvfeabs	Redshifted absorption with variable iron abundance.
zxipcf	Partial covering absorption by partially ionized material.

6.2 Additive Model Components

This and the following sections contain information on specific, installed XSPEC models. The parameters are given as `par1`, `par2`, ..., and `norm`, which is the normalization. Additive models represent sources of emission.

6.2.1 agauss, zagauss: gaussian line profile in wavelength space

A simple gaussian line profile. If the width is ≤ 0 then it is treated as a delta function. The zagauss variant computes a redshifted gaussian.

$$A(\lambda) = K \frac{1}{\sigma\sqrt{2\pi}} \exp(-(\lambda - \lambda_l)^2/2\sigma^2)$$

where:

par1 = λ_l	line wavelength in Angstrom
par2 = σ	line width in Angstrom
Norm = K	total photons/cm^2/s in the line

For **zagauss** the corresponding formula is:

$$A(\lambda) = K \frac{(1+z)}{\sigma\sqrt{2\pi}} \exp(-(\lambda/(1+z) - \lambda_l)^2/2\sigma^2)$$

and the parameters are:

par1 = λ_l	line wavelength in Angstrom
par2 = σ	line width in Angstrom
par3 = z	redshift
Norm = K	total photons/cm^2/s in the line

6.2.2 apec, vapec, vvapec: APEC emission spectrum

An emission spectrum from collisionally-ionized diffuse gas calculated from the AtomDB atomic database. More information can be found at http://atomdb.org/ which should be consulted by anyone running this model. This default version number can be changed by modifying the ATOMDB_VERSION string in your `Xspec.init` file.

By default this model reads atomic physics continuum and line data from the files apec_v[version]_coco.fits and apec_v[version]_line.fits in the `$HEADAS/../spectral/modelData directory`. Different files can be specified by using the command **xset APECROOT**. There are several options. APECROOT can be set to a version number (eg 1.10, 1.2.0, 1.3.1, 2.0.1, 2.0.2, 3.0.2, 3.0.3). In this case the value of APECROOT will be used to replace the default version number in the name of the standard files and the resulting files will be assumed to be in the `modelData` directory. Alternatively, a filename root (eg `apec_v1.2.0`) can be given. This root will be used as a prefix for the `_coco.fits` and `_line.fits` files. Finally, if neither of these work then the model will assume that the APECROOT value gives the complete directory path, e.g.

```
XSPEC12> xset APECROOT /foo/bar/apec_v1.2.0
```

will use the input files

```
/foo/bar/apec_v1.2.0_coco.fits
/foo/bar/apec_v1.2.0_line.fits.
```

Thermal broadening of lines can be included by using: **xset APECTHERMAL yes**. This runs significantly slower than the option without thermal broadening so you should only use this when necessary. Velocity broadening of lines can be included by using: **xset APECVELOCITY** <`velocity`>, where <`velocity`> is sigma in km/s. This is added in Gaussian quadrature with any thermal broadening in use.

A continuum-only spectrum can be obtained by using **xset APECNOLINES yes**. This will turn off lines for all models using the apec files. To get a line-free version of a single apec model use **nlapec**.

The **apec** model uses abundances set by the **abund** command. The **vapec** and **vvapec** variants allow the user to set the abundance using additional parameters. For **apec** and **vapec** the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Note that this means that the **apec** and **vapec** models will show emission lines even if the abundance parameters are set to zero.

For the **apec** model the parameters are:

par1	plasma temperature, keV
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the abund command. The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par3	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vapec** variant the parameters are as follows.

par1	plasma temperature, keV
par2–par14	Abundances for He, C, N, O, Ne, Mg,Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command). The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par15	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vvapec** variant the parameters are as follows.

par1	plasma temperature, keV
par2–par31	Abundances for H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn wrt Solar (defined by the **abund** command)
Par32	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

6.2.3 atable: tabulated additive model

An additive table model component. The filename to be used must be given immediately after **atable** in the **model** command. For example:

```
XSPEC12> model atable{mymod.mod}
```

uses `mymod.mod` as the input for the model. For specifications of the table model file, see the OGIP memo 92-009 on the FITS file format for table model files (available on the WWW or by anonymous ftp from `ftp://legacy.gsfc.nasa.gov/caldb/docs/memos` . Example additive table model files are `mekal.mod` and `raysmith.mod` in `$HEADAS/../spectral/modelData` and `testpo.mod` in `$HEADAS/../spectral/session`.

Any number of tabulated model components (additive, multiplicative or exponential) may be used simultaneously.

The model value is assumed zero outside the tabulated energies unless the keywords LOELIMIT and HIELIMIT are given in the primary extension of the FITS file. In the latter case the value given in LOELIMIT will be used for energies below those tabulated and HIELIMIT for energies above those tabulated.

6.2.4 bapec, bvapec, bvvapec: velocity broadened APEC thermal plasma model

A velocity- and thermally-broadened emission spectrum from collisionally-ionized diffuse gas calculated from the AtomDB atomic database. More information can be found at `http://atomdb.org/` which should be consulted by anyone running this model. This default version number can be changed by modifiying the ATOMDB_VERSION string in your `Xspec.init` file.

By default this model reads atomic physics continuum and line data from the files apec_v[version]_coco.fits and apec_v[version]_line.fits in the `$HEADAS/../spectral/modelData directory`. Different files can be specified by using the command **xset APECROOT**. There are three options. APECROOT can be set to a version number (eg 1.10, 1.2.0, 1.3.1, 2.0.1, 3.0.2, 3.0.3). In this case the value of APECROOT will be used to replace the default version number in the name of the standard files and the resulting files will be assumed to be in the `modelData` directory. Alternatively, a filename root (eg `apec_v1.2.0`) can be given. This root will be used as a prefix for the `_coco.fits` and `_line.fits` files. Finally, if neither of these work then the model will assume that the APECROOT value gives the complete directory path, e.g.

```
XSPEC12> xset APECROOT /foo/bar/apec_v1.2.0
```

will use the input files

```
/foo/bar/apec_v1.2.0_coco.fits
/foo/bar/apec_v1.2.0_line.fits.
```

The **bapec** model uses abundances set by the **abund** command. The **bvapec** and **bvvapec** variants allow the user to set the abundance using additional parameters. For bapec and bvapec the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar.

For the **bapec** model the parameters are:

par1	plasma temperature, keV
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the abund command. The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par3	Redshift, z
par4	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **bvapec** variant the parameters are as follows.

par1	plasma temperature, keV
par2–par14	Abundances for He, C, N, O, Ne, Mg,Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command). The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par15	redshift, z
par16	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **bvvapec** variant the parameters are as follows.

par1	plasma temperature, keV
par2–par31	Abundances for H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn wrt Solar (defined by the **abund** command)
Par32	redshift, z
Par33	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

6.2.5 bbody, zbbody: blackbody

A blackbody spectrum.

$$A(E) = \frac{K \times 8.0525 E^2 dE}{(kT)^4[\exp(E/kT) - 1]}$$

where

par1= kT	temperature keV
norm= K	L_{39}/D_{10}^2, where L_{39} is the source luminosity in units of 10^{39} erg/s and D_{10} is the distance to the source in units of 10 kpc.

The **zbbody** variant allows an additional redshift parameter

$$A(E) = \frac{K \times 8.0525[E(1+z)]^2 dE}{(1+z)(kT)^4[\exp[E(1+z)/kT] - 1]}$$

where

par1= kT	temperature keV
z	redshift
norm= K	$L_{39}/[D_{10}(1+z)]^2$, where L_{39} is the source luminosity in units of 10^{39} erg/s and D_{10} is the distance to the source in units of 10 kpc.

6.2.6 bbodyrad: blackbody spectrum, area normalized

A blackbody spectrum with normalization proportional to the surface area.

$$A(E) = \frac{K \times 1.0344 \times 10^{-3} E^2 dE}{\exp(E/kT) - 1}$$

where

par1= kT	temperature keV
norm= K	R_{km}^2/D_{10}^2, where R_{km} is the source radius in km and D_{10} is the distance to the source in units of 10 kpc.

6.2.7 bexrav: reflected e-folded broken power law, neutral medium

A broken power-law spectrum multiplied by exponential high-energy cutoff, exp(-E/Ec), and reflected from neutral material. See Magdziarz & Zdziarski 1995, MNRAS, 273, 837 for details.

The output spectrum is the sum of an e-folded broken power law and the reflection component. The reflection component alone can be obtained for

$|rel_{refl}| < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to change then the limits of $|rel_{refl}|$ excluding zero (as then the direct component appears). If $E_c = 0$, there is no cutoff in the power law. The metal and iron abundance are variable with respect to those set by the command **abund**. The opacities are those set by the command **xsect**. As expected in AGNs, H and He are assumed to be fully ionized.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting BEXRAV_PRECISION eg **xset BEXRAV_PRECISION 0.05**. The default precision is 0.01 (ie 1%).

par1	Γ_1, first power law photon index
par2	E_{break}, break energy (keV)
par3	Γ_2, second power law photon index
par4	E_c, the e-folding energy in keV (if $E_c = 0$ there is no cutoff)
par5	rel_{refl}, reflection scaling factor (1 for isotropic source above disk)
par6	cosine of inclination angle
par7	abundance of elements heavier than He relative to the solar abundances
par8	iron abundance relative to the above
par9	z, redshift
norm	photon flux at 1 keV of the cutoff broken power-law only (no reflection) in the observed frame.

6.2.8 bexriv: reflected e-folded broken power law, ionized medium

A broken power-law spectrum multiplied by exponential high-energy cutoff, exp(-E/Ec), and reflected from ionized material. See Magdziarz & Zdziarski 1995, MNRAS, 273, 837 for details.

The output spectrum is the sum of an e-folded broken power law and the reflection component. The reflection component alone can be obtained for $|rel_{refl}| < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to change then the limits of $|rel_{refl}|$ excluding zero (as then the direct component appears). If $E_c = 0$, there is no cutoff in the power law. The metal and iron abundance are variable with respect to those set by the command **abund**.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision

is reached. The precision can be changed by setting BEXRIV_PRECISION eg **xset BEXRIV_PRECISION 0.05**. The default precision is 0.01 (ie 1%).

par1	Γ_1, first power law photon index
par2	E_{break}, break energy (keV)
par3	Γ_2, second power law photon index
par4	E_c, the e-folding energy in keV (if $E_c = 0$ there is no cutoff)
par5	rel_{refl}, reflection scaling factor (1 for isotropic source above disk)
par6	z, redshift
par7	abundance of elements heavier than He relative to the solar abundances
par8	iron abundance relative to the above
par9	cosine of inclination angle
par10	disk temperature (K)
par11	disk ionization parameter, $\xi = 4\pi F_{ion}/n$, where F_{ion} is the 5eV–20keV irradiating flux and n is the density of the reflector; see Done et al. 1992, ApJ 395, 275.
norm	photon flux at 1 keV of the cutoff broken power-law only (no reflection) in the observed frame.

6.2.9 bknpower: broken power law

A broken power law.

$$A(E) = \begin{cases} KE^{-\Gamma_1} & \text{if } E \leq E_{break} \\ KE_{break}^{\Gamma_2-\Gamma_1}(E/1keV)^{-\Gamma_2} & \text{if } E > E_{break} \end{cases}$$

where:

par1 = Γ_1	power law photon index for $E < E_{break}$
par2 = E_{break}	break point for the energy in keV
par3 = Γ_2	power law photon index for $E > E_{break}$
norm = K	photons/keV/cm^2/s at 1 keV

If POW_EMIN and POW_EMAX have been defined by the **xset** command then the norm becomes the flux in units of 10^{-12} ergs/cm^2/s over the energy range (POW_EMIN, POW_EMAX) keV unless POW_EMIN = POW_EMAX in which case the norm becomes the flux density in micro-Jansky at POW_EMIN keV. In these cases it is important that POW_EMIN and POW_EMAX lie within the energy range on which the model is being evaluated.

6.2.10 bkn2pow: broken power law, 2 break energies

A three-segment broken power law (i.e. with two break energies).

$$A(E) = \begin{cases} KE^{-\Gamma_1} & \text{if } E \leq E_{break,1} \\ KE_{break,1}^{\Gamma_2-\Gamma_1}(E/1keV)^{-\Gamma_2} & \text{if } E_{break,1} \leq E \leq E_{break,2} \\ KE_{break,1}^{\Gamma_2-\Gamma_1}E_{break,2}^{\Gamma_3-\Gamma_2}(E/1keV)^{-\Gamma_3} & \text{if } E_{break,2} \leq E \end{cases}$$

where:

par1 = Γ_1	power law photon index for $E \leq E_{break,1}$
par2 = $E_{break,1}$	first break point for the energy in keV
par3 = Γ_2	power law photon index for $E_{break,1} \leq E \leq E_{break,2}$
par4 = $E_{break,2}$	second break point for the energy in keV
par5 = Γ_3	power law photon index for $E > E_{break,2}$
norm = K	photons/keV/cm^2/s at 1 keV

If POW_EMIN and POW_EMAX have been defined by the **xset** command then the norm becomes the flux in units of 10^{-12} ergs/cm^2/s over the energy range (POW_EMIN, POW_EMAX) keV unless POW_EMIN = POW_EMAX in which case the norm becomes the flux density in micro-Jansky at POW_EMIN keV. In these cases it is important that POW_EMIN and POW_EMAX lie within the energy range on which the model is being evaluated.

6.2.11 bmc: Comptonization by relativistic matter

This is an analytic model describing Comptonization of soft photons by matter undergoing relativistic bulk-motion. The typical scenario involves thermal X-rays from the inner region of an accretion disk in a black-hole binary illuminating in-falling matter in close proximity to the black-hole event horizon.

For a detailed description of the model, refer to Titarchuk, Mastichiadis & Kylafis 1997, ApJ, 487, 834; Titarchuk & Zannias, 1998, ApJ, 493, 863; Laurent & Titarchuk 1999, ApJ, 511, 289; Zannias, Borozdin, Revnivtsev, Trudolyubov, Shrader, & Titarchuk, 1999, ApJ, 517, 367; or Shrader & Titarchuk 1999, ApJ 521, L21.

The model parameters are the characteristic black-body temperature of the soft photon source, a spectral (energy) index, and an illumination parameter characterizing the fractional illumination of the bulk-motion flow by the thermal photon source. It must be emphasized that this model is not an additive combination of power law and thermal sources, rather it represents a self-consistent

convolution. The bulk-motion up-scattering and Compton recoil combine to produce the hard spectral tail, which combined with the thermal source results in the canonical high-soft-state spectrum of black hole accretion. The position of the sharp high energy cutoff (due to recoil) can be determined using the theta function $\theta(E_c - E)$. The model can also be used for the general Comptonization case when the energy range is limited from above by the plasma temperature (see **compTT** and **compST**).

par1	Temperature of thermal photon source in keV.
par2	Energy spectral index alpha.
par3	Log of the A parameter. Note that f in Borozdin et al. 1999 and Shrader & Titarchuk 1999 is 10^{par3}
norm	A_N defined in Borozdin et al 1999 and Shrader & Titarchuk (1999)

6.2.12 bremss, vbremss, zbremss: thermal bremsstrahlung

A thermal bremsstrahlung spectrum based on the Kellogg, Baldwin & Koch (ApJ 199, 299) polynomial fits to the Karzas & Latter (ApJS 6, 167) numerical values. A routine from Kurucz (private communication) is used at the low temperature end. The He abundance is assumed to be 8.5 % of H by number.

The **zbremss** variant includes a choice of redshift and **vbremss** allows the H to He abundance ratio to be varied.

For **bremss**:

par1	plasma temperature in keV
norm	$\frac{3.02\times10^{-15}}{4\pi D^2}\int n_e n_I dV$, where D is the distance to the source (cm) and n_e, n_I are the electron and ion densities (cm^{-3})

For **zbremss**:

par1	plasma temperature in keV
par2	z, redshift
norm	$\frac{3.02\times10^{-15}}{4\pi D^2}\int n_e n_I dV$, where D is the distance to the source (cm) and n_e, n_I are the electron and ion densities (cm^{-3})

For **vbremss**:

par1	plasma temperature in keV
par2	n(He)/n(H) ((note that the Solar ratio is 0.085)
norm	$\frac{3.02\times10^{-15}}{4\pi D^2}\int n_e n_I dV$, where D is the distance to the source (cm) and n_e, n_I are the electron and ion densities (cm^{-3})

6.2.13 btapec, bvtapec, bvvtapec: velocity broadened APEC emission spectrum with separate continuum and line temperatures

A velocity- and thermally-broadened emission spectrum from collisionally-ionized diffuse gas calculated from the AtomDB atomic database. More information can be found at http://atomdb.org/ which should be consulted by anyone running this model. This version of the model allows different temperatures for the continuum and lines. This default version number can be changed by modifiying the ATOMDB_VERSION string in your `Xspec.init` file.

By default this model reads atomic physics continuum and line data from the files apec_v[version]_coco.fits and apec_v[version]_line.fits in the `$HEADAS/../spectral/modelData directory`. Different files can be specified by using the command **xset APECROOT**. There are three options. APECROOT can be set to a version number (eg 1.10, 1.2.0, 1.3.1, 2.0.1, 3.0.2, 3.0.3). In this case the value of APECROOT will be used to replace the default version number in the name of the standard files and the resulting files will be assumed to be in the `modelData` directory. Alternatively, a filename root (eg `apec_v1.2.0`) can be given. This root will be used as a prefix for the `_coco.fits` and `_line.fits` files. Finally, if neither of these work then the model will assume that the APECROOT value gives the complete directory path, e.g.

```
XSPEC12> xset APECROOT /foo/bar/apec_v1.2.0
```

will use the input files

```
/foo/bar/apec_v1.2.0_coco.fits
/foo/bar/apec_v1.2.0_line.fits.
```

The **btapec** model uses abundances set by the **abund** command. The **bvtapec** and **bvvtapec** variants allow the user to set the abundance using additional parameters. For btapec and bvtapec the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar.

For the **btapec** model the parameters are:

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the abund command. The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par4	Redshift, z
par5	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **bvtapec** variant the parameters are as follows.

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3–par15	Abundances for He, C, N, O, Ne, Mg,Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command). The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par16	redshift, z
par17	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **bvvtapec** variant the parameters are as follows.

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3–par32	Abundances for H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn wrt Solar (defined by the **abund** command)
Par33	redshift, z
Par34	Gaussian sigma for velocity broadening (km/s)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

6.2.14 c6mekl, c6vmekl, c6pmekl, c6pvmkl: differential emission measure using Chebyshev representations with multi-temperature mekal

c6mekl is a multi-temperature **mekal** or **apec** model using sixth-order Chebyshev polynomial for the differential emission measure. The DEM is not constrained to be positive.. The `switch` parameter determines whether spectrum is calculated by running the mekal code, by interpolating on a pre-calculated mekal table, or using the apec model. The final option is now preferred. The reference for this model is Singh et al. (1996, ApJ, 456, 766).

c6pmekl differs by using the exponential of the 6th order Chebyshev polynomial

c6mekl and **c6pmekl** use abundances relative to the Solar abundances set by the **abund** command.

The variants **c6vmekl** and **c6pvmkl** with polynomial and exponential polynomial respectively allow the user to specify 14 elemental abundance.

For **c6mekl** and **c6pmekl** the parameters are:

par1–6	Chebyshev polynomial coefficients
par7	H density (cm^{-3})
par8	abundance wrt to Solar
par9	Redshift
par10	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using APEC model).
norm	Normalization

While for **c6vmekl** and **c6pvmkl** the parameters are:

par1–6	Chebyshev polynomial coefficients
par7	H density (cm^{-3})
par8–21	Abundances of He, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par22	Redshift
par23	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using APEC model).
norm	Normalization

6.2.15 carbatm: Nonmagnetic carbon atmosphere of a neutron star

The model provides the spectra emitted from a nonmagnetic carbon atmosphere of a neutron star. The model spectra in a 0 - 20 keV range of (unredshifted) photon energy are computed on a grid of surface gravity accelerations log(g) = 13.7 - 14.9 (in cgs units) and effective temperatures T = 1 - 4 MK. For a given set of the fitting parameters, the surface gravity g and the gravitational redshift z are derived from the mass M and radius R of the star. The trial spectra are computed using a linear interpolation between the nearest model spectra on the T-log(g) grid, and the boundaries of the energy bins. The number of photons in each bin are divided by (1+z). This is an updated version of the models presented in Suleimanov et al. (2014). The details can be found in Suleimanov et al. (2016; A&A, submitted).

The directory used for the files required by this model can be changed by using the **xset** command to set CARBATM.

The parameters are:

par1	Effective temperature T (MK)
par2	Neutron star mass M (M_{sol})
par3	Neutron star radius R (km)
par4	Normalization K = $A/(d_{10}kpc)^2$ where d_{10}kpc is the distance to the source in units of 10 kpc and A characterizes the fraction of the surface emitting the radiation (A=1 corresponds to the case when the radiation is emitted from the entire surface).

Please send any questions and comments about this model to suleimanov@astro.uni-tuebingen.de or klochkov@astro.uni-tuebingen.de.

6.2.16 cemekl, cevmkl: plasma emission, multi-temperature using mekal

A multi-temperature plasma emission model. Emission measures follow a power-law in temperature ($dEM = (T/T_{max})^{\alpha-1}dT/T_{max}$).The `switch` parameter determines whether spectrum is calculated by running the mekal code, by interpolating on a pre-calculated mekal table, or using the apec model. The final option is now preferred.

For the **cemekl** version, the abundance ratios are set by the **abund** command. The **cevmkl** variant allows the user to define the abundances.

For **cemekl** the parameters are:

par1	α, index for power-law emissivity function
par2	T_{max}, maximum temperature
par3	n_H (cm^{-3}
par4	Abundance relative to Solar
par5	z, redshift
par6	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using APEC model).
norm	Normalization

For the **cevmkl** variant the parameters are:

par1	α, index for power-law emissivity function
par2	T_{max}, maximum temperature
par3	n_H (cm^{-3}
par4–17	Abundances of He, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par18	z, redshift
par19	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using APEC model).
norm	Normalization

6.2.17 cflow: cooling flow

A cooling flow model after Mushotzky & Szymkowiak (*Cooling Flows in Clusters and Galaxies*, ed. Fabian, 1988). An index of zero for the power-law emissivity function corresponds to emission measure weighted by the inverse of the bolometric luminosity at that temperature. The abundance ratios are set by the **abund** command. Redshift is converted to distance using the cosmology set by the **cosmo** command.

par1	index for power-law emissivity function
par2	low temperature (keV)
par3	high temperature (keV)
par4	abundance relative Solar
par5	redshift, z
norm	Mass accretion rate (solar mass/yr)

6.2.18 compbb: Comptonization, black body

Comptonized blackbody model by Nishimura, Mitsuda and Itoh, 1986, PASJ, 38, 819. The electron temperature should normally be kept fixed since the Compton y parameter is the product of the electron temperature and optical depth.

par1	blackbody temperature (keV)
par2	electron temperature of the hot plasma (keV)
par3	optical depth of the plasma
norm	$(L_{39}/D_{10})^2$, where L_{39} is the source luminosity in units of 10^{39} erg/s and D_{10} is the distance to the source in units of 10 kpc (the same definition used for the **bbodyrad** model)

6.2.19 compLS: Comptonization, Lamb & Sanford

A Comptonization spectrum after Lamb and Sanford, 1979, MNRAS, 288, 555. This model calculates the self-Comptonization of a bremsstrahlung emission from an optically thick spherical plasma cloud with a given optical depth and temperature. It was popular for Sco X-1.

par1	temperature in keV
par2	optical depth
norm	normalization

6.2.20 compmag: Thermal and bulk Comptonization for cylindrical accretion onto the polar cap of a magnetized neutron star

This model describes the spectral formation in the accretion column onto the polar cap of a magnetized neutron star, with both thermal and bulk Comptonization processes taken into account. The details for the method adopted for the numerical solution of the radiative transfer equation are reported in Farinelli et al. (2012, A&A, 538, A67).

This model can be used for spectral fitting of both accreting X-ray pulsars and Supergiant Fast X-ray Trasients.

par1	kT_{bb}, temperature of the seed blackbody spectrum (keV).
par2	kT_e, electron temperature of the accretion column (keV).
par3	τ, vertical optical depth of the accretion column, with electron cross-section equal to 10^{-3} of the Thomson cross-section.
par4	η, index of the velocity profile when the accretion velocity increases towards the neutron star (valid when par8=1).
par5	β_0, terminal velocity of the accreting matter at the neutron star surface (valid when par8=1).
par6	r_0, radius of the accretion column in units of the neutron star Schwarzschild radius.
par7	A, albedo at the neutron star surface.
par8	Flag for setting the velocity profile of the accretion column ($1 \Rightarrow \beta(z) = A(Z_s/Z)^{-\eta}$, where $A = \beta_0(Z_0/Z_s)^{\beta}$, $2 \Rightarrow \beta(\tau) = -\alpha\tau$)
norm	(R_{km}^2/D_{10}^2, where R_{km} and D_{10} are the accretion column radius in km and the source distance in units of 10 kpc, respectively.

6.2.21 compPS: Comptonization, Poutanen & Svenson

Comptonization spectra computed for different geometries using exact numerical solution of the radiative transfer equation. The computational "iterative scattering method" is similar to the standard Lambda-iteration and is described in Poutanen J., Svensson R., 1996, ApJ, 470, 249 (PS96). The Compton scattering kernel is the exact one as derived by Jones F. C., 1968, Phys. Rev., 167, 1159 (see PS96 for references).

Comptonization spectra depend on the geometry (slab, sphere, hemisphere, cylinder), Thomson optical depth tau, parameters of the electron distribution, spectral distribution of soft seed photons, the way seed soft photons are injected to the electron cloud, and the inclination angle of the observer.

The resulting spectrum is reflected from the cool medium according to the computational method of Magdziarz & Zdziarski (1995) (see **reflect**, **pexrav**, **pexriv** models). rel_{refl} is the solid angle of the cold material visible from the Comptonizing source (in units of 2π), other parameters determine the abundances and ionization state of reflecting material (Fe_ab_re, Me_ab, xi, Tdisk). The reflected spectrum is smeared out by rotation of the disk due to special and general relativistic effects using "diskline"-type kernel (with parameters Betor10, Rin, Rout).

Electron distribution function can be Maxwellian, power-law, cutoff Maxwellian, or hybrid (with low temperature Maxwellian plus a power-law tail).

Possible geometries include plane-parallel slab, cylinder (described by the height-

to-radius ratio H/R), sphere, or hemisphere. By default the lower boundary of the "cloud" (not for spherical geometry) is fully absorbive (e.g. cold disk). However, by varying covering factor parameter cov_fac, it may be made transparent for radiation. In that case, photons from the "upper" cloud can also be upscattered in the "lower" cloud below the disk. This geometry is that for an accretion disk with cold cloudlets in the central plane (Zdziarski, Poutanen, et al. 1998, MNRAS, 301, 435). For cylinder and hemisphere geometries, an approximate solution is obtained by averaging specific intensities over horizontal layers (see PS96). For slab and sphere geometries, no approximation is made.

The seed photons can be injected to the electron cloud either isotropically and homogeneously through out the cloud, or at the bottom of the slab, cylinder, hemisphere or center of the sphere (or from the central plane of the slab if cov_frac is not 1). For the sphere, there exist a possibility (IGEOM=-5) for photon injection according to the eigenfunction of the diffusion equation $\sin(\pi * \tau'/\tau)/(\pi * \tau'/\tau)$, where τ' is the optical depth measured from the center (see Sunyaev & Titarchuk 1980).

Seed photons can be black body (**bbodyrad**) for Tbb positive or multicolor disk (**diskbb**) for Tbb negative. The normalization of the model also follows those models: (1) Tbb positive, K = (RKM)**2 /(D10)**2, where D10 is the distance in units of 10 kpc and RKM is the source radius in km; (2) Tbb negative, K = (RKM)**2 /(D10)**2 cos(theta), where theta is the inclination angle.

Thomson optical depth of the cloud is not always good parameter to fit. Instead the Compton parameter y=4 * tau * Theta (where Theta= Te (keV) / 511) can be used. Parameter y is directly related to the spectral index and therefore is much more stable in fitting procedure. The fitting can be done taking 6th parameter negative, and optical depth then can be obtained via tau= y/(4* Te / 511).

The region of parameter space where the numerical method produces reasonable results is constrained as follows : 1) Electron temperature Te >10 keV; 2) Thomson optical depth tau <1.5 for slab geometry and tau <3, for other geometries.

In versions 4.0 and above the Compton reflection is done by a call to the **ireflect** model code and the relativistic blurring by a call to **rdblur**. This does introduce some changes in the spectrum from earlier versions. For the case of a neutral reflector (i.e. the ionization parameter is zero) more accurate opacities are calculated. For the case of an ionized reflector the old version assumed that for the purposes of calculating opacities the input spectrum was a power-law (with index based on the 2–10 keV spectrum). The new version uses the actual input spectrum, which is usually not a power law, giving different opacities for a given ionization parameter and disk temperature. The Greens' function integration required for the Compton reflection calculation is performed to an accuracy of

0.01 (i.e. 1%). This can be changed using e.g. **xset COMPPS_PRECISION 0.05**.

The model parameters are as follows :

par1	Te, electron temperature in keV
par2	p, electron power-law index [N(gamma)=gamma^{-p}]
par3	gmin, minimum Lorentz factor gamma
par4	gmax, maximum Lorentz factor gamma

- if any of gmin or gmax <1 then Maxwellian electron distribution with parameter Te
- if Te=0 then power-law electrons with parameters p, gmin, gmax
- if both gmin,gmax>=1 but gmax<gmin then cutoff Maxwellian with Te, p, gmin (cutoff Lorentz factor) as parameters
- if Te is non-zero, gmin, gmax >= 1 then hybrid electron distribution with parameters Te, p, gmin, gmax

par5	Tbb, temperature of soft photons. If Tbb is positive then blackbody, if Tbb negative then multicolor disk with inner disk temperature —Tbb—
par6	if >0 : tau, vertical optical depth of the corona; if <0 : y = 4*Theta*tau. limits: for the slab geometry - tau <1, if say tau∼2 increase MAXTAU to 50, for sphere - tau <3
par7	geom, 0 - approximate treatment of radiative transfer using escape probability for a sphere (very fast method); 1 - slab; 2 - cylinder; 3 - hemisphere; 4,5 - sphere input photons at the bottom of the slab, cylinder, hemisphere or center of the sphere (or from the central plane of the slab if cov_fact not 1). if <0 then geometry defined by —geom— and sources of incident photons are isotropic and homogeneous. -5 - sphere with the source of photons distributed according to the eigenfunction of the diffusion equation f(tau')=sin(pi*tau'/tau)/(pi*tau'/tau) where tau' varies between 0 and tau.
par8	H/R for cylinder geometry only
par9	cosIncl, cosine of inclination angle (if <0 then only black body)
par10	cov_fac, covering factor of cold clouds. if geom =+/- 4,5 then cov_fac is dummy
par11	R, amount of reflection Omega/(2*pi) (if R <0 then only reflection component)
par12	FeAb, iron abundance in units of solar
par13	MeAb, abundance of heavy elements in units of solar
par14	xi, disk ionization parameter L/(nR^2)
par15	temp, disk temperature for reflection in K
par16	beta, reflection emissivity law (r^{beta}, if beta=-10 then non-rotating disk, if beta=10 then 1.-sqrt(6./rg))/rg**3
par17	Rin/Rg, inner radius of the disk (Schwarzschild units)
par18	Rout/Rg, outer radius of the disk
par19	redshift

6.2.22 compST: Comptonization, Sunyaev & Titarchuk

A Comptonization spectrum after Sunyaev and Titarchuk 1980, A&A, 86, 121. This model is the Comptonization of cool photons on hot electrons.

par1	temperature in keV
par2	optical depth
norm	$Nf/4\pi d^2$, where N is the total number of photons from the source, d is the distance to the source, f is the factor $f = z(z+3)y^z/\Gamma(2z+4)/\Gamma(z)$, z is the spectral index, y is the injected photon energy in units of the temperature, and Γ is the incomplete gamma function.

6.2.23 comptb: Thermal and bulk Comptonization of a seed blackbody-like spectrum

This model describes the Comptonization spectrum of soft photons off electrons which are either purely thermal or additionally subjected to an inward bulk motion. It consists of two components: one is the direct seed photon spectrum and the other one is the Comptonized spectrum. The latter is obtained as a self-consistent convolution of the seed photon spectrum with the system Green's function.

The model is not specific to bulk Comptonization but it includes in a coherent way different spectral shapes such as simple blackbody (i.e. neither thermal nor bulk Comptonization), thermal Comptonization (equivalent to **compTT**) and thermal plus bulk Comptonization. In the latter case, it can be considered a completion and update of the **bmc** model, as it includes the cut-off term in the spectrum.

All mathematical details of the model and its validity limits for applications are reported in Farinelli et al. (2008, ApJ, 680, 602).

par1	kT_S, temperature of the seed photons (keV).
par2	gamma, index of the seed photon spectrum (default gamma=3).
par3	alpha, energy index of the Comptonization spectrum.
par4	delta, bulk parameter, efficiency of bulk over thermal Comptonization.
par5	kT_e, temperature of the electrons (keV).
par6	log(A), log of the illuminating factor parameter (A).
norm	Normalization of the seed photon spectrum, defined in the same way as the **bbody** model.

6.2.24 compTT: Comptonization, Titarchuk

This is an analytic model describing Comptonization of soft photons in a hot plasma, developed by L. Titarchuk (see ApJ, 434, 313). This replaces the Sunyaev-Titarchuk Comptonization model in the sense that the theory is extended to include relativistic effects. Also, the approximations used in the model work well for both the optically thin and thick regimes. The Comptonized spectrum is determined completely by the plasma temperature and the so-called β parameter which is independent of geometry. The optical depth is then determined as a function of β for a given geometry. Thus `par5` switches between spherical and disk geometries so that β is not a direct input here. This parameter MUST be frozen. If `par5` ≥ 0, β is obtained from the optical depth using analytic approximation (e.g. Titarchuk 1994). If par5 <0 and $0.1 < \tau < 10$, β is obtained by interpolation from a set of accurately calculated pairs of β and τ from Sunyaev & Titarchuk 1985 (A&A 143, 374).

In this incarnation of the model, the soft photon input spectrum is a Wien law [x^2e^{-x} photons] because this lends itself to a particularly simple analytical form of the model. For present X-ray detectors this should be adequate. Note that in energy flux space the peak of the Wien law occurs at 3kT as opposed to 2.8kT for a blackbody. The plasma temperature may range from 2–500 keV, but the model is not valid for simultaneously low temperatures and low optical depth, or for high temperatures and high optical depth. The user is strongly urged to read the following references (esp. HT95 Fig 7) before and after using this model in order to fully understand and appreciate the physical assumptions made: Titarchuk, L., 1994, ApJ, 434, 313; Hua, X-M., Titarchuk, L., 1995, ApJ, 449, 188; Titarchuk, L., Lyubarskij, Y., 1995, ApJ, 450, 876.

par1	Redshift
par2	Input soft photon (Wien) temperature (keV)
par3	Plasma temperature (keV)
par4	Plasma optical depth
par5	Geometry switch. (sign denotes approximation technique, magnitude determines geometry). $\leq$1 disk, >1 sphere, 0 use analytic approximation for β vs. τ, <0 β vs. τ from interpolation.
norm	normalization

6.2.25 cplinear: a non-physical piecewise-linear model for low count background spectra.

This is a simple non-physical model for low-count background spectra, used by fitting scripts in the ACIS Extract (AE) package. Using this model outside the context of the AE package should be done with extreme caution since it requires

a choice on vertex energies and number of segments. AE places the first and last vertices at the lowest and highest energy of the background counts. Intermediate vertices are placed at energies where a background count exists such that each segment covers a similar number of background counts. Any results using this model should cite Broos et al. (2010).

Note that if all the rate parameters in use are thawed then the norm parameter is degenerate and must be frozen. Freezing one of the rate parameters will not work because if that vertex is driven to zero in the fit then the norm will be zero and the other rate parameters infinite.

par1	energy00 (keV)
par2	energy01 (keV)
par3	energy02 (keV)
par4	energy03 (keV)
par5	energy04 (keV)
par6	energy05 (keV)
par7	energy06 (keV)
par8	energy07 (keV)
par9	energy08 (keV)
par10	energy09 (keV)
par11	log_rate00
par12	log_rate01
par13	log_rate02
par14	log_rate03
par15	log_rate04
par16	log_rate05
par17	log_rate06
par18	log_rate07
par19	log_rate08
par10	log_rate09
norm	

6.2.26 cutoffpl: power law, high energy exponential cutoff

A power law with high energy exponential rolloff.

$$A(E) = KE^{-\alpha} \exp(-E/\beta)$$

par1	α, power law photon index
par2	β, e-folding energy of exponential rolloff (in keV)
norm	K in photons/keV/cm^2/s at 1 keV

If POW_EMIN and POW_EMAX have been defined by the **xset** command then

the norm becomes the flux in units of 10^{-12} ergs/cm^2/s over the energy range (POW_EMIN, POW_EMAX) keV unless POW_EMIN = POW_EMAX in which case the norm becomes the flux density in micro-Jansky at POW_EMIN keV. In these cases it is important that POW_EMIN and POW_EMAX lie within the energy range on which the model is being evaluated.

6.2.27 disk: accretion disk, black body

The spectrum from an accretion disk, where the opacities are dominated by free-free absorption, i.e., the so-called blackbody disk model. Not correct for a disk around a neutron star.

par1	accretion rate in Eddington Luminosities
par2	central mass in solar mass units
par3	inner disk radius in gravitational (= 3 Schwarzschild radii)
norm	$2\cos(i)/d^2$ where i is the inclination of the disk and d is the distance in units of 10 kpc

6.2.28 diskbb: accretion disk, multi-black body components

The spectrum from an accretion disk consisting of multiple blackbody components. For example, see Mitsuda et al., PASJ, 36, 741, (1984), Makishima et al., ApJ 308, 635, (1986).

par1	temperature at inner disk radius (keV)
norm	$(R_{in}/D_{10})^2 \cos\theta$, where where R_{in} is "an apparent" inner disk radius in km, D_{10} the distance to the source in units of 10 kpc, and θ the angle of the disk ($\theta = 0$ is face-on). On the correction factor between the apparent inner disk radius and the realistic radius, see e.g., Kubota et al. 1998, PASJ, 50, 667.

6.2.29 diskir: Irradiated inner and outer disk

The inner disk can be irradiated by the Compton tail. This can substantially change the inner disk temperature structure from that expected from an unilluminated disk in the limit where the ratio of luminosity in the tail to that in the disk, $L_c/L_d >> 1$. This is generally the case in the low/hard state of accreting black holes, and neglecting this effect leads to an underestimate of the inner

disk radius (Gierlinski, Done & Page 2008a MNRAS, 388, 753).

The irradiated inner disk and Compton tail can illuminate the rest of the disk, and a fraction f_{out} of the bolometric flux is thermalized to the local blackbody temperature at each radius. This reprocessed flux generally dominates the optical and UV bandpass of LMXBs (Gierlinski, Done & Page 2009 MNRAS, 392, 1106).

par1	kT_disk, innermost temperature of the UNILLUMINATED disk
par2	Gamma, asymptotic power-law photon index
par3	kT_e, electon temperature (high energy rollover)
par4	Lc/Ld, ratio of luminosity in the Compton tail to that of the UNILLUMINATED disk
par5	fin, fraction of luminosity in the Compton tail which is thermalized in the inner disk (generally fix at 0.1 as appropriate for an albedo of 0.3 and solid angle of 0.3)
par6	rirr, radius of the Compton illuminated disk in terms of the inner disk radius
par7	fout, fraction of bolometric flux which is thermalized in the outer disk
par8	logrout, log10 of the outer disk radius in terms of the inner disk radius K = normalization, as in **diskbb**

6.2.30 diskline: accretion disk line emission, relativistic

A line emission from a relativistic accretion disk. See Fabian et al., MNRAS 238, 729. Setting par2 to 10 is the special case of the accretion disk emissivity law $(1 - \sqrt{6/R})/R^3$.

par1	line energy
par2	power law dependence of emissivity. If this parameter is 10 or greater then the accretion disk emissivity law $(1 - \sqrt{6/R})/R^3$ is used. Otherwise the emissivity scales as R^{par2}
par3	inner radius (units of GM/c^2)
par4	outer radius (units of GM/c^2)
par5	inclination (degrees)
norm	photon/cm^2/s in the spectrum

6.2.31 diskm: accretion disk with gas pressure viscosity

A disk model with gas pressure viscosity. The spectrum from an accretion disk where the viscosity scales as the gas pressure. From Stella and Rosner 1984, ApJ, 277, 312.

par1	accretion rate in Eddington Luminosities
par2	central mass in solar mass units
par3	inner disk radius in gravitational (= 3 Schwarzschild radii)
par4	viscosity
norm	$2\cos(i)/d^2$ where i is the inclination of the disk and d is the distance in units of 10 kpc

6.2.32 disko: accretion disk, inner, radiation pressure viscosity

A modified blackbody disk model. The spectrum from the inner region of an accretion disk where the viscosity is dominated by radiation pressure.

par1	accretion rate in Eddington Luminosities
par2	central mass in solar mass units
par3	inner disk radius in gravitational (= 3 Schwarzschild radii)
par4	viscosity
norm	$2\cos(i)/d^2$ where i is the inclination of the disk and d is the distance in units of 10 kpc

6.2.33 diskpbb: accretion disk, power-law dependence for T(r)

A multiple blackbody disk model where local disk temperature T(r) is proportional to r^{-p}, where p is a free parameter. The standard disk model, diskbb, is recovered if $p = 0.75$. If radial advection is important then $p < 0.75$. See the discussion and examples in, e.g., Mineshige et al. 1994, ApJ, 426, 308; Hirano et al. 1995, ApJ, 446, 350; Watarai et al. 2000, PASJ, 52, 133; Kubota and Makishima 2004, ApJ, 601, 428; Kubota et al. 2005, ApJ, 631, 1062.

par1	Tin, temperature at inner disk radius (keV)
par2	p, exponent of the radial dependence of the disk temperature
norm	$(R_{in}/D_{10})^2 \cos\theta$, where where R_{in} is "an apparent" inner disk radius in km, D_{10} the distance to the source in units of 10 kpc, and θ the angle of the disk ($\theta = 0$ is face-on). On the correction factor between the apparent inner disk radius and the realistic radius, see e.g., Kubota et al. 1998, PASJ, 50, 667.

6.2.34 diskpn: accretion disk, black hole, black body

Blackbody spectrum of an accretion disk. This is an extension of the **diskbb** model, including corrections for temperature distribution near the black hole. The temperature distribution was calculated in Paczynski-Wiita pseudo-Newtonian potential. An accretion rate can be computed from the maximum temperature found. For details see Gierlinski et al., 1999, MNRAS, 309, 496. Please note that the inner disk radius (`par2`) can be a free parameter only close to `par2` = 6; otherwise `par2` is strongly correlated with `K`.

par1	maximum temperature in the disk (keV)
par2	inner disk radius in Rg = GM/c^2 units, $6 \leq$ `par2` ≤ 1000
norm	$M^2 \cos(i)/(D^2 * \beta^4)$, where M = central mass (solar masses), D distance to the source (kpc), i inclination of the disk, and b color/effective temperature ratio.

6.2.35 eplogpar: log-parabolic blazar model with νF_ν normalization

eplogpar is a power-law with an index which varies with energy as a log parabola.

$$A(E) = 10^{-\beta(\log(E/E_p))^2}/E^2$$

par1	E_p, the peak energy in νF_ν
par2	β, curvature term
norm	K, the flux in νF_ν units at energy E_p keV

See for instance Tramacere et al., 2007 and Tramacere et al., 2009.

6.2.36 Eqpair, eqtherm, compth: Paolo Coppi's hybrid (thermal/non-thermal) hot plasma emission models.

These models are based on Paolo Coppi's hybrid thermal/non-thermal hot plasma emission model for X-ray binaries. The underlying physics and a detailed description of the code are included in the draft paper http://www.astro.yale.edu/coppi/eqpair/eqpap4.ps.

Do not use these models without reading and understanding this paper. Simplified models **eqtherm** and **compth** are provided for cases where non-thermal processes are not important and photon-photon pair production can be ignored. These should only be used if $l_{bb} < 10$.

The temperature of the thermal component of the electron distribution and the total electron optical depth (for both ionization electrons and electron-positron pairs) are written out if the chatter level is set to 15. This information is important for checking self-consistency.

In versions 1.10 and above the Compton reflection is done by a call to the **ireflect** model code and the relativistic blurring by a call to **rdblur**. This does introduce some changes in the spectrum from earlier versions. For the case of a neutral reflector (i.e. the ionization parameter is zero) more accurate opacities are calculated. For the case of an ionized reflector the old version assumed that for the purposes of calculating opacities the input spectrum was a power-law (with index based on the 2–10 keV spectrum). The new version uses the actual input spectrum, which is usually not a power law, giving different opacities for a given ionization parameter and disk temperature. The Greens' function integration required for the Compton reflection calculation is performed to an accuracy of 0.01 (i.e. 1%). This can be changed using e.g. **xset** **EQPAIR_PRECISION 0.05**.

The parameters for all three models are:

par1	lh/ls, ratio of the hard to soft compactness
par2	lbb, the soft photon compactness
par3	kTbb, if >0 then temperature of a blackbody is the Tmax parameter for the **diskpn** mod
par4	lnt/lh, fraction of power supplied to energet into accelerating non-thermal particles
par5	tp, the Thomson scattering depth
par6	radius, the size of the scattering region (cm
par7	gmin, minimum Lorentz factor of the pairs
par8	gmax, maximum Lorentz factor of the pairs
par9	Ginj, if <0 then non-thermal spectrum is as at gmax; if >0 then a power-law from gmin
par10	pairinj, if = 0 then accelerated particles are pool; if = 1 then accelerated particles are e
par11	cosIncl, inclination of reflecting material wr
par12	Refl, fraction of scattering region's emission ing material
par13	Fe_abund, relative abundance of iron
par14	Ab>He, relative abundance of other metals
par15	Tdisk, temperature of reflecting disk
par16	x, ionization parameter of reflector
par17	b, power-law index with radius of disk refle
par18	Rin, inner radius of reflecting material (GM
par19	Rout, outer radius of reflecting material (G
par20	Redshift, z
norm	

6.2.37 equil, vequil: collisional plasma, ionization equilibrium

Ionization equilibrium collisional plasma model. This is the equilibrium version of Kazik Borkowski's NEI models. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The

vequil variant allows the user to set the abundances for the model. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0.

For the equil model the parameters are:

par1	plasma temperature (keV)
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par3	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vequil** model, the parameters are:

par1	plasma temperature (keV)
par2–par13	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par14	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

The references for this model are as follows:
Borkowski, Lyerly & Reynolds, 2001, ApJ, 548, 820
Hamilton, A.J.S., Sarazin, C. L. & Chevalier, R. A. , 1983,ApJS, 51,115
Borkowski, K.J., Sarazin, C.L. & Blondin, J.M. 1994, ApJ, 429, 710
Liedahl, D.A., Osterheld, A.L. Goldstein, W.H. 1995, ApJ, 438, L11

6.2.38 expdec: exponential decay

An exponential decay.

$$A(E) = K \exp(-\alpha E)$$

where:

par1	α, exponential factor
norm	normalization

6.2.39 ezdiskbb: multiple blackbody disk model with zero-torque inner boundary

A multi-temperature blackbody model for a thin, steady-state, Newtonian accretion disk, assuming zero torque at the inner boundary for the disk at radius R_{in}. The temperature of the disk as a function of radius is assumed to be $T(r) = T_* r^{-3/4}(1 - r^{-1/2})^{1/4}$, where $r = R/R_{in}$ and $T_* = f(3GM\dot{M}/8\pi R_{in}^3\sigma)^{1/4}$. The maximum temperature in the disk is given by $T_{max} = 0.488T_*$.

This model is an alternative to **diskbb**, which assumes a non-zero torque at the inner edge and a temperature profile $T(r) = T_* r^{-3/4}$, and it should be used to fit spectra of disks when the zero-torque inner boundary condition is appropriate. For details see Zimmerman et al. (2004) astro-ph/0408209.

par1	maximum temperature in the disk (keV)
par2	$(1/f^4)(R_{in}/D)^2 cosi$, where R_{in} is the inner radius of the disk in km, D is the distance to the source in units of 10 kpc, i is the inclination, and f is the color to effective temperature ratio.
norm	normalization

6.2.40 gadem, vgadem: plasma emission, multi-temperature with gaussian distribution of emission measure.

A multi-temperature plasma emission model built on top of the **apec** or **mekal** codes. The emission measure distribution is a gaussian with mean and sigma given by the first two model parameters. The `switch` parameter determines whether the apec or mekal codes will be used. For the mekal code there are also the options to run the code for each temperature or interpolate from a pre-calculated table. The former is slower but more accurate. See the documentation on the **apec** model for additional information on using different AtomDB versions or applying thermal or velocity broadening.

For the **gadem** version, the abundance ratios are set by the **abund** command. The **vgadem** variant allows the user to define abundances for the more common elements. See the documentation on the **apec** model for information on using additional elements included in AtomDB.

The parameters for **gadem** are:

par1	mean temperature for gaussian emission measure distribution
par2	sigma temperature for gaussian emission measure distribution
par3	nH (cm^{-3})
par4	abundance relative to Solar
par5	redshift z
par6	switch (0 = calculate using MEKAL model; 1 = interpolate using MEKAL model; 2 = interpolate using APEC model)
norm	Normalization

For the **vgadem** variant the parameters are:

par1	mean temperature for gaussian emission measure distribution
par2	sigma temperature for gaussian emission measure distribution
par3	nH (cm^{-3})
par4–17	abundance relative to solar Abundances for He, C, N, O, Ne, Na, Mg, Al, Si, S,Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par18	redshift z
par19	switch (0 = calculate using MEKAL model; 1 = interpolate using MEKAL model; 2 = interpolate using APEC model)
norm	Normalization

6.2.41 gauss, zgauss: gaussian line profile

A simple gaussian line profile. If the width is ≤ 0 then it is treated as a delta function. The **zgauss** variant computes a redshifted gaussian.

$$A(E) = K \frac{1}{\sigma * \sqrt{2 * \pi}} \exp\left(\frac{-(E - E_l)^2}{2\sigma^2}\right)$$

where:

par1	E_l, line energy in keV
par2	σ, line width in keV
Norm	K, total photons/cm^{-2}/s in the line

For **zgauss** the corresponding formula is:

$$A(E) = K \frac{1}{(1+z)\sigma * \sqrt{2*\pi}} \exp\left(\frac{-(E(1+z)-E_l)^2}{2\sigma^2}\right)$$

and parameter settings are:

par1	E_l, source frame line energy in keV
par2	σ, source frame line width in keV
par3	z, redshift
Norm	K, source frame total photons/cm^{-2}/s in the line

6.2.42 gnei, vgnei, vvgnei: collisional plasma, non-equilibrium, temperature evolution

Non-equilibrium ionization collisional plasma model. This is a generalization of the **nei** model where the temperature is allowed to have been different in the past i.e. the ionization timescale averaged temperature is not necessarily equal to the current temperature. For example, in a standard Sedov model with equal electron and ion temperatures, the ionization timescale averaged temperature is always higher than the current temperature for each fluid element. The references for this model can be found under the description of the **equil** model. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The **vgnei** variant allows the user to set the abundances for the more common elements. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full control over all the elemental abundances is available in the **vvgnei** model.

For the **gnei** model the parameters are:

par1	plasma temperature (keV)
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par3	Ionization timescale in units of s/cm^3.
par4	Ionization timescale averaged plasma temperature (keV)
par5	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vgnei** model, the parameters are:

par1	plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par14	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par15	Ionization timescale in units of s/cm^3.
par16	Ionization timescale averaged plasma temperature (keV)
par17	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvgnei** model, the parameters are:

par1	plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par31	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par32	Ionization timescale in units of s/cm^3.
par33	Ionization timescale averaged plasma temperature (keV)
par34	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.43 grad: accretion disk, Schwarzschild black hole

General Relativistic Accretion Disk model around a Schwarzschild black hole. Inner radius is fixed to be 3 Schwarzschild radii, thus the energy conversion efficiency is 0.057. See Hanawa, T., 1989, ApJ, 341, 948 and Ebisawa, K. Mitsuda, K. and Hanawa, T. 1991, ApJ, 367, 213. Several bugs were found in the old GRAD model which was included in xspec 11.0.1ae and before. Due to these bugs, it turned out that the mass obtained by fitting the old GRAD model to the observation was 1.4 times over-estimated. These bugs were fixed, and a new parameter (`par6`) was added to make the distinction between the old and new codes clear.

par1	distance (kpc)
par2	disk inclination angle (deg; 0 for face-on)
par3	mass of the central object (solar units)
par4	mass accretion rate 10^{18} gm/s
par5	spectral hardening factor, T_{col}/T_{eff}. Should be greater than 1.0, and considered to be 1.5–1.9 for accretion disks around a stellar-mass black hole. See, e.g., Shimura and Takahara, 1995, ApJ, 445, 780
par6	A flag to switch on/off the relativistic effects (never allowed to be free). If positive, relativistic calculation; if negative or zero, Newtonian calculation (the inner radius is still fixed at 3 Schwarzschild radii, and the efficiency is 1/12).
norm	Should be fixed to 1.

6.2.44 grbm: gamma-ray burst continuum

A model for gamma-ray burst continuum spectra developed by D. Band, et. al., 1993 (ApJ 413, 281).

$$A(E) = \begin{cases} K(E/100.)^{\alpha_1} \exp(-E/E_c) & \text{if } E < E_c(\alpha_1 - \alpha_2) \\ K[(\alpha_1 - \alpha_2)E_c/100]^{(\alpha_1-\alpha_2)}(E/100)^{\alpha_2} \exp(-(\alpha_1 - \alpha_2)) & \text{if } E > E_c(\alpha_1 - \alpha_2) \end{cases}$$

where E is in units of keV and the parameters are:

par1	α_1, first power law index
par2	α_2, second power law index
par3	E_c, characteristic energy in keV
norm	K, normalization constant

6.2.45 hatm: Nonmagnetic hydrogen atmosphere of a neutron star

This model provides the spectra emitted from a nonmagnetic hydrogen atmosphere of a neutron star. The model spectra in a 0 - 20 keV range of (unredshifted) photon energy are computed on a grid of surface gravity accelerations log(g)=13.7 - 14.9 (in cgs units) and effective temperatures T = 0.5 - 10 MK. For a given set of the fitting parameters, the surface gravity g and the gravitational redshift z are derived from the mass M and radius R of the star. The trial spectra are computed using a linear interpolation between the nearest model spectra on the T-log(g) grid, and the boundaries of the energy bins and the number of photons in each bin are divided by (1+z). The details of the model can be found in Suleimanov et al. (2016; A&A, submitted) and Klochkov et al. (2015).

The directory used for the files required by this model can be changed by using the **xset** command to set HATM.

The parameters are:

par1	Effective temperature T (MK)
par2	Neutron star mass M (M_{sol})
par3	Neutron star radius R (km)
par4	Normalization K = A/(d_{10}kpc)^2 where d_{10}kpc is the distance to the source in units of 10 kpc and A characterizes the fraction of the surface emitting the radiation (A=1 corresponds to the case when the radiation is emitted from the entire surface).

6.2.46 kerrbb: multi-temperature blackbody model for thin accretion disk around a Kerr black hole

A multi-temperature blackbody model for a thin, steady state, general relativistic accretion disk around a Kerr black hole. The effect of self-irradiation of the disk is considered, and the torque at the inner boundary of the disk is allowed to be non-zero. This model is intended as an extension to **grad**, which assumes that the black hole is non-rotating. For details see Li et al., ApJSuppl, 157, 335, 2005.

par1	eta, ratio of the disk power produced by a torque at the disk inner boundary to the disk power arising from accretion. It must be ≥ 0 and ≤ 1. When eta = 0, the solution corresponds to that of a standard Keplerian disk with zero torque at the inner boundary.
par2	specific angular momentum of the black hole in units of the black hole mass M (geometrized units G=c=1). Should be ≥ -1 and <1.
par3	disk's inclination angle (the angle between the axis of the disk and the line of sight). It is expressed in degrees. i=0 is for a "face-on" accretion disk. i should be ≤ 85 degree.
par4	the mass of the black hole in units of the solar mass.
par5	the "effective" mass accretion rate of the disk in units of 10^{18} g/s. When eta = 0 (zero torque at the inner boundary), this is just the mass accretion rate of the disk. When eta is nonzero, the effective mass accretion rate = (1+eta) times the true mass accretion rate of the disk. The total disk luminosity is then "epsilon" times "the effective mass accretion rate" times c^2, where epsilon is the radiation efficiency of a standard accretion disk around the Kerr black hole
par6	the distance from the observer to the black hole in units of kpc.
par7	spectral hardening factor, T_{col}/T_{eff}. It should be greater than 1.0, and considered to be 1.5–1.9 for accretion disks around a stellar-mass black hole. See, e.g., Shimura and Takahara 1995, ApJ, 445, 780
par8	a flag to switch on/off the effect of self-irradiation (never allowed to be free). Self-irradiation is included when >0. Self-irradiation is not included when ≤ 0.
par9	a flag to switch on/off the effect of limb-darkening (never allowed to be free). The disk emission is assumed to be limb-darkened when >0. The disk emission is assumed to be isotropic when lflag is ≤ 0.
K	normalization. Should be set to 1 if the inclination, mass and distance are frozen.

6.2.47 kerrd: optically thick accretion disk around a Kerr black hole

Optically thick extreme-Kerr disk model based on the same tranfer-function used in the **laor** Kerr disk-line model. Local emission is simply assumed to be the diluted blackbody. See Laor 1991, ApJ, 376, L90 for explanation of the transfer function. See Ebisawa et al. 2003, ApJ, 597, 780 for examples of using this model.

par1	distance (kpc)
par2	spectral hardening factor, Tcol/Teff. Should be greater than 1.0, and considered to be 1.5–1.9 for accretion disks around a stellar-mass black hole. See, e.g., Shimura and Takahara, 1995, ApJ, 445, 780
par3	mass of the central object (solar unit)
par4	mass accretion rate (10^{18} g/s)
par5	disk inclination angle (deg; 0 for face-on)
par6	inner radius (units of GM/c^2). 1.235 is the last stable orbit.
par7	outer radius (units of GM/c^2)
K	normalization factor. should be fixed to 1.

6.2.48 kerrdisk: accretion disk line emission with BH spin as free parameter

Model for an accretion disk broad emission line with the black hole spin allowed to be a free parameter. A detailed description can be found in Brenneman & Reynolds (2006ApJ...652.1028B).

This model is quite slow so is best used after models such as **laor** or **diskline** have been employed to get an estimate of the best-fit parameters.

par1	rest frame line energy (keV)
par2	emissivity index for the inner disk
par3	emissivity index for the outer disk
par4	break radius separating the inner and outer portions of the disk (gravitational radii)
par5	dimensionless black hole spin
par6	disk inclination angle to the line of sight (degrees)
par7	inner radius of the disk in units of the radius of marginal stability
par8	outer radius of the disk in units of the radius of marginal stability
par9	redshift z
norm	flux in line (photons/cm^2/s)

6.2.49 laor: accretion disk, black hole emission line

An emission line from an accreti on disk around a black hole. Ari Laor's calculation including GR effects (ApJ 376, 90).

par1	Line energy in keV
par2	a, power law dependence of emissivity (scales as R^{-a})
par3	inner radius (units of GM/c^2)
par4	outer radius (units of GM/c^2)
par5	inclination (degrees)
norm	photons/cm^2/s in the line

6.2.50 laor2: accretion disk with broken-power law emissivity profile, black hole emission line

An emission line from an accreti on disk with a broken power-law emissivity profile around a black hole. Uses Ari Laor's calculation including GR effects (ApJ 376, 90). Modified from laor model by Andy Fabian.

par1	Line energy in keV
par2	$Index$: power law dependence of emissivity (scales as R^{-Index})
par3	inner radius (units of GM/c^2)
par4	outer radius (units of GM/c^2)
par5	inclination (degrees)
par6	radius at which emissivity power-law index changes
par7	Emissivity power-law index for radii >par6
norm	photons/cm^2/s in the line

6.2.51 logpar: log-parabolic blazar model

logpar is a power-law with an index which varies with energy as a log parabola.

par1	a, slope at the pivot energy
par2	b, curvature term
par3	pivotE, fixed pivot energy (best near low end of energy range).
norm= K	

See for instance Massaro et al., 2004.

6.2.52 lorentz: lorentz line profile

A Lorentzian line profile.

$$A(E) = K(E_L/(2\pi))/[(E - E_L)^2 + (\sigma/2)^2]$$

where:

par1	E_L, line energy in keV
par2	σ, FWHM line width in keV
norm	K, photons/cm^2/s in the line

6.2.53 meka, vmeka: emission, hot diffuse gas (Mewe-Gronenschild)

An emission spectrum from hot diffuse gas based on the model calculations of Mewe and Gronenschild (as amended by Kaastra). The model includes line emissions from several elements. Abundances are the number of nuclei per Hydrogen nucleus relative to the Solar abundances set by the **abund** command.

For the **meka** model the parameters are:

par1	plasma temperature in keV
par2	H density (cm^{-3}
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the **abund** command.
par4	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vmeka** variant the parameters are as follows.

par1	plasma temperature in keV
par2	H density (cm^{-3}
par3–par14	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command).
par15	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

The references for the **meka** model are as follows :
Mewe, R., Gronenschild, E.H.B.M., and van den Oord, G.H.J. 1985 A&AS, 62, 197
Mewe, R., Lemen, J.R., and van den Oord, G.H.J. 1986, A&AS, 65, 511
Kaastra, J.S. 1992, An X-Ray Spectral Code for Optically Thin Plasmas (Internal SRON-Leiden Report, updated version 2.0)

Similar credit may also be given for the adopted ionization balance:
Arnaud, M., and Rothenflug, M. 1985, A& AS, 60, 425
Arnaud, M., and Raymond, J. 1992, ApJ, 398, 394

6.2.54 mekal, vmekal: emission, hot diffuse gas (Mewe-Gronenschild-Kaastra)

An emission spectrum from hot diffuse gas based on the model calculations of Mewe and Kaastra with Fe L calculations by Liedahl. The model includes line emissions from several elements. The `switch` parameter determines whether spectrum is calculated by running the mekal code, by interpolating on a pre-calculated mekal table, or using the AtomDB data. Relative abundances are set by the **abund** command for the **mekal** model. The **vmekal** variant allows the user to set the individual abundances for the model.

For the **mekal** model the parameters are:

par1	plasma temperature in keV
par2	H density (cm^{-3}
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the **abund** command.
par4	Redshift, z
par5	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using AtomDB data)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vmekal** variant the parameters are as follows.

par1	plasma temperature in keV
par2	H density (cm^{-3}
par3–par16	Abundances for He, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command).
par17	redshift, z
par18	Switch (0 = calculate, 1 = interpolate, 2 = interpolate using AtomDB data)
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

The references for the **meka** model are as follows :
Mewe, R., Gronenschild, E.H.B.M., and van den Oord, G.H.J. 1985 A&AS, 62, 197
Mewe, R., Lemen, J.R., and van den Oord, G.H.J. 1986, A&AS, 65, 511
Kaastra, J.S. 1992, An X-Ray Spectral Code for Optically Thin Plasmas (Internal SRON-Leiden Report, updated version 2.0)
Liedahl, D.A., Osterheld, A.L., and Goldstein, W.H. 1995, ApJL, 438, 115

Similar credit may also be given for the adopted ionization balance:
Arnaud, M., and Rothenflug, M. 1985, A& AS, 60, 425
Arnaud, M., and Raymond, J. 1992, ApJ, 398, 394

6.2.55 mkcflow, vmcflow: cooling flow, mekal

A cooling flow model after Mushotzky & Szymkowiak (Cooling Flows in Clusters and Galaxies ed. A. C. Fabian, 1988). This one differs from **cflow** in setting the

emissivity function to be the inverse of the bolometric luminosity. Abundance ratios are set by the **abund** command. Redshift is converted to distance using the cosmology set by the **cosmo** command. The `switch` parameter determines whether spectrum is calculated by running the mekal code, by interpolating on a pre-calculated mekal table, or using the AtomDB data.

For the **mkcflow** model the parameters are:

par1	low temperature (keV)
par2	high temperature (keV)
par3	abundance relative to Solar
par4	redshift
par5	switch (0 = calculate, 1 = interpolate, 2 = use AtomDB data)
norm	Mass accretion rate (solar mass/yr)

While for the **vmcflow** variant the parameters are:

par1	low temperature (keV)
par2	high temperature (keV)
par3–par16	Abundances for He, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Fe, Ni wrt Solar
par17	Redshift
par18	switch (0 = calculate, 1 = interpolate, 2 = use AtomDB data)
norm	Mass accretion rate (solar mass/yr)

6.2.56 nei, vnei, vvnei: collisional plasma, non-equilibrium, constant temperature

Non-equilibrium ionization collisional plasma model. This assumes a constant temperature and single ionization parameter. It provides a characterization of the spectrum but is not a physical model. The references for this model can be found under the description of the **equil** model. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The **vnei** variant allows the user to set the abundances for the more common elements. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full control over all the elemental abundances is available in the **vvnei** model.

For the **nei** model the parameters are:

par1	plasma temperature (keV)
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par3	Ionization timescale in units of s/cm^3.
par4	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vnei** model, the parameters are:

par1	plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par14	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par15	Ionization timescale in units of s/cm^3.
par16	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvnei** model, the parameters are:

par1	plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par31	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par32	Ionization timescale in units of s/cm^3.
par33	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.57 nlapec: continuum-only APEC emission spectrum

A continuum-only emission spectrum from collisionally-ionized diffuse gas calculated using the ATOMDB code. More information can be found at http://atomdb.org/ which should be consulted by anyone running this model. This default version number can be changed by modifiying the ATOMDB_VERSION string in your `Xspec.init` file.

By default this model reads atomic physics continuum data from the file apec_v[version]_coco.fits in the `$HEADAS/../spectral/modelData` directory. Different files can be specified by using the command **xset** APECROOT. There are three options. APECROOT can be set to a version number (eg 1.10, 1.2.0, 1.3.1, 2.0.1). In this case the value of APECROOT will be used to replace 2.0.2 in the name of the standard files and the resulting files will be assumed to be in the `modelData` directory. Alternatively, a filename root (eg `apec_v1.2.0`) can be given. This root will be used as a prefix for the `_coco.fits` . Finally, if neither of these work then the model will assume that the APECROOT value gives the complete directory path, e.g.

```
XSPEC12> xset APECROOT /foo/bar/apec_v1.2.0
```

will use the input file `/foo/bar/apec_v1.2.0_coco.fits`.

The **nlapec** model uses abundances set by the **abund** command. The abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** APEC_TRACE_ABUND. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Note that this means that the **apec** and **vapec** models will show emission lines even if the abundance parameters are set to zero.

The parameters are:

par1	plasma temperature, keV
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the **abund** command. The trace element abundances are from **xset** APEC_TRACE_ABUND, the default is 1.0.
par3	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int \eta_e \eta_H \mathrm{d}V$, where D_A is the angular diameter distance to the source (cm), η_e and η_H are the electron and H densities (cm^{-3})

6.2.58 npshock, vnpshock, vvnpshock: shocked plasma, plane parallel, separate ion, electron temperatures

Plane-parallel shock plasma model with separate ion and electron temperatures. This model is slow. `par1` provides a measure of the average energy per particle (ions+electrons) and is constant throughout the postshock flow in plane shock models (Borkowski et al., 2001, ApJ, 548, 820). `par2` should always be less than `par1`. If `par2` exceeds `par1` then their interpretations are switched (ie the larger of `par1` and `par2` is always the mean temperature). Additional references can be found under the help for the **equil** model. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The

vnpshock variant allows the user to set the abundances for the more common elements. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full control over all the elemental abundances is available in the **vvnpshock** model.

For the **npshock** model the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par4	Lower limit on ionization timescale in units of s/cm^3.
par5	Upper limit on ionization timescale in units of s/cm^3.
par6	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vnpshock** model, the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par15	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par16	Lower limit on ionization timescale in units of s/cm^3.
par17	Upper limit on ionization timescale in units of s/cm^3.
par18	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvnpshock** model, the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par32	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par33	Lower limit on ionization timescale in units of s/cm^3.
par34	Upper limit on ionization timescale in units of s/cm^3.
par35	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.59 nsa: neutron star atmosphere

This model provides the spectra in the X-ray range (0.05–10 keV) emitted from a hydrogen atmosphere of a neutron star. There are three options : nonmagnetized (B $< 10^8 - 10^9$ G) with a uniform surface (effective) temperature in the range of $\log T_{eff}(K) = 5.0 - 7.0$; a field B $= 10^{12}$ G with a uniform surface (effective) temperature in the range of $\log T_{eff}(K) = 5.5 - 6.8$; a field B $= 10^{13}$ G with a uniform surface (effective) temperature in the range of $\log T_{eff}(K) = 5.5 - 6.8$. The atmosphere is in radiative and hydrostatic equilibrium; sources of heat are well below the atmosphere. The Comptonization effects (significant at $T_{eff} > 3 \times 10^6$ K) are taken into account. The model spectra are provided as seen by a distant observer, with allowance for the GR effects. The user is advised to keep M_{ns} and R_{ns} fixed and fit the temperature and the normalization. MagField must be fixed at one of 0, 10^{12}, or 10^{13}.

The values of the effective temperature and radius as measured by a distant observer ("values at infinity") are :

$$T_{eff}^{\rm inf} = T_{eff} \times g_r R_{ns}^{\rm inf} = R_{ns}/g_r$$

where

$$g_r = \sqrt{1 - 2.952 \times M_{ns}/R_{ns}}$$

is the gravitational redshift parameter.

Please send your comments/questions to Slava Zavlin (VYACHESLAV.ZAVLIN@msfc.nasa.gov) and/or George Pavlov (pavlov@astro.psu.edu). If you publish results obtained using these models, please reference Zavlin, V.E., Pavlov, G.G., & Shibanov, Yu.A. 1996, A&A, 315, 141 for nonmagnetic models, and Pavlov, G.G., Shibanov, Yu.A., Zavlin, V.E., & Meyer, R.D. 1995, in "The Lives of the Neutron Stars,"

ed. M.A. Alpar, U. Kiziloglu, & J. van Paradijs (NATO ASI Ser. C, 450; Dordrecht: Kluwer), p. 71 for magnetic models.

par1	$\log T_{eff}$, (unredshifted) effective temperature
par2	M_{ns}, neutron star gravitational mass (in units of solar mass)
par3	R_{ns}, neutron star radius (in km)
par4	neutron star magnetic field strength (0, 10^{12}, or 10^{13} G)
norm	$1/D^2$, where D is the distance of the object in pc.

6.2.60 nsagrav: NS H atmosphere model for different g

This model provides the spectra emitted from a nonmagnetic hydrogen atmosphere of a neutron star with surface gravitational acceleration g ranging from 10^{13} to 10^{15} cm/s^2, allowed by equations of state for the neutron star matter (the **nsa** model gives the spectra calculated for $g = 2.43 \times 10^{14}$ cm/s^2). The uniform surface (effective) temperature is in the range of $\log T_{eff}(K) = 5.5 - 6.5$. The atmosphere is in radiative and hydrostatic equilibrium; sources of heat are well below the atmosphere. The radiative force and electron heat conduction are included in the models, but they are of no importance in the specified ranges of T_{eff} and g. The model spectra are provided as seen by a distant observer, with allowance for the GR effects.

The neutron star mass M and radius R determine the redshift parameter,

$$g_r = sqrt1 - 2.952 \times M/R$$

and the gravitational acceleration at the surface,

$$g = 1.33 \times 10^{16} \times M/R^2/g_r$$

cm/s^2, where M is in units of solar mass, and R is in km.

The allowed domain in the M–R plane corresponds to $g_r^2 > 1/3$ and $10^{13} < g < 10^{15}$ cm/s^2. (This domain is restricted by the solid curves in the figure). If chosen M and R values correspond to g_r or/and g values outside the allowed domain, then the code sets the latter to be the closest limiting values (e.g., if one chooses $M = 2$, $R = 8$, then the code will use $g_r = 3 - 1/2 = 0.578$ instead of $g_r = 0.512$ corresponding to the M and R chosen), which would lead to unphysical results.

The values of the effective temperature and radius as measured by a distant observer ("values at infinity") are:

$$T_{inf} = g_r \times T_{eff}, R_{inf} = R/g_r$$

The **nsagrav** model may be useful for putting constraints on M and R from spectral fits to thermal emission detected from neutron stars, provided the quality of the observational data are good enough to warrant a detailed analysis. The parameters M and R can be fixed at specific values or allowed to vary within a reasonable range (see the note above). For example, one can run spectral fits on a M–R grid (using the **steppar** command) within the allowed parameter domain (see above).

Please send your comments/questions (if any) to Slava Zavlin (vyacheslav.zavlin@msfc.nasa.gov) and/or George Pavlov (pavlov@astro.psu.edu). If you publish results obtained using this model please reference Zavlin et al. (1996, A&A 315, 141).

par1	$\log T_{eff}$: (unredshifted) effective temperature
par2	M_{ns} : neutron star gravitational mass (in units of Solar mass)
par3	R_{ns} : "true" neutron star radius (km)
norm	$1/D^2$ where D is the distance to the object in pc

6.2.61 nsatmos: NS Hydrogen Atmosphere model with electron conduction and self-irradiation

This model interpolates from a grid of NS atmosphere calculations provided by George Rybicki and Ramesh Narayan to output a NS atmosphere spectrum. The model grids cover a wide range of surface gravity and effective temperature, and incorporate thermal electron conduction and self-irradiation by photons from the compact object. This code assumes negligible (less than 10^9 G) magnetic fields and a pure hydrogen atmosphere. A detailed description of the model is given in Heinke et al. (2006), ApJ 644, 1090 (see also McClintock et al. 2004, ApJ, 615, 402).

par1	$\log T_{eff}$: (unredshifted) effective temperature
par2	M_{ns} : neutron star gravitational mass (in units of Solar mass)
par3	R_{ns} : "true" neutron star radius (km)
par4	dist : distance to the neutron star (in kpc)
norm	fraction of the neutron star surface emitting

6.2.62 nsmax: Neutron Star Magnetic Atmosphere

This model has been superseded by **nsmaxg**.

This model interpolates from a grid of neutron star (NS) atmosphere spectra to produce a final spectrum that depends on the parameters listed below. The

atmosphere spectra are obtained using the latest equation of state and opacity results for a partially ionized, strongly magnetized hydrogen or mid-Z element plasma. The models are constructed by solving the coupled radiative transfer equations for the two photon polarization modes in a magnetized medium, and the atmosphere is in radiative and hydrostatic equilibrium. The atmosphere models mainly depend on the surface effective temperature T_{eff} and magnetic field strength B and inclination Θ_B; there is also a dependence on the surface gravity $g = (1 + z_g)GM/R^2$, where $1 + z_g = \sqrt{1 - 2GM/R}$ is the gravitational redshift and M and R are the NS mass and radius, respectively.

Two sets of models are given: one set with a single surface B and T_{eff} and a set which is constructed with B and T_{eff} varying across the surface according to the magnetic dipole model (for the latter, θ_m is the angle between the direction to the observer and the magnetic axis). The effective temperatures span the range $\log T_{eff} = 5.5 - 6.8$ for hydrogen and $\log T_{eff} = 5.8 - 6.9$ for mid-Z elements. The models with single (B,T_{eff}) cover the energy range 0.05–10 keV, while the models with (B,T_{eff})-distributions cover the range 0.09–5 keV.

par1	$\log T_{eff}$, surface (unredshifted) effective temperature
par2	$1 + z_g$, gravitational redshift
par3	switch indicating model to use
norm	$A = (R_{em}/d)^2$, normalization, where R_{em} is the size (in km) of the emission region and d is the distance (kpc) to the object Note: A is added automatically by XSPEC.

The models available by setting `par3` are:

Switch	Element	$B(10^{12}$ G)	Θ_B	θ_m	$\log g$ (cm/s^2)	$\log T_{eff}$	E (keV)
1000	H	0.01	0	N/A	2.4	5.5 – 6.7	0.05 - 10
1060	H	0.04	0	N/A	2.4	5.5 – 6.7	0.05 - 10
1085	H	0.07	0	N/A	2.4	5.5 – 6.7	0.05 - 10
1100	H	0.1	0	N/A	2.4	5.5 – 6.7	0.05 - 10
1200	H	1.0	0	N/A	0.4 – 2.5	5.5 – 6.7	0.05 - 10
1230	H	2.0	0	N/A	2.4	5.5 – 6.8	0.05 - 10
1260	H	4.0	0	N/A	2.4	5.5 – 6.8	0.05 - 10
1280	H	7.0	0	N/A	2.4	5.5 – 6.8	0.05 - 10
1300	H	10.0	0	N/A	0.4 – 2.5	5.5 – 6.7	0.05 - 10
1330	H	20.0	0	N/A	2.4	5.6 – 6.8	0.05 - 10
1350	H	30.0	0	N/A	2.4	5.7 – 6.8	0.05 - 10
1211	H	1.26	0	N/A	1.6	5.5 – 6.8	0.05 - 10
1281	H	7.0	0	N/A	1.6	5.5 – 6.8	0.05 - 10
12006	C	1.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
13006	C	10.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
12008	O	1.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
13008	O	10.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
12010	Ne	1.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
13010	Ne	10.0	0	N/A	2.4	5.8 – 6.9	0.05 - 10
123100	H	1.0 – 1.82	0-90	0	1.6	5.5 – 6.8	0.09 - 5
123190	H	1.0 – 1.82	0-90	90	1.6	5.5 – 6.8	0.09 - 5
130100	H	5.5 – 10.0	0-90	0	1.6	5.5 – 6.8	0.09 - 5
130190	H	5.5 – 10.0	0-90	90	1.6	5.5 – 6.8	0.09 - 5

If you publish results obtained using **nsmax**, please reference Ho, W.C.G., Potekhin, A.Y., & Chabrier, G. (2008, ApJS, 178, 102) and also Mori, K. & Ho, W.C.G. (2007, MNRAS, 377, 905) if using the mid-Z models. See Ho, W.C.G. (2014, Proc. of IAUS 302, submitted) for discussion of **nsmaxg** and **nsmax**.

6.2.63 nsmaxg: Neutron Star with a Magnetic Atmosphere

The **nsmaxg** model interpolates from a grid of neutron star (NS) atmosphere spectra to produce a final spectrum that depends on the parameters listed below. Atmosphere spectra are obtained using the latest equation of state and opacity results for a partially ionized, strongly magnetized hydrogen or mid-Z element plasma. Models are constructed by solving the coupled radiative transfer equations for the two photon polarization modes in a magnetized medium, and the atmosphere is in radiative and hydrostatic equilibrium. Atmosphere models mainly depend on the surface effective temperature T_{eff} and magnetic field strength B and inclination Θ_B; there is also a dependence on the surface gravity $g = (1 + z_g)GM/R^2$, where $1 + z_g = \sqrt{1 - 2GM/R}$ is the gravitational redshift and M and R are the NS mass and radius, respectively.

Two sets of models are available: one set with a single surface B and T_{eff} [some models allow for varying g, in the range $\log g$ (cm/s^2) = 13.6–15.4] and a set which is constructed with B and T_{eff} varying across the surface according to the magnetic dipole model (θ_m is the angle between the direction to the observer and the magnetic axis). Effective temperatures span the range $\log T_{eff}(K)$ = 5.5–6.8. Models with single (B,T_{eff}) cover the energy range 0.05–10 keV, while models with (B,T_{eff})-distributions cover the range 0.09–5 keV.

par1	$\log T_{eff}$, surface (unredshifted) effective temperature
par2	M, neutron star gravitational mass
par3	R, neutron star radius
par4	d, distance to neutron star
par5	switch indicating model to use
norm	1, normalization (though not strictly correct, can be varied to change the size of the emission region, $(R_{em}/R)^2$)

The models available by setting `par5` are:

Switch	Element	$B(10^{12}$ G)	Θ_B	θ_m	$\log g$ (cm/s^2)	$\log T_{eff}$	E (keV)
1000	H	0.01	0	N/A	2.4	5.5 – 6.7	0.05 – 10
1060	H	0.04	0	N/A	2.4	5.5 – 6.7	0.05 – 10
1085	H	0.07	0	N/A	2.4	5.5 – 6.7	0.05 – 10
1100	H	0.1	0	N/A	2.4	5.5 – 6.7	0.05 – 10
1200	H	1.0	0	N/A	0.4 – 2.5	5.5 – 6.8	0.05 – 10
1230	H	2.0	0	N/A	2.4	5.5 – 6.8	0.05 – 10
1260	H	4.0	0	N/A	2.4	5.5 – 6.8	0.05 – 10
1280	H	7.0	0	N/A	2.4	5.5 – 6.8	0.05 – 10
1300	H	10.0	0	N/A	0.4 – 2.5	5.5 – 6.8	0.05 – 10
1330	H	20.0	0	N/A	2.4	5.6 – 6.8	0.05 – 10
1350	H	30.0	0	N/A	2.4	5.7 – 6.8	0.05 – 10
1211	H	1.26	0	N/A	1.6	5.5 – 6.8	0.05 – 10
1281	H	7.0	0	N/A	1.6	5.5 – 6.8	0.05 – 10
12006	C	1.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
13006	C	10.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
12008	O	1.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
13008	O	10.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
12010	Ne	1.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
13010	Ne	10.0	0	N/A	2.4	5.8 – 6.9	0.05 – 10
123100	H	1.0 – 1.82	0-90	0	1.6	5.5 – 6.8	0.09 – 5
123190	H	1.0 – 1.82	0-90	90	1.6	5.5 – 6.8	0.09 – 5
130100	H	5.5 – 10.0	0-90	0	1.6	5.5 – 6.8	0.09 – 5
130190	H	5.5 – 10.0	0-90	90	1.6	5.5 – 6.8	0.09 – 5

If you publish results obtained using **nsmaxg**, please reference Ho, W.C.G., Potekhin, A.Y., & Chabrier, G. (2008, ApJS, 178, 102) and also Mori, K. & Ho, W.C.G. (2007, MNRAS, 377, 905) if using mid-Z models. See Ho, W.C.G.

(2014, Proc. of IAUS 302, 435) for discussion of **nsmaxg** and **nsmax**.

6.2.64 nsx: neutron star with a non-magnetic atmosphere

The **nsx** model interpolates from a grid of neutron star (NS) atmosphere spectra to produce a final spectrum that depends on the parameters listed below. Atmosphere spectra are obtained using opacity tables computed by The Opacity Project and are for non-magnetic atmospheres (note that **nsx** is fully compatible with the magnetic atmosphere spectral tables of **nsmaxg**, and both **nsx** and **nsmaxg** spectral tables can easily be made compatible with other XSPEC NS fitting models). Atmosphere models are constructed by solving the radiative transfer equation, and the atmosphere is assumed to be in radiative and hydrostatic equilibrium. Atmosphere models depend on the surface effective temperature T_{eff} and surface gravity $g = (1+z_g)GM/R^2$, where $1+z_g = \sqrt{1-2GM/R}$ is the gravitational redshift and M and R are the NS mass and radius, respectively. The parameters are:

par1	$\log T_{eff}$, surface (unredshifted) effective temperature
par2	M, neutron star gravitation mass (in units of Solar mass)
par3	R, neutron star radius (in km)
par4	d, distance to neutron star (in kpc)
par5	Switch indicating model to use
norm	1, normalization (though not strictly correct, can be varied to change the size of the emission region, $(R_{em}/R)^2$)

The models available by setting `par5` are:

Switch	Element	$\log g$ (cm/s^2)	$\log T_{eff}$	E (keV)
2	He	13.6 - 14.9	5.5 - 6.7	0.03 - 15
6	C	13.6 - 15.3	5.9 - 6.6	0.01 - 15

If you publish results obtained using **nsx**, please reference Ho, W.C.G. & Heinke, C.O. (2009, Nature, 462, 71).

6.2.65 nteea: non-thermal pair plasma

A nonthermal pair plasma model based on that of Lightman & Zdziarski (1987, ApJ 319, 643) from Magdziarz and Zdziarski. It includes angle-dependent reflection from Magdziarz & Zdziarski (1995, MNRAS 273, 837). In versions 1.1 and above the Compton reflection is done through an internal call to the **reflect** model. The Greens' function integration required for the Compton reflection calculation is performed to an accuracy of 0.01 (i.e. 1%). This can be changed

using e.g. **xset NTEEA_PRECISION 0.05**.The abundances are set up by the command **abund**. Send questions or comments to aaz@camk.edu.pl

par1	nonthermal electron compactness
par2	blackbody compactness
par3	scaling factor for reflection (1 for isotropic source above disk)
par4	blackbody temperature in eV
par5	the maximum Lorentz factor
par6	thermal compactness (0 for pure nonthermal plasma)
par7	Thomson optical depth of ionization electrons (e.g., 0)
par8	electron injection index (0 for monoenergetic injection)
par9	minimum Lorentz factor of the power law injection (not used for monoenergetic injection)
par10	minimum Lorentz factor for nonthermal reprocessing
par11	radius in cm (for Coulomb/bremsstrahlung only)
par12	pair escape rate in c (0-1, see Zdziarski 1985, ApJ, 289, 514)
par13	cosine of inclination angle
par14	iron abundance relative to that defined by **abund**
par15	redshift z
norm	photon flux of the direct component (w/o reflection) at 1 keV in the observer's frame.

6.2.66 Nthcomp: Thermally comptonized continuum

Nthcomp is a **much** better description of the continuum shape from thermal comptonisation than an exponentially cutoff power law, but is not that much more complicated in terms of parameters. The high energy cutoff is sharper than an exponential, and is parameterized by the electron temperature (kT_e). VERY roughly, an exponential rollover energy $E_c = 2 - 3kT_e$ but the shape is very different, so it impacts on the reflected fraction as well. Another major effect (especially for X-ray binaries) is that it incorporates the low energy rollover. The hot electrons Compton UPscatter seed photons so there are few photons in the scattered spectrum at energies below the typical seed photon energies, making it significantly different to a power law below this energy. Typically the physical picture is that these seed photons are (quasi)blackbody (eg neutron star boundary layer) or disk blackbody in shape. Either of these shapes can be selected (input type), both being parameterized by a seed photon temperature (kT_{bb}). Between the low and high energy rollovers the shape of the spectrum is set by the combination of electron scattering optical depth and electron temperature. It is not necessarily a power law, but can be parameterized by an asymptotic power law index (Γ). Details of this are given in Zycki, Done & Smith (1999), including a self-consistent reflection component which is NOT released here as it was written using non-FITS standard files so has significant

issues with portability.

This is the thermally comptonized continuum model of Zdziarski, Johnson & Magdziarz 1996, MNRAS, 283, 193, as extended by Zycki, Done & Smith 1999, MNRAS 309, 561. Please reference these papers if you use it.

par1	Γ, asymptotic power-law photon index.
par2	kT_e, electron temperature (high energy rollover)
par3	kT_{bb}, seed photon temperature (low energy rollover)
par4	inp_type, 0 or 1 for blackbody or disk-blackbody seed photons, respectively
par5	redshift
norm	normalization, unity at 1 keV for a norm of 1.

6.2.67 Optxagnf, optxagn: Colour temperature corrected disc and energetically coupled Comptonisation model for AGN.

AGN spectral energy distributions are complex, but can be phenomenologically fit by a disc, optically thick, low temperature thermal Comptonisation (to produce the soft X-ray excess) and an optically thin, high temperature themal Comptonisation (to produce the power law emission which dominates above 2 keV). Here we combine these three components together assuming that they are all ultimately powered by gravitational energy released in accretion. We assume that the gravitational energy released in the disc at each radius is emitted as a (colour temperature corrected) blackbody only down to a given radius, R_{corona}. Below this radius, we further assume that the energy can no longer completely thermalise, and is distributed between powering the soft excess component and the high energy tail. This imposes an important energetic self consistency on the model. The key aspect of this model is that the optical luminosity constrains the mass accretion rate through the outer disc, $\dot{M}$, provided there is an independent estimate of the black hole mass (from e.g. the Hβ emission line profile). The total luminosity available to power the entire SED is $L_{tot} = eff\dot{M}c^2$, where the efficiency is set by black hole spin assuming Novikov-Thorne emissivity.

There are two versions of the model. **optxagnf** is the one recommended for most purposes, and has the colour temperature correction calculated for each temperature from the approximations given in Done et al. (2011). **optxagn** instead allows the user to define their own colour temperature correction, f_{col}, which is then applied to annuli with effective temperature $> T_{scatt}$. In both models the flux is set by the physical parameters of mass, $L/L)Edd$, spin and distance, hence the model normalisations MUST be frozen at unity.

Parameters in **optxagnf**:

par1	mass, black hole mass in solar masses
par2	dist, comoving (proper) distance in Mpc
par3	$\log L/L_{edd}$, Eddington ratio
par4	a_{star}, dimensionless black hole spin
par5	r_{cor}, coronal radius in $R_g = GM/c^2$ marking the transition from (colour temperature corrected) blackbody emission to a Comptonised spectrum. If this parameter is negative then only the blackbody component is used.
par6	$\log r_{out}$, log of the outer radius of the disc in units of R_g; if this is -ve the code will use the self gravity radius as calculated from Laor & Netzer (1989)
par7	kT_e, electron temperature for the soft Comptonisation component (soft excess) in keV
par8	τ, pptical depth of the soft Comptonisation component. If this parameter is negative then only the soft Compton component is used.
par9	Γ, spectral index of the hard Comptonisation component ("power law") which has temperature fixed to 100 keV.
par10	f_{pl}, fraction of the power below r_{cor} which is emitted in the hard comptonisation component. If this parameter is negative then only the hard Compton component is used.
par11	Redshift
norm	K, must be frozen

Parameters in **optxagn**:

par1	mass, black hole mass in solar masses
par2	dist, comoving (proper) distance in Mpc
par3	$\log L/L_{edd}$, Eddington ratio
par4	a_{star}, dimensionless black hole spin
par5	r_{cor}, coronal radius in $R_g = GM/c^2$ marking the transition from (colour temperature corrected) blackbody emission to a Comptonised spectrum. If this parameter is negative then only the blackbody component is used.
par6	$\log r_{out}$, log of the outer radius of the disc in units of R_g; if this is -ve the code will use the self gravity radius as calculated from Laor & Netzer (1989)
par7	kT_e, electron temperature for the soft Comptonisation component (soft excess) in keV
par8	τ, pptical depth of the soft Comptonisation component. If this parameter is negative then only the soft Compton component is used.
par9	Γ, spectral index of the hard Comptonisation component ("power law") which has temperature fixed to 100 keV.
par10	f_{pl}, fraction of the power below r_{cor} which is emitted in the hard comptonisation component. If this parameter is negative then only the hard Compton component is used.
par11	f_{col}, colour temperature correction to apply to the disc blackbody emission for radii below r_{cor} with effective temperature $> T_{scatt}$
par12	T_{scatt}, effective temperature criterion used as described above (in K).
par13	Redshift
norm	K, must be frozen

6.2.68 pegpwrlw: power law, pegged normalization

A power law with pegged normalization.

$$A(E) = KE^{-\alpha}$$

where :

par1	α, photon index of power law (dimensionless)
par2	lower peg energy range (keV)
par3	upper peg energy range (keV)
norm	flux (in units of 10^{-12} ergs/cm^2/s over the energy `par2`–`par3`). If `par2` = `par3`, it is the flux in micro-Jy at `par2`.

6.2.69 pexmon: neutral Compton reflection with self-consistent Fe and Ni lines.

This model from Nandra et al. (2007; MNRAS 382, 194) combines **pexrav** with self-consistently generated Fe Kα, Fe Kβ, Ni Kα and the Fe Kα Compton shoulder. Line strengths are based on Monte Carlo calculations by George and Fabian (1991; MNRAS 249, 352) which are parametrized for $1.1 < \gamma < 2.5$ by :

$$EW = 9.66 EW_0(\gamma^{-2.8} - 0.56)$$

with inclination dependence for $i < 85$ degrees :

$$EW = EW_0(2.20\cos i - 1.749(\cos i)^2 + 0.541(\cos i)^3)$$

and abundance dependence :

$$\log EW = \log EW_0(0.0641 \log A_{Fe} - 0.172(\log A_{Fe})^2)$$

The Fe Kβ and Ni Kα line fluxes are 11.3% and 5% respectively of that for Fe Kα. The Fe Kα Compton shoulder is approximated as a gaussian with E = 6.315 keV and $\sigma = 0.035$ keV. The inclination dependence is taken from Matt (2002; MNRAS 337, 147) such that :

$$EW_{shoulder} = EW_{FeK\alpha}(0.1 + 0.1\cos i)$$

The model parameters are :

par1	γ, power-law photon index, $N_E \propto E^{-\gamma}$.
par2	E_c, cutoff energy in keV (if $E_c = 0$ there is no cutoff).
par3	scale, the scaling factor for reflection; <0 = no direct component, 1 = isotropic source above the disk
par4	z, redshift
par5	A, abundance of elements heavier than He relative to Solar.
par6	A_{fe}, iron abundance relative to Solar.
par7	the inclination angle (degrees).
norm	normalization is the photon flux at 1 keV (photons/keV/cm^2/s) of the cutoff power law only (without reflection) and in the Earth frame.

6.2.70 pexrav: reflected powerlaw, neutral medium

Exponentially cut off power law spectrum reflected from neutral material (Magdziarz & Zdziarski 1995, MNRAS, 273, 837). The output spectrum is the sum of the cut-off power law and the reflection component. The reflection component alone can be obtained for $rel_{refl} < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to then change the limits of rel_{refl} to exclude zero (as then the direct component appears). If $E_c = 0$ there is no cutoff in the power law. The metal and iron abundance are variable with respect to those defined by the command **abund**. The opacities are those set by the command **xsect**. As expected in AGNs, H and He are assumed to be fully ionized

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting PEXRAV_PRECISION eg **xset PEXRAV_PRECISION 0.05**. The default precision is 0.01 (ie 1%).

par1	Γ, first power law photon index, $N_E \propto E^{\Gamma}$
par2	E_c, cutoff energy (keV) (if $E_c = 0$ there is no cutoff)
par3	rel_{refl}, reflection scaling factor (0 = no reflected component, <0 reflection component only)
par4	redshift, z
par5	abundance of elements heavier than He relative to the solar abundances
par6	iron abundance relative to that defined by **abund**
par7	cosine of inclination angle
norm	photon flux at 1 keV (photons/keV/cm^2/s) of the cutoff broken power-law only (no reflection) in the observed frame.

6.2.71 pexriv: reflected powerlaw, ionized medium

Exponentially cut off power law spectrum reflected from ionized material (Magdziarz & Zdziarski 1995, MNRAS, 273, 837). Ionization and opacities of the reflecting medium is computed as in the **absori** model. The output spectrum is the sum of the cut-off power law and the reflection component. The reflection component alone can be obtained for $rel_{refl} < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to then change the limits of rel_{refl} to exclude zero (as then the direct component appears). If $E_c = 0$ there is no cutoff in the power law. The metal and iron abundance are variable with respect to those defined by the command **abund**.

The core of this model is a Greens' function integration with one numerical inte-

gral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting PEXRIV_PRECISION eg **xset PEXRIV_PRECISION 0.05**. The default precision is 0.01 (ie 1%).

par1	Γ, first power law photon index, $N_E \propto E^{\Gamma}$
par2	E_c, cutoff energy (keV) (if $E_c = 0$ there is no cutoff)
par3	rel_{refl}, reflection scaling factor (0 = no reflected component, <0 reflection component only)
par4	redshift, z
par5	abundance of elements heavier than He relative to the solar abundances
par6	iron abundance relative to that defined by **abund**
par7	cosine of inclination angle
par8	disk temperature in K
par9	disk ionization parameter, $\xi = 4\pi F_{ion}/n$, where F_{ion} is the 5eV – 20keV irradiating flux, n is the density of the reflector; see Done et al., 1992, ApJ, 395, 275.
norm	photon flux at 1 keV (photons/keV/cm^2/s) of the cutoff broken power-law only (no reflection) in the observed frame.

6.2.72 plcabs: powerlaw observed through dense, cold matter

This model describes X-ray transmission of an isotropic source of photons located at the center of a uniform, spherical distribution of matter, correctly taking into account Compton scattering. The model can be used for radial column densities up to 5×10^{-24} cm^{-2}. The valid energy range for which data can be modeled is between 10 and 18.5 keV, depending on the column density. Details of the physics of the model, the approximations used and further details on the regimes of validity can be found in Yaqoob (1997; ApJ, 479, 184). In this particular incarnation, the initial spectrum is a power law modified by a high-energy exponential cut-off above a certain threshold energy.

Also, to improve the speed, a FAST option is available in which a full integration over the input spectrum is replaced by a simple mean energy shift for each bin. This option is obtained by setting `par9` to a value of 1 or greater and cannot be made variable. Further, for single-scattering albedos less than the critical albedo (i.e. `par8`) energy shifts are neglected altogether. The recommended value for `par8` is 0.1 which corresponds to about 4 keV for cosmic abundances and is more than adequate for ASCA data.

Note that for column densities in the range 10^{23} – 10^{24} cm^{-2}, the maximum number of scatterings which need be considered for convergence of the spectrum of better than 1% is between 1 and 5. For column densities as high as 5×10^{24} cm^{-2}, the maximum number of scatterings which need be considered for the same level of convergence is 12. This parameter cannot be made variable.

par1	Column density in units 10^{22} cm^{-2}
par2	Maximum number of scatterings to consider.
par3	Iron abundance.
par4	Iron K edge energy.
par5	Power-law photon index.
par6	High-energy cut-off threshold energy.
par7	High-energy cut-off e-folding energy.
par8	Critical albedo for switching to elastic scattering.
par9	If `par9` >1, function uses mean energy shift, not integration.
par10	Source redshift, z
norm	Normalization factor

6.2.73 posm: positronium continuum

Positronium continuum (Brown & Leventhal 1987 ApJ 319, 637)

$$A(E) = K\frac{2}{(\pi^2-9)E_c}[\frac{E(E_c-E)}{(2E_c-E)^2}+\frac{2E_c(E_c-E)}{E^2}\log(\frac{E_c-E}{E_c})-\frac{2E_c(E_c-E)^2}{(2E_c-E)^3}\log(\frac{E_c-E}{E_c})+\frac{(2E_c-E)}{E}]$$

for $E < E_c = 511$ keV, where:

norm	K, normalization

6.2.74 powerlaw, zpowerlw: power law photon spectrum

powerlaw is a simple photon power law. The **zpowerlw** variant computes a redshifted spectrum.

$$A(E) = KE^{-\alpha}$$

where:

par1	α, photon index of power law (dimensionless)
norm	K, photons/keV/cm^2/s at 1 keV

For **zpowerlw** the formula and corresponding parameters are:

$$A(E) = K[E(1+z)]^{-\alpha}$$

where:

par1	α, photon index of power law (dimensionless)
par2	z, redshift
norm	K, photons/keV/cm^2/s at 1 keV

If POW_EMIN and POW_EMAX have been defined by the **xset** command then the norm becomes the flux in units of 10^{-12} ergs/cm^2/s over the energy range (POW_EMIN, POW_EMAX) keV unless POW_EMIN = POW_EMAX in which case the norm becomes the flux density in micro-Jansky at POW_EMIN keV. In these cases it is important that POW_EMIN and POW_EMAX lie within the energy range on which the model is being evaluated.

6.2.75 pshock, vpshock, vvpshock: plane-parallel shocked plasma, constant temperature

Constant temperature plane-parallel shock plasma model. The references for this model can be found under the description of the **equil** model. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The **vpshock** variant allows the user to set the abundances for the more common elements. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set

either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full control over all the elemental abundances is available in the **vvpshock** model.

For the **pshock** model the parameters are:

par1	Plasma temperature (keV)
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par3	Lower limit on ionization timescale in units of s/cm^3.
par4	Upper limit on ionization timescale in units of s/cm^3.
par5	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vpshock** model, the parameters are:

par1	Plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par14	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par15	Lower limit on ionization timescale in units of s/cm^3.
par16	Upper limit on ionization timescale in units of s/cm^3.
par17	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvpshock** model, the parameters are:

par1	Plasma temperature (keV)
par2	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par3–par31	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par32	Lower limit on ionization timescale in units of s/cm^3.
par33	Upper limit on ionization timescale in units of s/cm^3.
par34	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.76 raymond, vraymond: emission, hot diffuse gas, Raymond-Smith

An emission spectrum from hot, diffuse gas based on the model calculations of Raymond and Smith (ApJS 35, 419 and additions) including line emissions from several elements. This model interpolates on a grid of spectra for different temperatures. The grid is logarithmically spaced with 80 temperatures ranging from 0.008 to 80 keV.

The **vraymond** variant allows independent parameters to set the abundances. Abundances are the number of nuclei per Hydrogen nucleus relative to the Solar abundances as set by the **abund** command.

For the **raymond** model the parameters are:

par1	plasma temperature, keV
par2	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the **abund** command.
par3	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2} \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vraymond** variant the parameters are as follows.

par1	plasma temperature, keV
par2–par13	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command).
par14	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

6.2.77 redge: emission, recombination edge

Recombination edge emission.

$$A(E) = \begin{cases} 0 & \text{if } E < E_c \\ K(1/T_p)\exp[\frac{-(E-E_c)}{T_p}] & \text{if } E >= E_c \end{cases}$$

where:

par1	E_c, threshold energy
par2	T_p, plasma temperature (keV)
norm	K, photons/cm^2/s in the line

6.2.78 refsch: reflected power law from ionized accretion disk

Exponentially cut-off power-law spectrum reflected from an ionized relativistic accretion disk. In this model, spectrum of **pexriv** is convolved with a relativistic disk line profile **diskline**. See Magdziarz & Zdziarski 1995 MNRAS, 273, 837 for details of Compton reflection. See Fabian et al. 1989, MNRAS, 238, 729 for details of the disk line profile.

par1	Γ, power law photon index, $N_E \propto E^{-\Gamma}$
par2	E_c, cutoff energy (keV) (if $E_c = 0$ there is no cutoff)
par3	rel_{refl}, reflection scaling factor (0 = no direct component, 1 corresponds to an isotropic source above disk)
par4	redshift, z
par5	abundance of elements heavier than He relative to the solar abundances
par6	iron abundance relative to that defined by **abund**
par7	inclination angle (degrees)
par8	disk temperature in K
par9	disk ionization parameter, $\xi = 4\pi\frac{F_{ion}}{n}$, where F_{ion} is the 5eV – 20keV irradiating flux, n is the density of the reflector; see Done et al., 1992, ApJ, 395, 275.
par10	e, power law dependence of emissivity. the emissivity $\propto R^e$
par11	inner radius (units of GM/c^2)
par12	outer radius (units of GM/c^2)
par13	internal model accuracy - points of spectrum per energy decade
norm	photon flux at 1 keV (photons/keV/cm^2/s) of the cutoff broken power-law only (no reflection) in the observed frame.

6.2.79 rnei, vrnei, vvrnei: non-equilibrium recombining collisional plasma

Non-equilibrium ionization collisional plasma model. This is a beta test version of a model for a recombining plasma where the plasma is assumed to have started in collisional equilibrium with the initial temperature given by the relevant input parameter. This model will only work for **xset neivers 3.0** and above. The default is 3.0.

The **vrnei** variant allows the user to set the abundances of the most common elements. The abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full

control over all the elemental abundances is available in the **vvrnei** model.

For the **rnei** model the parameters are:

par1	Plasma temperature (keV)
par2	Initial plasma temperature (keV)
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par4	Ionization timescale in units of s/cm^3.
par5	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vrnei** model, the parameters are:

par1	Plasma temperature (keV)
par2	Initial plasma temperature (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par15	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par16	Ionization timescale in units of s/cm^3.
par17	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvrnei** model, the parameters are:

par1	Plasma temperature (keV)
par2	Initial plasma temperature (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par32	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par33	Ionization timescale in units of s/cm^3.
par34	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2))\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.80 sedov, vsedov, vvsedov: sedov model, separate ion, electron temperatures

Sedov model model with separate ion and electron temperatures. This model is slow. `par1` provides a measure of the average energy per particle (ions+electrons) and is constant throughout the postshock flow in plane shock models (Borkowski et al., 2001, ApJ, 548, 820). `par2` should always be less than `par1`. If `par2` exceeds `par1` then their interpretations are switched (ie the larger of `par1` and `par2` is always the mean temperature). Additional references can be found under the help for the **equil** model. Several versions are available. To switch between them use the **xset** **neivers** command. The versions available are:

1.0	the version from xspec v11.1
1.1	as 1.0 but with updated ionization fractions using dielectronic recombination rates from Mazzotta et al (1998)
2.0	same ionization fractions as 1.1 but uses AtomDB v2 to calculate the resulting spectrum
3.0	ionization fractions and spectrum calculation uses AtomDB v3

Note that versions 1.x have no emission from Ar. The default is version 3.0. The **vsedov** variant allows the user to set the abundances for the more common elements. For versions 3 and above the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset** **APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Full control over all the elemental abundances is available in the **vvsedov** model.

For the **sedov** model the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni. Abundances are defined by the **abund** command
par4	Ionization age (s/cm^3) of the remnant (= electron density immediately behind the shock front multiplied by the age of the remnant).
par5	redshift
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

For the **vsedov** model, the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par15	Abundances for He, C, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command)
par16	Ionization age (s/cm^3) of the remnant (= electron density immediately behind the shock front multiplied by the age of the remnant).
par17	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

Finally, for the **vvsedov** model, the parameters are:

par1	Mean shock temperature (keV)
par2	Electron temperature immediately behind the shock front (keV)
par3	H abundance (should take values of 0 or 1 to switch on and off free-free continuum)
par4–par32	Abundances for all elements with $2 \leq Z \leq 30$ wrt Solar (defined by the **abund** command)
par33	Ionization age (s/cm^3) of the remnant (= electron density immediately behind the shock front multiplied by the age of the remnant).
par34	Redshift, z
norm	$(10^{-14}/(4\pi[D_A(1+z)]^2)) \int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^{-3}), and n_H is the hydrogen density (cm^{-3}).

6.2.81 sirf: self-irradiated funnel

The multi-blackbody "Self-IrRadiated Funnel" model is designed to model optically-thick outflow-dominated accretion. The basic idea is simple: you just assume a lot of matter, angular momentum and energy emerges in a limited volume. Momentum conservation leads to non-sphericity of the flow that has subsequently conical (funnel-like) shape. The model calculates temperature distribution at the funnel walls (taking into account irradiation by iterative process) and the outer photosphere. We also assume that inside the cone there is a deep pseudo-photosphere. Relativistic boosts are taken into account for high velocities. For a comprehensive description of the physical model, see: Abolmasov, P., Karpov, S. and Kotani, T. PASJ, 61, 2, 213.

par1	t_{in}, inner temperature (at the inner, inside-the-funnel photosphere).
par2	r_{in}, inner (inside-the-funnel photosphere) radius in "spherisation radius" units (the latter is defined as $3K\dot{M}/\Omega_f c$).
par3	r_{out}, outer photosphere radius in "spherisation radius" units.
par4	θ, half-opening angle of the cone.
par5	incl, inclination angle of the funnel. Affects mainly self-occultation and relativistic boost effects.
par6	v_α, velocity law exponent, $v \propto r^{v_\alpha}$.
par7	γ, adiabatic index. It affects the inner, hotter parts of the flow, therefore we set is to 4/3 by default.
par8	$\dot{m}$, mass ejection rate in Eddington (critical) units.
par9	irrad, number of iterations for irradiation.
norm	

6.2.82 slimbh: Stationary slim accretion disk

This is a relativistic model for X-ray continuum of stationary slim accretion disks around stallar mass black holes. This model is based on disk radial structure solutions presented in Sadowski et al. (2010) combined with vertical structure computed using TLUSTY code to make the best model for X-ray continuum for a wide range of luminosities up to Eddington limit. See also Straub et al. (2011).

Using the **xset** command to set SLIMBB_DEBUG to 1 will cause debug information to be written to a log file. The table file used by this model can be replaced by setting SLIMBB_DIR and SLIMBB_TABLE to the directory and name of the file, respectively.

The parameters are:

par1	M: mass of the black hole (in solar units)
par2	a: black hole spin parameter (dimensionless; 0¡a¡0.999)
par3	L: luminosity (total disk luminosity in Eddington units)
par4	alpha: alpha-viscosity
par5	i: inclination (in degrees; 0¡=i¡=85)
par6	D: distance to the source (in kpc)
par7	f: hardening factor; negative value means to use TLUSTY spectra
par8	lflag: a flag to switch on/off the effect of limb-darkening (not allowed to be free). If positive, disk emission is limb-darkened; if ¡=0, disk emission is assumed to be isotropic
par9	vflag: switch on/off surface profile; if positive, raytracing is done from the actual photosphere; if ¡=0, spectrum is raytraced from the equatorial plane ignoring the height profile of the disk
par10	normalization (should be fixed to 1)

6.2.83 smaug: optically-thin, spherically-symmetric thermal plasma.

This model performs an analytical deprojection of an extended, optically-thin and spherically-symmetric source. A thorough description of the model is given in Pizzolato et al. (ApJ 592, 62, 2003). In this model the 3D distributions of hydrogen, metals and temperature throughout the source are given specific functional forms dependent on a number of parameters, whose values are determined by the fitting procedure. The user has to extract the spectra in annular sectors, concentric about the emission peak. The inner boundary (in arcmin), the outer by the fitting procedure. The user has to extract the spectra in annular sectors, concentric about the emission peak. Three additional XFLTnnnn keywords must be added (e.g. with the ftools fkeypar). These should take the values "inner: x", "outer: y", "width: z" where x, y, z are the inner boundary (in arcmin), the outer boundary (also in arcmin), and the width (in degrees), respectively, of each annular sector. Some parameters of **smaug** define the redshift and other options (see below). The other, "relevant" ones define the 3D distributions of hydrogen density, temperature and metal abundance, determined by a simultaneous fit of the spectra. The cosmological parameters can be set using the **cosmo** command.

par1	central temperature [keV]
par2	max difference of temperature [keV]
par3	exponent of the inner temperature
par4	radius of the inner temperature [Mpc]
par5	exponent of the middle temperature
par6	radius of the middle temperature [Mpc]
par7	exponent of the outer temperature
par8	radius of the outer temperature [Mpc]
par9	central hydrogen density [cm^{-3}]
par10	fraction of nH.cc relative to the 1st beta component
par11	exponent of the first beta component
par12	radius of the 1st beta component [Mpc]
par13	exponent of the 2nd beta component
par14	radius of the 2nd beta component [Mpc]
par15	central metallicity [solar units]
par16	exponent of the metal distribution
par17	radius of the metal distribution [Mpc]
par18	redshift of the source
par19	number of mesh-points of the dem summation grid
par20	cutoff radius for the calculation [Mpc]
par21	mode of spectral evaluation: 0 = calculate, 1 = interpolate, 2 = APEC interpolate
par22	type of plasma emission code, 1 = Raymond-Smith, 2 = Mekal, 3 = Meka, 4 = APEC
K	model normalisation (nH.cc squared [cm^{-6}])

Note that if the interactive chattiness level in XSPEC is set to a value >10, **smaug** also prints on screen the following quantities:

H_0	Hubble constant [km/s/Mpc]
q_0	deceleration parameter
L_0	cosmological constant
DA	source angular distance [Mpc]
DSET	dataset no. to which the quantities listed below are
IN	inner rim of the projected annular sector [Mpc]
OUT	outer rim of the projected annular sector [Mpc]
WID	width of the projected annular sector [deg]
EVOL	emitting volume within the integration radius cut-off [Mpc^3]
EINT	emission integral within the integration radius cut-off [Mpc^3/cm^6]. If nH.cc is frozen to 1, the actual EI is obtained by multiplying this figure by the square root of the model normalisation

6.2.84 snapec: galaxy cluster spectrum using SN yields

This model calculates the spectrum for a galaxy cluster using the **apec** model with relative abundances based on SN yields. The relative amounts of SNIa and SNII can be set and there are a wide variety of different SN yield calculations available. It is straightforward to add the results of new yield calculations as they become available in the literature. This model is described in Bulbul, Smith and Loewenstein 2012 (ApJ 753, 54). All the standard **apec xset** options can be used.

The parameters are:

par1	plasma temperature, keV
par2	number of SNe (in units of 10^9)
par3	percentage of SNIa
par4	SNIa yield model
par5	SNII yield model
par6	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

The yield models available are as followed:

Number	Model	Reference
1	SN Type Ia, W7	R1, Table 2, Z=1.0
2	SN Type II, m_u=13	R1, Table 1
3	SN Type II, m_u=15	R1, Table 1
4	SN Type II, m_u=18	R1, Table 1
5	SN Type II, m_u=20	R1, Table 1
6	SN Type II, m_u=25	R1, Table 1
7	SN Type II, m_u=40	R1, Table 1
8	SN Type II, m_u=70	R1, Table 1
9	SN Type II, m_u=average (0-50)	R1, Table 2
10	SN Type II, m_u=average (0-70)	R1, Table 2
11	SN Type II, m_u=11, Z=1.0	R2, Table 5a
12	SN Type II, m_u=12, Z=1.0	R2, Table 5a
13	SN Type II, m_u=13, Z=1.0	R2, Table 5a
14	SN Type II, m_u=15, Z=1.0	R2, Table 5a
15	SN Type II, m_u=18, Z=1.0	R2, Table 5a
16	SN Type II, m_u=19, Z=1.0	R2, Table 5a
17	SN Type II, m_u=20, Z=1.0	R2, Table 5a
18	SN Type II, m_u=22, Z=1.0	R2, Table 5a
19	SN Type II, m_u=25, Z=1.0	R2, Table 5a
20	SN Type II, m_u=30A, Z=1.0	R2, Table 5b

21	SN Type II, m_u=30B, Z=1.0	R2, Table 5b
22	SN Type II, m_u=35A, Z=1.0	R2, Table 5b
23	SN Type II, m_u=35B, Z=1.0	R2, Table 5b
24	SN Type II, m_u=35C, Z=1.0	R2, Table 5b
25	SN Type II, m_u=40A, Z=1.0	R2, Table 5b
26	SN Type II, m_u=40B, Z=1.0	R2, Table 5b
27	SN Type II, m_u=40C, Z=1.0	R2, Table 5b
28	SN Type II, m_u=12, Z=0.1	R2, Table 10a (P12A)
29	SN Type II, m_u=13, Z=0.1	R2, Table 10a (P13A)
30	SN Type II, m_u=15, Z=0.1	R2, Table 10a (P15A)
31	SN Type II, m_u=18, Z=0.1	R2, Table 10a (P18A)
32	SN Type II, m_u=20, Z=0.1	R2, Table 10a (P20A)
33	SN Type II, m_u=22, Z=0.1	R2, Table 10a (P22A)
34	SN Type II, m_u=25, Z=0.1	R2, Table 10a (P25A)
35	SN Type II, m_u=30A, Z=0.1	R2, Table 10b (P30A)
36	SN Type II, m_u=30B, Z=0.1	R2, Table 10b (P30B)
37	SN Type II, m_u=35A, Z=0.1	R2, Table 10b (P35A)
38	SN Type II, m_u=35B, Z=0.1	R2, Table 10b (P35B)
39	SN Type II, m_u=35C, Z=0.1	R2, Table 10b (P35C)
40	SN Type II, m_u=40A, Z=0.1	R2, Table 10b (P40A)
41	SN Type II, m_u=40B, Z=0.1	R2, Table 10b (P40B)
42	SN Type II, m_u=40C, Z=0.1	R2, Table 10b (P40C)
43	SN Type II, m_u=11, Z=0.01	R2, Table 12a (P12A)
44	SN Type II, m_u=13, Z=0.01	R2, Table 12a (P13A)
45	SN Type II, m_u=15, Z=0.01	R2, Table 12a (P15A)
46	SN Type II, m_u=18, Z=0.01	R2, Table 12a (P18A)
47	SN Type II, m_u=20, Z=0.01	R2, Table 12a (P20A)
48	SN Type II, m_u=22, Z=0.01	R2, Table 12a (P22A)
49	SN Type II, m_u=25, Z=0.01	R2, Table 12a (P25A)
50	SN Type II, m_u=30A, Z=0.01	R2, Table 12b (T30A)
51	SN Type II, m_u=30B, Z=0.01	R2, Table 12b (T30B)
52	SN Type II, m_u=35A, Z=0.01	R2, Table 12b (T35A)
53	SN Type II, m_u=35B, Z=0.01	R2, Table 12b (T35B)
54	SN Type II, m_u=35C, Z=0.01	R2, Table 12b (T35C)
55	SN Type II, m_u=40A, Z=0.01	R2, Table 12b (T40A)
56	SN Type II, m_u=40B, Z=0.01	R2, Table 12b (T40B)
57	SN Type II, m_u=40C, Z=0.01	R2, Table 12b (T40C)
58	SN Type II, m_u=12, Z=0.0001	R2, Table 14a (U12A)
59	SN Type II, m_u=13, Z=0.0001	R2, Table 14a (U13A)
60	SN Type II, m_u=15, Z=0.0001	R2, Table 14a (U15A)
61	SN Type II, m_u=18, Z=0.0001	R2, Table 14a (U18A)
62	SN Type II, m_u=20, Z=0.0001	R2, Table 14a (U20A)
63	SN Type II, m_u=22, Z=0.0001	R2, Table 14a (U22A)
64	SN Type II, m_u=25, Z=0.0001	R2, Table 14a (U25A)
65	SN Type II, m_u=12, Z=0.0001	R2, Table 14b (U30A)
66	SN Type II, m_u=13, Z=0.0001	R2, Table 14b (U30B)

67	SN Type II, m_u=15, Z=0.0001	R2, Table 14b (U35A)
68	SN Type II, m_u=18, Z=0.0001	R2, Table 14b (U35B)
69	SN Type II, m_u=20, Z=0.0001	R2, Table 14b (U35C)
70	SN Type II, m_u=22, Z=0.0001	R2, Table 14b (U40A)
71	SN Type II, m_u=25, Z=0.0001	R2, Table 14b (U40B)
72	SN Type II, m_u=25, Z=0.0001	R2, Table 14b (U40C)
73	SN Type II, m_u=12, Z=0	R2, Table 16a (Z12A)
74	SN Type II, m_u=13, Z=0	R2, Table 16a (Z13A)
75	SN Type II, m_u=15A, Z=0	R2, Table 16a (Z15A)
76	SN Type II, m_u=18, Z=0	R2, Table 16a (Z18A)
77	SN Type II, m_u=20, Z=0	R2, Table 16a (Z20A)
78	SN Type II, m_u=22, Z=0	R2, Table 16a (Z22A)
79	SN Type II, m_u=25, Z=0	R2, Table 16a (Z25A)
80	SN Type II, m_u=15B, Z=0	R2, Table 16a (Z15B)
81	SN Type II, m_u=30A, Z=0	R2, Table 16b (Z30A)
82	SN Type II, m_u=30B, Z=0	R2, Table 16b (Z30B)
83	SN Type II, m_u=10-50	R3, Table 3
84	SN Type I, W7	R3, Table 3
85	SN Type I, W70	R3, Table 3
86	SN Type I, WDD1	R3, Table 3
87	SN Type I, WDD2	R3, Table 3
88	SN Type I, WDD3	R3, Table 3
89	SN Type I, CDD1	R3, Table 3
90	SN Type I, CDD3	R3, Table 3
91	SN Type II, m_u=13, Z=0	R4, Table 2
92	SN Type II, m_u=15, Z=0	R4, Table 2
93	SN Type II, m_u=18, Z=0	R4, Table 2
94	SN Type II, m_u=20, Z=0	R4, Table 2
95	SN Type II, m_u=25, Z=0	R4, Table 2
96	SN Type II, m_u=30, Z=0	R4, Table 2
97	SN Type II, m_u=40, Z=0	R4, Table 2
98	SN Type II, m_u=13, Z=0.001	R4, Table 2
99	SN Type II, m_u=15, Z=0.001	R4, Table 2
100	SN Type II, m_u=18, Z=0.001	R4, Table 2
101	SN Type II, m_u=20, Z=0.001	R4, Table 2
102	SN Type II, m_u=25, Z=0.001	R4, Table 2
103	SN Type II, m_u=30, Z=0.001	R4, Table 2
104	SN Type II, m_u=40, Z=0.001	R4, Table 2
105	SN Type II, m_u=13, Z=0.004	R4, Table 2
106	SN Type II, m_u=15, Z=0.004	R4, Table 2
107	SN Type II, m_u=18, Z=0.004	R4, Table 2
108	SN Type II, m_u=20, Z=0.004	R4, Table 2
109	SN Type II, m_u=25, Z=0.004	R4, Table 2
110	SN Type II, m_u=30, Z=0.004	R4, Table 2
111	SN Type II, m_u=40, Z=0.004	R4, Table 2
112	SN Type II, m_u=13, Z=0.02	R4, Table 2

113	SN Type II, m_u=15, Z=0.02	R4, Table 2
114	SN Type II, m_u=18, Z=0.02	R4, Table 2
115	SN Type II, m_u=20, Z=0.02	R4, Table 2
116	SN Type II, m_u=25, Z=0.02	R4, Table 2
117	SN Type II, m_u=30, Z=0.02	R4, Table 2
118	SN Type II, m_u=40, Z=0.02	R4, Table 2
119	SN Type Ia, W7	R5
120	SN Type Ia, cDEF	R5
121	SN Type Ia, CDDT	R5
122	SN Type Ia, ODDT	R5

The references are

R1 Tsujimoto, T., et al. 1995, MNRAS, 277, 945
R2 Woosley, S.E. & Weaver, T.A. 1995, ApJS, 101, 181
R3 Iwamoto, K., et al. 1999, ApJ, 125, 439
R4 Nomoto, K., et al. 2006, Nuclear Physics A, 777, 424
R5 Maeda, K., et al. 2010, ApJ, 708, 1703

6.2.85 srcut: synchrotron spectrum, cutoff power law

srcut describes the synchrotron spectrum from an exponentially cut off power-law distribution of electrons in a homogeneous magnetic field. This spectrum is itself a power-law, rolling off more slowly than exponential in photon energies. Though more realistic than a power-law, it is highly oversimplified, but does give the maximally curved physically plausible spectrum and can be used to set limits on maximum accelerated-electron energies even in remnants whose X-rays are thermal. See Reynolds, S.P. & Keohane, J.W. 1999, ApJ, 525, 368 and Reynolds, S.P., 1998 ApJ 493, 357. Note that the radio spectral index and flux can be obtained from Green's Catalogue at `http://www.mrao.cam.ac.uk/surveys/snrs` for galactic SNRs.

par1	alpha: radio spectral index
par2	break Hz: approximately the frequency at which the flux has dropped by a factor of 10 from a straight power law.
norm	1 GHz flux (Jy)

6.2.86 sresc: synchrotron spectrum, cut off by particle escape

The synchrotron spectrum from an electron distribution limited by particle escape above some energy. The electrons are shock-accelerated in a Sedov blast wave encountering a constant-density medium containing a uniform magnetic field. The model includes variations in electron acceleration efficiency with shock obliquity, and post-shock radiative and adiabatic losses, as described in Reynolds, S.P., ApJ 493, 357 1998. This is a highly specific, detailed model for a fairly narrow set of conditions. See also Reynolds, S.P., ApJL 459, L13 1996. Note that the radio spectral index and flux can be obtained from Green's Catalogue at http://www.mrao.cam.ac.uk/surveys/snrs for galactic SNRs.

par1	alpha: radio spectral index (flux proportional to frequency $f^{-\alpha}$)
par2	break Hz: approximately the frequency at which the flux has dropped by a factor of 6 below a straight power law extrapolation from radio frequencies. This frequency is 5.3 times the peak frequency radiated by electrons with energy E_{m3} in a magnetic field of $4B_1$, in the notation of Reynolds (1998), Eq. (19).
norm	1 GHz flux (Jy)

6.2.87 step: step function convolved with gaussian

A step function convolved with a gaussian.

$$A(E) = \frac{K}{2}[1 - \mathrm{erf}(\frac{(E - E_s)}{\sqrt{2}\sigma})]$$

par1	E_s, start energy (keV)
par2	σ, gaussian sigma (keV)
norm	K, step amplitude

6.2.88 tapec, vtapec, vvtapec: APEC emission spectrum with separate continuum and line temperatures

An emission spectrum from collisionally-ionized diffuse gas calculated from the AtomDB atomic database. More information can be found at http://atomdb.org/ which should be consulted by anyone running this model. This version of the model allows different temperatures for the continuum and lines. This

default version number can be changed by modifying the ATOMDB_VERSION string in your `Xspec.init` file.

By default this model reads atomic physics continuum and line data from the files apec_v[version]_coco.fits and apec_v[version]_line.fits in the `$HEADAS/../spectral/modelData directory`. Different files can be specified by using the command **xset APECROOT**. There are several options. APECROOT can be set to a version number (eg 1.10, 1.2.0, 1.3.1, 2.0.1, 2.0.2, 3.0.2, 3.0.3). In this case the value of APECROOT will be used to replace the default version number in the name of the standard files and the resulting files will be assumed to be in the `modelData` directory. Alternatively, a filename root (eg `apec_v1.2.0`) can be given. This root will be used as a prefix for the `_coco.fits` and `_line.fits` files. Finally, if neither of these work then the model will assume that the APECROOT value gives the complete directory path, e.g.

```
XSPEC12> xset APECROOT /foo/bar/apec_v1.2.0
```

will use the input files

```
/foo/bar/apec_v1.2.0_coco.fits
/foo/bar/apec_v1.2.0_line.fits.
```

Thermal broadening of lines can be included by using: **xset APECTHERMAL yes**. This runs significantly slower than the option without thermal broadening so you should only use this when necessary. Velocity broadening of lines can be included by using: **xset APECVELOCITY** `<velocity>`, where `<velocity>` is sigma in km/s. This is added in Gaussian quadrature with any thermal broadening in use.

A continuum-only spectrum can be obtained by using **xset APECNOLINES yes**. This will turn off lines for all models using the APEC files. To get a line-free version of a single APEC model use **nlapec**.

The **tapec** model uses abundances set by the **abund** command. The **vtapec** and **vvtapec** variants allow the user to set the abundance using additional parameters. For **tapec** and **vtapec** the abundances of the trace elements (ie Li, Be, B, F, Na, P, Cl, K, Sc, Ti, V, Cr, Mn, Co, Cu, Zn) can be set using **xset APEC_TRACE_ABUND**. These trace element abundances can be set either to the abundance of one of the main elements or to a numerical value (relative to Solar). For instance,

```
XSPEC12> xset APEC_TRACE_ABUND Fe
```

sets trace element abundances to that of iron while

```
XSPEC12> xset APEC_TRACE_ABUND 1.0
```

sets them to Solar. The default value for APEC_TRACE_ABUND is 1.0. Note that this means that the **tapec** and **vtapec** models will show emission lines even if the abundance parameters are set to zero.

For the **tapec** model the parameters are:

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3	Metal abundances (He fixed at cosmic). The elements included are C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni. Relative abundances are set by the abund command. The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par4	Redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vtapec** variant the parameters are as follows.

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3–par15	Abundances for He, C, N, O, Ne, Mg,Al, Si, S, Ar, Ca, Fe, Ni wrt Solar (defined by the **abund** command). The trace element abundances are from **xset** **APEC_TRACE_ABUND**, the default is 1.0.
par16	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

For the **vvtapec** variant the parameters are as follows.

par1=kT	continuum temperature, keV
par2=kTi	line temperature, keV
par3–par32	Abundances for H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn wrt Solar (defined by the **abund** command)
Par33	redshift, z
norm	$\frac{10^{-14}}{4\pi[D_A(1+z)]^2}\int n_e n_H dV$, where D_A is the angular diameter distance to the source (cm), n_e and n_H are the electron and H densities (cm^{-3}), respectively

6.2.89 voigt: Voigt line profile

A simple Voigt line profile. The Voigt profile is the convolution of a Gaussian and a Lorentzian. This model uses a pseudo-Voigt approximation which is good to 1% and is due to Ida, Ando & Toraya 2000, Journal of Applied Crystallography 33 (6), 1311.

The parameters are:

par1	E_l, line energy in keV
par2	σ, gaussian line width in keV
par3	γ, lorentzian FWHM line width in keV
Norm	K, total photons/cm^{-2}/s in the line

6.3 Multiplicative Model Components

6.3.1 absori: ionized absorber

An ionized absorber based on that of Done et al. (1992, ApJ 395, 275) and developed by Magdziarz & Zdziarski. See also Zdziarski et al. (1995, ApJ 438, L63). Photoionization rates are from Reilman & Manson (1979, ApJS 40, 815), who employ the Hartree-Slater approximation (accurate to about 5%), and recombination rates are from Shull & Steenburg (1982, ApJS 48, 95). The cross sections are extrapolated with E^{-3} above 5 keV. The abundances are set up by the command abund. Send questions or comments to **aaz@camk.edu.pl**

par1	power-law photon index.
par2	Hydrogen column in units of $10^{22} cm^{-2}$
par3	Absorber temperature in K.
par4	Absorber ionization state (L/nR2), see Done et al. (1992)
par5=z	redshift.
par6	Iron abundance relative to that defined by the command abund

6.3.2 acisabs: Chandra ACIS q.e. decay

This model accounts for the decay in the ACIS quantum efficiency most likely caused by molecular contamination of the ACIS filters. The user needs to supply the number of days between Chandra launch and observation. The **acisabs** parameters related to the composition of the hydrocarbon and the rate of decay should be frozen and not modified. The present version of **acisabs** is to be used for the analysis of bare ACIS I and ACIS S data. For the present version of **acisabs** one must use the standard qe file vN0003 instead of the optional vN0004 file.

Because of the present large uncertainty in the ACIS gain at energies below 350eV we recommend that events in the 0-350eV range be ignored in the spectral analysis until the gain issue is resolved.

acisabs calculates the mass absorption coefficients of the contaminant from atomic scattering factor files provided at

http://henke.lbl.gov/optical_constants/asf.html

par1	Days between Chandra launch and ACIS observation
par2	Slope of linear quantum efficiency decay
par3	Offset of linear quantum efficiency decay
par4	Number of carbon atoms in hydrocarbon
par5	Number of hydrogen atoms in hydrocarbon
par6	Number of oxygen atoms in hydrocarbon
par7	Number of nitrogen atoms in hydrocarbon

6.3.3 cabs: Optically-thin Compton scattering.

Optically-thin Compton scattering.

$$M(E) = exp(-\eta_H \sigma_T(E))$$

where $\sigma_T(E)$ is the Thomson cross-section with Klein-Nishina corrections at high energies. Note that this model does not do frequency downshifting so is only valid for scattering out of the beam.

par1=η_H	hydrogen column (in units of 10^{22} atoms cm^{-2})

6.3.4 constant: energy-independent factor

An energy-independent multiplicative factor.

par1 = factor

6.3.5 cyclabs: absorption line, cyclotron

A cyclotron absorption line as used in pulsar spectra. See Mihara *et al.*, Nature, 1990 or Makishima *et al.* PASJ, 1990.

$$M(E) = exp\left[-D_f \frac{(W_f E/E_{cycl})^2}{(E-E_{cycl})^2 + W_f^2} + D_{2h} \frac{(W_{2h} E/2E_{cycl})^2}{(E-2E_{cycl})^2 + W_{2h}^2}\right]$$

par1=D_f	depth of the fundamental
par2=E_{cycl}	cyclotron energy
par3=W_f	width of the fundamental
par4=D_{2h}	depth 2nd harmonic
par5=W_{2h}	width of the 2nd harmonic

6.3.6 dust: dust scattering

A modification of a spectrum due to scattering off dust on the line-of-sight. The model assumes that the scattered flux goes into a uniform disk whose size has a 1/E dependence and whose total flux has a $1/E^2$ dependence.

par1	scattering fraction at 1 keV
par2	size of halo at 1 keV in units of the detector beamsize

6.3.7 edge, zedge: absorption edge

The **edge** model is absorption edge, given by

$$M(E) = \begin{array}{ll} 1 & E \leq E_c \\ exp\left[-D(E/E_c)^{-3}\right] & E \geq E_c \end{array}$$

where:

par1=E_c	threshold energy
par2=D	absorption depth at the threshold

The **zedge** model given by

$$M(E) = \begin{array}{ll} 1 & E < E_c \\ exp\left(-D\left[E(1+z)/E_c\right]^3\right) & E > E_c \end{array}$$

allows a redshift z where:

par1=E_c	threshold energy
par2=D	absorption depth at the threshold
par3=z	redshift

6.3.8 etable: exponential tabular model

An exponential table model. The filename to be used should be given immediately after etable in the model command. For example:

```
XSPEC12> model etable{mymod.mod}
```

uses `mymod.mod` as the input for the model. XSPEC will multiply the contents of the model by -1 then take the exponential *i.e.* this model is for calculating absorption functions. For specifications of the table model file, see the OGIP memo 92-009 on the FITS file format for table model files (available on the WWW or by anonymous ftp from ftp://legacy.gsfc.nasa.gov/caldb/docs/memos).

The model value is assumed unity outside the tabulated energies unless the keywords LOELIMIT and HIELIMIT are given in the primary extension of the FITS file. In the latter case the value given in LOELIMIT will be used for energies below those tabulated and HIELIMIT for energies above those tabulated.

6.3.9 expabs: exponential roll-off at low E

A low-energy exponential rolloff.

$$M(E) = exp(-E_c/E)$$

where:

par1=E_c	e-folding energy for the absorption

6.3.10 expfac: exponential modification

An exponential modification of a spectrum.

$$M(E) = \begin{array}{ll} 1 + Aexp(-fE) & E > E_c \\ 1 & E < E_c \end{array}$$

where:

par1=A	amplitude of effect
par2=f	exponential factor
par3=E_c	start energy of modification

6.3.11 gabs: gaussian absorption line

$$M(E) = exp\left(-\left(par3/(\sqrt{2\pi}par2)\right) exp\left(-.5\left((E-par1)/par2\right)^2\right)\right)$$

where:

par1 = line energy in keV.

par2 = line width (sigma) in keV.

par3 = line depth. The optical depth at line center is par3/par2/$\sqrt{2\pi}$.

6.3.12 heilin: Voigt absorption profiles for He I series

This model calculates the Voigt absorption profiles for the He I series.

par1=nHeI	He I column density (10^{22} atoms/cm^2)
par2=b	b value (km/s)
par3=z	Redshift

6.3.13 highecut, zhighect: high-energy cutoff

A high energy cutoff.

$$M(E) = \begin{array}{ll} exp\left[(E_c - E)/E_f\right] & E \geq E_c \\ 1.0 & E \leq E_c \end{array}$$

where

par1=E_c	cutoff energy in keV
par2=E_f	e-folding energy in keV

The redshifted version **zhighect** has

$$M(E) = \begin{array}{ll} exp\left[(E_c - E[1+z])/E_f\right] & E \geq E_c \\ 1.0 & E \leq E_c \end{array}$$

where

par1=E_c	cutoff energy in keV
par2=E_f	e-folding energy in keV
par3=z	redshift

6.3.14 hrefl: reflection model

A simple multiplicative reflection model due to Tahir Yaqoob. This model gives the reflected X-ray spectrum from a cold, optically thick, circular slab with inner and outer radii (Ri & Ro, respectively) illuminated by a point source a height H above the center of the slab. The main difference between this model and other reflection models is that analytic approximations are used for the Chandrasekar H functions (and their integrals) and ELASTIC SCATTERING is assumed (see Basko 1978, ApJ, 223, 268). The elastic-scattering approximation means that the model is ONLY VALID UP TO $\approx$ 15 keV in the source frame. Future enhancements will include fudge factors that will allow extension up to 100 keV. The fact that no integration is involved at any point makes the routine very fast and particularly suitable for generating error contours, especially when fitting a large number of data channels. The model is multiplicative, and so can be used with ANY incident continuum.

Parameters are as follows:

par1	minimum angle (degrees) between source photons incident on the slab and the slab normal [$= tan^{-1}(R_i/H)$]
par2	maximum angle (degrees) between source photons incident on the slab and the slab normal [$= tan^{-1}(R_o/H)$]
par3	Angle (degrees) between the observer's line of sight and the slab normal.
par4	Iron abundance relative to Solar
par5	Iron K-edge energy
par6	Fraction of the direct flux seen by the observer
par7	Normalization of the reflected continuum
par8	redshift

Suppose the incident photon spectrum is N(E) photons $cm^{-2}s^{-1}keV^{-1}$ and that the incident continuum is steady in time, and suppose further that the reflected continuum from the slab is R(E). When you multiply the incident spectrum with **hrefl**, what you actually get is the following:

$$M(E) = par6 * N(E) + par7 * R(E)$$

Thus, the actual physical situation described above corresponds to

par6 = 1.0
par7 = 1.0

You may decide to float par6 and/or par7. In that case, you must decide what the best-fitting values of these parameters mean physically for your case. It may imply time-lags between the direct and reflected components, different source and/or disk geometries to those assumed, or something else.

6.3.15 ismabs: A high resolution ISM absorption model with variable columns for individual ions

This is an X-ray photoabsorption model for the interstellar medium that takes into account both neutral and ionized species from H, He, N, O, Ne, Mg, Si, S, Ar, Ca, Fe, Ni and Zn. This model has been developed by Efrain Gatuzz in collaboration with Javier Garcia, Tim Kallman, Claudio Mendoza, and Tom Gorczyca. A complete description of the science behind the model is described in Gatuzz et al. (2014b).

Note that the He I column density is not included as a free parameter in the model, due to an inherent degeneracy between the relative columns of H, He I, He II. Instead, the He I column is assumed to be one tenth that of the H column.

The sources of the cross-sections are as follows:

- Neutral states of Si, S, Ar and Ca from Verner et al. (1995)
- Singly and doubly ionized states of Si, S, Ar and Ca from Witthoeft et al. (2009) and Witthoeft et al. (2011)
- Neutral, singly and doubly ionized states of N from Garcia et al. (2009)
- Neutral states of O from Gorczyca et al. (2013)
- Singly and doubly ionized states of O from Garcia et al. (2005), including corrections applied by Gatuzz et al. (2013)
- Neutral state of Ne from Gorczyca et al. (2000)

- Singly and doubly ionized states of Ne from Gorczyca et al. (2005)

- For the Fe-L edge region we use the measurement of metallic iron by Kortright & Kim (2000)

- Neutral, singly and doubly ionized states of Mg from Hasoglu et al. (2014).

The parameters are:

par1	H column (in units of 10^{22} cm^{-}2)
par2	He II column (in units of 10^{16} cm^{-}2)
par3	C I column (in units of 10^{16} cm^{-}2)
par4	C II column (in units of 10^{16} cm^{-}2)
par5	C III column (in units of 10^{16} cm^{-}2)
par6	N I column (in units of 10^{16} cm^{-}2)
par7	N II column (in units of 10^{16} cm^{-}2)
par8	N III column (in units of 10^{16} cm^{-}2)
par9	O I column (in units of 10^{16} cm^{-}2)
par10	O II column (in units of 10^{16} cm^{-}2)
par11	O III column (in units of 10^{16} cm^{-}2)
par12	Ne I column (in units of 10^{16} cm^{-}2)
par13	Ne II column (in units of 10^{16} cm^{-}2)
par14	Ne III column (in units of 10^{16} cm^{-}2)
par15	Mg I column (in units of 10^{16} cm^{-}2)
par16	Mg II column (in units of 10^{16} cm^{-}2)
par17	Mg III column (in units of 10^{16} cm^{-}2)
par18	Si I column (in units of 10^{16} cm^{-}2)
par19	Si II column (in units of 10^{16} cm^{-}2)
par20	Si III column (in units of 10^{16} cm^{-}2)
par21	S I column (in units of 10^{16} cm^{-}2)
par22	S II column (in units of 10^{16} cm^{-}2)
par23	S III column (in units of 10^{16} cm^{-}2)
par24	Ar I column (in units of 10^{16} cm^{-}2)
par25	Ar II column (in units of 10^{16} cm^{-}2)
par26	Ar III column (in units of 10^{16} cm^{-}2)
par27	Ca I column (in units of 10^{16} cm^{-}2)
par28	Ca II column (in units of 10^{16} cm^{-}2)
par29	Ca III column (in units of 10^{16} cm^{-}2)
par30	Fe column (in units of 10^{16} cm^{-}2)
par31	Redshift, z

6.3.16 lyman: Voigt absorption profiles for H I or He II Lyman series

This model calculates the Voigt absorption profiles for the H I and He II Lyman series.

par1	n: H I or He II column density (10^{22} atoms/cm^2)
par2	b: b value (km/s)
par3	z: Redshift
par4	ZA: Atomic number of species being calculated.

6.3.17 mtable: multiplicative tabular model

A multiplicative table model. The filename to be used should be given immediately after **mtable** in the **model** command. For example:

```
XSPEC12>model mtable{mymod.mod}*powerlaw
```

uses `mymod.mod` as the input for the model. For specifications of the table model file, see the OGIP memo 92-009 on the FITS file format for table model files (available on the WWW or by anonymous ftp from ftp://legacy.gsfc.nasa.gov/caldb/docs/memos. A sample multiplicative table model file is `testpcfabs.mod` in `$HEADAS/../ftools/spectral/session`.

The model value is assumed unity outside the tabulated energies unless the keywords LOELIMIT and HIELIMIT are given in the primary extension of the FITS file. In the latter case the value given in LOELIMIT will be used for energies below those tabulated and HIELIMIT for energies above those tabulated.

6.3.18 notch: absorption line, notch

A notch line absorption. This is model is equivalent to a very saturated absorption line.

$$M(E) = \begin{array}{ll} (1-f) & E_L - W/2 < E < E_L + W/2 \\ 1 & \text{elsewhere} \end{array}$$

where

par1=E_L	line energy (keV)
par2=W	line width (keV)
par3=f	covering fraction

6.3.19 pcfabs, zpcfabs: partial covering fraction absorption

A partial covering fraction absorption. The relative abundances are set by the **abund** command.

$$M(E) = f \exp[-\eta_H \sigma(E)] + (1 - f)$$

where $\sigma(E)$ is the photo-electric cross-section (NOT including Thomson scattering) (see **phabs**) and:

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=f	dimensionless covering fraction ($0 < f \leq 1$)

The redshifted variant **zpcfabs** is given by:

$$M(E) = f \exp\left[-\eta_H \sigma(E[1+z])\right] + 1 - f$$

where $\sigma(E)$ is the photo-electric cross-section (NOT including Thomson scattering) (see **phabs**). Relative abundances are as for **pcfabs**. Parameters are:

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=f	dimensionless covering fraction ($0 < f \leq 1$)
par3=z	redshift

6.3.20 phabs, vphabs, zphabs, zvphabs: photoelectric absorption

A photoelectric absorption using cross-sections set by the **xsect** command. The relative abundances are set by the **abund** command.

$$M(E) = \exp[-\eta_H \sigma(E)]$$

where $\sigma(E)$ is the photo-electric cross-section (NOT including Thomson scattering). Note that the default He cross-section changed in v11. The old version can be recovered using the command

```
XSPEC12>xsect obcm
```

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})

The redshifted variant, **zphabs**, uses the formula

$$M(E) = \exp[-\eta_H \sigma(E[1+z])]$$

and has parameters

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=z	redshift

The variants **vphabs**, **zvphabs** allow the user to set fixed abundance parameters with respect to the solar composition, as defined by the **abund** command. For **vphabs** (rest-frame) the parameters are

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2-par18	abundances for He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni wrt to Solar

While the corresponding redshifted variant **zvphabs** has parameters

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2-par18	abundances for He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni wrt to Solar (defined by the abund command)
par19=z	redshift

6.3.21 plabs: power law absorption

Absorption as a power-law in energy. Useful for things like dust.

$$M(E) = KE^{-\alpha}$$

par1=α	index
par2=K	coefficient

6.3.22 pwab: power-law distribution of neutral absorbers

An extension of partial covering fraction absorption into a power-law distribution of covering fraction as a function of column density, built from the **wabs** code. See Done & Magdziarz 1998 (MNRAS 298, 737) for details.

par1	minimum equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2	maximum equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par3	power law index for covering fraction

6.3.23 recorn: change correction norm for a spectrum

This model is a replacement for and improvement on the old xspec command recornrm. If a correction file is in use for a spectrum then its normalization can be fitted for using this model. The first parameter, which is not variable, is the spectrum number and the second the correction file normalization. The starting value of the second parameter should be set to the current value of the correction file norm (this can be independently set using the **cornorm** command).

Note that in order to fit the cornorm parameter, the USE_NUMERICAL_DIFFERENTIATION setting in the user's Xspec.init start-up file must be set to `true`. This causes XSPEC to use a slower full numerical differentiation algorithm when calculating parameter derivatives during a fit, and therefore is not recommended for general usage.

par1	specnum: spectrum number
par2	cornorm: correction file normalization

6.3.24 redden: interstellar extinction

IR/optical/UV extinction from Cardelli et al. (1989, ApJ, 345, 245). The transmission is set to unity shortward of the Lyman limit. This is incorrect

physically but does allow the model to be used in combination with an X-ray photoelectric absorption model such as **phabs**.

par1	E(B-V)

6.3.25 smedge: smeared edge

A smeared edge (Ebisawa PhD thesis, implemented by Frank Marshall).

$$M(E) = \begin{array}{ll} 1 & E < E_c \\ \exp\left[-f(E/E_c)^{\alpha}\right]\left[1-\exp((E_c-E)/W)\right] & E \geq E_c \end{array}$$

where:

par1=E_c	the threshold energy (keV)
par2=f	the maximum absorption factor at threshold
par3=α	index for photo-electric cross-section (normally -2.67)
par4=W	smearing width (keV)

6.3.26 spexpcut: super-exponential cutoff absorption

A high-energy super-exponential roll-off.

$$M(E) = \exp\left(-(E/E_c)^{\alpha}\right)$$

useful for fitting gamma-ray spectra of pulsars (see eg Nel & de Jager 1995), where:

par1=E_c	e-folding energy for the absorption
par2=α	exponent index

Caveat : the absorption for an energy bin is calculated as the arithmetic mean of the function value at the start and end energies of the bin. If the energy bins are large this can be inaccurate and the energies command should be used to define a finer energy grid on which to calculate the model.

6.3.27 spline: spline modification

A cubic spline modification.

par1	start x-value
par2	start y-value
par3	end y-value
par4	start dy/dx
par5	end dy/dx
par6	end x-value

6.3.28 SSSice: Einstein SSS ice absorption

The Einstein Observatory SSS ice absorption.

par1	ice thickness parameter

6.3.29 swind1: absorption by partially ionized material with large velocity shear

A model to fit the soft excess in AGN by partially ionized absorbing material with large velocity shear. It approximates this by using XSTAR kn5 photoionization absorption model grids (calculated assuming a microturbulent velocity of 100km/s), and then convolving this with Gaussian smearing. This is the model used by Gierlinski & Done 2006, Sobolewska & Done 2006 and Done et al 2006. It is an update (uses a newer version of XSTAR) of the original model of Gierlinski & Done 2004.

par1	column density (10^{22} cm^{-2})
par2	log(xi) where xi = L/nr^2
par3	sigma : Gaussian sigma for velocity smearing (v/c)
par4	redshift

6.3.30 tbabs, ztbabs, tbfeo, tbgas, tbgrain, tbpcf, tbvarabs, tbrel: ISM grain absorption

The Tuebingen-Boulder ISM absorption model. This model calculates the cross section for X-ray absorption by the ISM as the sum of the cross sections for X-ray absorption due to the gas-phase ISM, the grain-phase ISM, and the molecules in the ISM. In the grain-phase ISM, the effect of shielding by the grains is accounted for, but is extremely small. In the molecular contribution to the ISM cross section, only molecular hydrogen is considered. In the gas-phase ISM, the cross section is the sum of the photoionization cross sections of the different elements, weighted by abundance and taking into account depletion onto grains.

In addition to the updates to the photoionization cross sections, the gas-phase cross section differs from previous values as a result of updates to the ISM abundances. These updated abundances are available through the **abund** `wilm` command. Details of updates to the photoionization cross sections as well as to abundances can be found in Wilms, Allen and McCray (2000, ApJ 542, 914).

Two versions of this model are available depending on the setting of TBABSVERSION. The newer version (the default) dates from 2016 and includes high resolution edge structures for the K-edges of oxygen and neon and the L-edges of iron. The newer version is 25 times faster than the older version or the phabs model. To recover the older version use xset TBABSVERSION 1.

tbabs allows the user to vary just the hydrogen column.

par1=nH	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})

ztbabs is similar, but allows the user to set a fixed redshift parameter

par1=nH	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=z	redshift

tbfeo is similar, but allows the user to vary the oxygen and iron abundances

par1=nH	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=O	oxygen abundance relative to Solar
par3=Fe	iron abundance relative to Solar
par4=z	redshift

tbgas is similar to **ztbabs** but with no grain absorption

par1=nH	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=z	redshift

tbgrain allows the user to vary the molecular hydrogen column and the grain distribution parameters.

par1	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2	molecular hydrogen column (in units of 10^{22} atoms cm^{-2})
par3	grain density (in gm cm^{-3}
par4	grain minimum size (in mm)
par5	grain maximum size (in mm)
par6	power-law index of grain sizes

tbpcf is similar to **ztbabs** but a partial covering fraction parameter

par1=nH	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=pcf	partial covering fraction
par3=z	redshift

tbvarabs additionally allows the user to vary the elemental abundances and the redshift

par1	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2 -par18	abundance (relative to Solar) of He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni
par19	molecular hydrogen column (in units of 10^{22} atoms cm^{-2})
par20	grain density (in gm cm^{-3})
par21	grain minimum size (in mm)
par22	grain maximum size (in mm)
par23	power-law index of grain sizes
par24-par41	grain depletion fractions of He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni
par42	redshift

tbrel is similar to **tbvarabs** but the interpretation of the abundances is different in that this model also allows "negative" nH. Use this, e.g., if you want to measure changes in column relative to an average column. For example, you can fit a model such as tbrel*tbabs*continuum, where all values of tbabs are frozen at the average column of the source. If the fit results in a negative column

for nH during that observation, then the total column of the source is less than the average.

par1	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2 -par18	abundance (relative to Solar) of He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni
par19	molecular hydrogen column (in units of 10^{22} atoms cm^{-2})
par20	grain density (in gm cm^{-3})
par21	grain minimum size (in mm)
par22	grain maximum size (in mm)
par23	power-law index of grain sizes
par24-par41	grain depletion fractions of He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni
par42	redshift

6.3.31 uvred: interstellar extinction, Seaton Law

A UV reddening using Seaton's law (MNRAS 187, 75p). Valid from 1000-3704Å. The transmission is set to unity shortward of the Lyman limit. This is incorrect physically but does allow the model to be used in combination with an X-ray photoelectric absorption model such as **phabs**.

par1	E(B-V)

6.3.32 varabs, zvarabs: photoelectric absorption

A photoelectric absorption with variable abundances using cross-sections set by the **xsect** command. The column for each element is in units of the column in a solar abundance column of an equivalent hydrogen column density of 10^{22}cm^{-2}. The Solar abundance table used is set by the **abund** command. These models differ from the models **vphabs** and **zvphabs** only by the units in which the abundances are expressed (**vphabs** and **zvphabs** define these relative to the solar abundance, not in terms of column density).

par1-par18	equivalent columns for H, He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Co, Ni

The **zvarabs** variant allows the user to specify a (fixed) redshift, i.e. the parameters are:

par1-par18	equivalent columns for H, He, C, N, O, Ne, Na, Mg, Al, Si, S, Cl, Ar, Ca, Cr, Fe, Ni, Co
par19	redshift

6.3.33 wabs, zwabs: photoelectric absorption, Wisconsin cross-sections

A photo-electric absorption using Wisconsin (Morrison and McCammon; ApJ 270, 119) cross-sections.

$$M(E) = \exp[-\eta_H \sigma(E)]$$

where $\sigma(E)$ is the photo-electric cross-section (NOT including Thomson scattering). Note that this model uses the Anders & Ebihara relative abundances (1982, Geochimica et Cosmochimica Acta 46, 2363) regardless of the **abund** command.

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})

The **zwabs** variant allows the user to specify a (fixed) redshift parameter, and uses the corresponding formula

$$M(E) = \exp[-\eta_H \sigma(E[1+z])]$$

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=z	redshift

6.3.34 wndabs, zwndabs: photo-electric absorption, warm absorber

Photo-electric absorption from approximation to a warm absorber using Balucinska,-Church, and McCammon (ApJ 400, 699) cross-sections. Relative abundances are set by the **abund** command.

$$M(E) = \begin{array}{ll} 1 & E > E_w \\ \exp[-\eta_H \sigma(E)] & E \leq E_w \end{array}$$

where $\sigma(E)$ is the photo-electric cross-section (NOT including Thomson scattering) and

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=E_w	window energy (keV)

The **zwndabs** variant allows the user to specify a (fixed) redshift and uses the corresponding formula:

$$M(E) = \begin{array}{ll} 1 & E > E_w \\ \exp[-\eta_H \sigma(E[1+z])] & E \leq E_w \end{array}$$

with parameters:

par1=η_H	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2=E_w	window energy (keV)
par3=z	redshift

6.3.35 xion: reflected spectrum of photo-ionized accretion disk/ring

This model describes the reflected spectra of a photo-ionized accretion disk or a ring if one so chooses. The approach is similar to the one used for tables with stellar spectra. Namely, a large number of models are computed for a range of values of the spectral index, the incident X-ray flux, disk gravity, the thermal disk flux and iron abundance. Each model's output is an un-smeared reflected spectrum for 5 different inclination angles ranging from nearly pole-on to nearly face on, stored in a look-up table. The default geometry is that of a lamppost, with free parameters of the model being the height of the X-ray source above the disk, h_X, the dimensionless accretion rate through the disk, $\dot{m}$, the luminosity of the X-ray source, L_X, the inner and outer disk radii, and the spectral index. This defines the gravity parameter, the ratio of X-ray to thermal fluxes, etc., for each radius, which allows the use of a look-up table to approximate the reflected spectrum. This procedure is repeated for about 30 different radii. The total

disk spectrum is then obtained by integrating over the disk surface, including relativistic smearing of the spectrum for a non-rotating black hole (e.g., Fabian 1989).

In addition, the geometry of a central sphere (with power-law optically thin emissivity inside it) plus an outer cold disk, and the geometry of magnetic flares are available (par13 = 2 and 3, respectively). One can also turn off relativistic smearing to see what the local disk spectrum looks like (par12 = 2 in this case; otherwise leave it at 4). In addition, par11 = 1 produces reflected plus direct spectrum/direct; par11 =2 produces (incident + reflected)/incident [note that normalization of incident and direct are different because of solid angles covered by the disk; 2 should be used for magnetic flare model]; and par11 =3 produces reflected/incident. Abundance is controlled by par9 and varies between 1 and 4 at the present. A much more complete description of the model will be presented in Nayakshin et al. 2001 (currently available at http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=2001ApJ...546..406N\&db_key=AST\&data_type=HTML\&format=\&high=4230b2429423803))

par1	height of the source above the disk (in Schwarzschild radii)
par2	ratio of the X-ray source luminosity to that of the disk
par3	accretion rate (in Eddington units)
par4	$\cos i$ the inclination angle (1 = face-on)
par5	inner radius of the disk (in Schwarzschild radii)
par6	outer radius of the disk (in Schwarzschild radii)
par7	photon index of the source
par8	redshift z
par9	Fe abundance relative to Solar (which is defined as 3.16×10^{-5} by number relative to H)
par10	Exponential high energy cut-off energy for the source
par11	1 ⇒ (reflected+direct)/direct 2 ⇒ (reflected+incident)/incident 3 ⇒ reflected/incident
par12	2 ⇒ no relativistic smearing 4 ⇒ relativistic smearing
par13	1 ⇒ lamppost 2 ⇒ central hot sphere with outer cold disk 3 ⇒ magnetic flares above a cold disk

Note that setting par13 to 2 gives a central hot sphere with luminosity law $dL/dR = 4\pi R^2 R^{-10y}$. The inner radius of the sphere is 3 Schwarzschild radii

and the outer radius is equal to par1. Only the case with par5 $\geq$ par1 has been tested so far.

6.3.36 xscat: dust scattering

This model calculates the X-ray scattering cross section of a population of dust grains as a function of energy, given a specific dust grain position and an extraction region. As the grain position gets closer to the source of the X-rays for a constant extraction region, the cross section drops as more X-rays remain within the extraction circle. For constant dust grain position, the cross section decreases with increasing extraction region size because more photons remain within it.

A detailed description of the model can be found in Smith, Valencic & Corrales (2016).

The model parameters are as follows.

par1	= NH	interstellar hydrogen column density (10^{22} cm^{-2}).
par2	= $Xpos$	the relative position of the dust along the line of sight. Th between 0 for dust at the observer and 1 for dust at the so
par3	= $Rext$	the radius of the circular extraction region (arcsec).
par4	= $DustModel$	the dust model used (see Smith et al. 2016).

The third and fourth parameters must be frozen.

6.3.37 zbabs: EUV ISM attenuation

The ISM attenuation due to neutral H, neutral He and once ionized He. This is a modified version of the Rumph, et al. 1994 model, using the ismatten program by Pat Jelinsky, and allows the user to set the model redshift.

par1=η_H	H column density (10^{22} atoms/cm^2)
par2=η_{HeI}	HeI column density (10^{22} atoms/cm^2)
par3=η_{HeII}	HeII column density (10^{22} atoms/cm^2)
par4=z	redshift

6.3.38 zdust: extinction by dust grains

Extinction by dust grains from Pei (1992, ApJ 395, 130), suitable for IR, optical and UV energy bands, including the full energy ranges of the Swift UVOT and XMM-Newton OM detectors. Three models are included which characterize the extinction curves of (1) the Milky Way, (2) the LMC and (3) the SMC. The models can be modified by redshift and can therefore be applied to extragalactic sources. The transmission is set to unity shortward of 912 Angstroms in the rest frame of the dust. This is incorrect physically but does allow the model to be used in combination with an X-ray photoelectric absorption model such as **phabs**. Parameter 1 (method) describes which extinction curve (MW, LMC or SMC) will be constructed and should never be allowed to float during a fit. The extinction at V, A(V) = E(B-V) x Rv. Rv should typically remain frozen for a fit. Standard values for Rv are MW = 3.08, LMC = 3.16 and SMC = 2.93 (from table 2 of Pei 1992), although these may not be applicable to more distant dusty sources.

par1	method: 1 = Milky Way, 2 = LMC, 3 = SMC
par2	E(B-V): color excess
par3	Rv: ratio of total to selective extinction
par4	z: redshift

6.3.39 zigm: UV/Optical attenuation by the intergalactic medium

This multiplicative model computes the mean attenuation of the optical/UV spectrum of an object at redshift z at a random position on the sky due to intergalactic medium (IGM) clouds following either Madau (1995, ApJ 441, 18) or Meiksin (2006, MNRAS, 365, 807). The model calculates the mean expected attenuation due to resonant scattering by Lyman transitions and photoelectric absorption shortward of the Lyman limit. Attenuation by Helium and metals are not included in the Meiksin model, but are expected to be small. The total attenuation is set to zero for wavelengths less than 900 Angstroms. The user chooses whether to include attenuation due to photoelectric absorption.

par1	z: redshift
par2	model: use Madau(0) or Meiksin(1)
par3	include photoelectric absorption(1) or not(0)

6.3.40 zredden: redshifted version of redden

IR/optical/UV extinction from Cardelli et al. (1989, ApJ, 345, 245). The transmission is set to unity shortward of 900 Angstroms. This is incorrect physically but does allow the model to be used in combination with an X-ray photoelectric absorption model such as **phabs**.

par1	E(B-V)
par2	redshift

6.3.41 zsmdust: extinction by dust grains in starburst galaxies

Extinction by dust grains suited to starburst galaxies and the hosts of gamma ray bursts. The model can be applied over the IR, optical and UV energy bands, including the full energy ranges of the Swift UVOT and XMM-Newton OM detectors. The transmission is set to unity shortward of 912 Angstroms in the rest frame of the dust. This is incorrect physically but does allow the model to be used in combination with an X-ray photoelectric absorption model such as **phabs**. The extinction curve contains no spectral features and is characterized by a powerlaw slope over spectral wavelength. This model has been justified by e.g. Savaglio & Fall (2004, ApJ, 614, 293) because the apparent low metallicities within GRB hosts result in no significant spectral features within the extinction curve, unlike those found in local galaxies. The extinction at V, A(V) = E(B-V) x Rv. Standard values for Rv are Milky Way = 3.08, LMC = 3.16 and SMC = 2.93 (from table 2 of Pei 1992, ApJ, 395, 130), although these may not be applicable to more distant dusty sources.

par1	E(B-V): color excess
par2	ExtIndex: spectral index of the extinction curve
par3	Rv: ratio of total to selective extinction
par4	z: redshift

6.3.42 zvfeabs: photoelectric absorption with free Fe edge energy

Redshifted photoelectric absorption with all abundances tied to Solar except for iron. The Fe K edge energy is a free parameter.

par1	equivalent hydrogen column (in units of 10^{22} atoms cm^{-2})
par2	abundance relative to Solar
par3	iron abundance relative to Solar
par4	Fe K edge energy
par5	redshift z

6.3.43 zxipcf: partial covering absorption by partially ionized material

This model uses a grid of XSTAR photionized absorption models (calculated assuming a microturbulent velocity of 200km/s) for the absorption, then assumes that this only covers some fraction f of the source, while the remaining (1-f) of the spectrum is seen directly. This is the model used by Reeves et al (2008) 'On why the iron K-shell absorption in AGN is not the signature of the local warm-hot intergalactic medium', and may also be more generally applicable to the spectral complexity seen in Narrow Line Seyfert 1 AGN (Miller et al 2007, A&A, 463, 13).

par1	column density (10^{22} cm^{-2})
par2	log(xi) where xi=L/nr^2
par3	covering fraction
par4	redshift

6.4 Convolution Model Components

Convolution components apply a convolution operator to an input model calculated from another model. They differ from multiplicative components, which only apply a bin-wise multiplicative factor, by allowing transformations of the model across energy bins.

6.4.1 cflux: calculate flux

A convolution model to calculate the flux of other model components. For example:

```
cflux*phabs*(pow + gauss)
```

with the normalization of the power-law model fixed to a non-zero value gives the flux and error on the entire model.

```
phabs*cflux*(pow + gauss)
```

again with the normalization of the power-law fixed to a non-zero value gives the unabsorbed flux and error. Finally,

```
phabs*(pow + cflux*gauss)
```

with the normalizaton of the gaussian fixed to a non-zero value gives the flux and error on the gaussian component. Note that when the **cflux** model is used the normalization of one of the additive models **must** be fixed to a non-zero value. It is also important to ensure that the energy range over which the model is calculated (which is determined by the response matrix in use) covers the energy range for which the flux is calculated. If the model to which the **cflux** is applied integrates to zero then a divide-by-zero error will occur resulting in NaN values for the fit statistic.

Parameters are:

par1=Emin	Minimum energy over which flux is calculated.
par2=Emax	Maximum energy over which flux is calculated.
par3=lg10Flux	log (base 10) flux in erg/cm^2/s

6.4.2 clumin: calculate luminosity

A convolution model to calculate the luminosity of other model components. For example:

```
clumin*phabs*(pow + gauss)
```

with the normalization of the power-law model fixed to a non-zero value gives the luminosity and error on the entire model.

```
phabs*clumin*(pow + gauss)
```

again with the normalization of the power-law fixed to a non-zero value gives the unabsorbed luminosity and error. Finally,

```
phabs*(pow + clumin*gauss)
```

with the normalizaton of the gaussian fixed to a non-zero value gives the luminosity and error on the gaussian component. Note that when the **clumin** model is used the normalization of one of the additive models **must** be fixed to a non-zero value. It is also important to ensure that the energy range over which the model is calculated (which is determined by the response matrix in use) covers the energy range for which the luminosity is calculated. If the model to which the **clumin** is applied integrates to zero then a divide-by-zero error will occur resulting in NaN values for the fit statistic.

Parameters are:

par1=Emin	Minimum source frame energy over which luminosity is calculated.
par2=Emax	Maximum source frame energy over which luminosity is calculated.
par3=Redshift	Redshift of the source.
par4=lg10Lum	log (base 10) luminosity in units of erg/s.

6.4.3 cpflux: calculate photon flux

A convolution model to calculate the photon flux of other model components. For example :

```
cpflux*phabs*(pow + gauss)
```

with the normalization of the power-law model fixed to a non-zero value gives the photon flux and error on the entire model.

```
phabs*cpflux*(pow + gauss)
```

again with the normalization of the power-law fixed to a non-zero value gives the unabsorbed photon flux and error. Finally,

```
phabs*(pow + cpflux*gauss)
```

with the normalizaton of the gaussian fixed to a non-zero value gives the photon flux and error on the gaussian component. Note that when the **cpflux** model is used the normalization of one of the additive models **must** be fixed to a non-zero value. It is also important to ensure that the energy range over which the model is calculated (which is determined by the response matrix in use) covers the energy range for which the photon flux is calculated. If the model to which the **cpflux** is applied integrates to zero then a divide-by-zero error will occur resulting in NaN values for the fit statistic.

Parameters are:

par1=Emin	Minimum energy over which photon flux is calculated.
par2=Emax	Maximum energy over which photon flux is calculated.
par3=Flux	Photon flux in ph/cm^2/s

6.4.4 gsmooth: gaussian smoothing

Gaussian smoothing with a variable width $\Sigma(E)$, which varies as the par2 power of the energy. The width at 6 keV is set with par1. Note that the energy binning must be uniform. If the response energies are not uniformly spaced then the **energies** command should be used to set uniform energy binning.

$$dC(E) = \frac{1}{\sqrt{2\pi\Sigma(E)^2}} \exp\left[-\frac{1}{2}\left(\frac{E-X}{\Sigma(E)}\right)^2\right] A(X)dX$$

$$\Sigma(E) = \sigma(E/6)^\alpha$$

where:

par1=σ	gaussian sigma at 6 keV
par2=α	power of energy for sigma variation

6.4.5 ireflect: reflection from ionized material

Convolution model for reflection from ionized material according to the method of Magdziarz & Zdziarski (1995, MNRAS, 273, 837). This is a generalization

of the **pexriv** and **bexriv** models. Ionization and opacities of the reflecting medium is computed as in the **absori** model. The reflection component alone can be obtained for $|rel_{refl}| < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to change then the limits of $|rel_{refl}|$ excluding zero (as then the direct component appears). If $E_c = 0$, there is no cutoff in the power law. The metal and iron abundance are variable with respect to those set by the command **abund**.

When using this model it is essential to extend the energy range over which the model is calculated both on the high and low end. The high end extension is required because photons at higher energies are Compton down-scattered into the target energy range. The low energy extension may be required to calculate ionization fractions correctly. The energy range can be extended using the **energies** `extend` command. The upper limit on the energies should be set above that for which the input spectrum has significant flux. To speed up the model, calculation of the output spectrum can be limited to energies below a given value by using **xset** to define IREFLECT_MAX_E (in units of keV). For instance, suppose that the original data extends up to 100 keV. To accurately determine the reflection it may be necessary to extend the energy range up to 500 keV. Now to avoid calculating the output spectrum between 100 and 500 keV use the command **xset** IREFLECT_MAX_E 100.0.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting IREFLECT_PRECISION eg **xset** IREFLECT_PRECISION 0.05. The default precision is 0.01 (ie 1%).

par1	reflection scaling factor (1 for isotropic source above disk)
par2=z	redshift
par3	abundance of elements heavier than He relative to the solar abundances
par4	iron abundance relative to the above
par5	cos i, the inclination angle
par6	disk temperature in K
par7	disk ionization parameter, $\xi = 4\pi\frac{F_{ion}}{n}$, where F_{ion} is the 5eV - 20keV irradiating flux, n is the density of the reflector; see Done et al., 1992, ApJ, 395, 275.

6.4.6 kdblur: convolve with the laor model shape

A convolution model to smooth a spectrum by relativistic effects from an accretion disk around a rotating black hole. Uses Ari Laor's calculation including GR effects (ApJ 376, 90). Modified from **laor** model by Andy Fabian and Roderick Johnstone.

par1	Index: power law dependence of emissivity (scales as R^{-par1})
par2	inner radius (units of GM/c^2)
par3	outer radius (units of GM/c^2)
par4	inclination (degrees)

6.4.7 kdblur2: convolve with the laor2 model shape

A convolution model to smooth a spectrum by relativistic effects from an accretion disk around a rotating black hole. The accretion disk has a broken-power law emissivity profile. Uses Ari Laor's calculation including GR effects (ApJ 376, 90). Modified from **laor2** model by Andy Fabian and Roderick Johnstone.

par1	Index: power law dependence of emissivity (scales as R^{-par1})
par2	inner radius (units of GM/c^2)
par3	outer radius (units of GM/c^2)
par4	inclination (degrees)
par5	radius at which emissivity power-law index changes
par6	Emissivity power-law index for radii >par6

6.4.8 kerrconv: accretion disk line shape with BH spin as free parameter

Convolves the current spectrum with the line shape from the **kerrdisk** model. A detailed description can be found in Brenneman & Reynolds (2006ApJ...652.1028B). This model is quite slow so is best used after models such as **laor** or **diskline** have been employed to get an estimate of the best-fit parameters.

par1	emissivity index for the inner disk
par2	emissivity index for the outer disk
par3	break radius separating the inner and outer portions of the disk (gravitational radii)
par4	dimensionless black hole spin
par5	disk inclination angle to the line of sight (degrees)
par6	inner radius of the disk in units of the radius of marginal stability
par7	outer radius of the disk in units of the radius of marginal stability

6.4.9 lsmooth: lorentzian smoothing

Lorentzian smoothing with a variable width, which varies as the par2 power of the energy. The width at 6 keV is set with par1.

$$\begin{aligned} dC(E) = & \quad \frac{\Sigma(E)}{2\pi[(E-X)^2+(\Sigma(E)/2)^2]} A(X)d(X) \\ \Sigma(E) = & \quad \sigma(E/6)^{\alpha} \end{aligned}$$

where:

par1=σ	lorentzian sigma at 6 keV
par2=α	power of energy for sigma variation

6.4.10 partcov: partial covering

A convolution model to convert some absorption model into a partial covering absorption. If the absorption model is M(E) then this is converted to (1-CvrFract) + Cvrfact * M(E). Note that when specifying the model it is important to put parentheses in the right place. Let this model be P(E) which we want to apply to an absorption model M(E) then use the result to multiply an additive model A(E). The combined model should be specified as (P*M)*A, not P*M*A or P*(M*A).

The parameters are:

par1=CvrFract	Covering fraction (0 <par1 <1).

6.4.11 rdblur: convolve with the diskline model shape

A convolution model to smooth a spectrum by relativistic effects from an accretion disk around a non-rotating black hole. Modified from **diskline** model (Fabian et al., MNRAS 238, 729) by Andy Fabian and Roderick Johnstone.

par1	Index: power law dependence of emissivity (scales as R^{par1}). If this parameter is 10 or greater then the accretion disk emissivity law $(1-\sqrt{6/R})/R^3$ is used.
par2	inner radius (units of GM/c^2)
par3	outer radius (units of GM/c^2)
par4	inclination (degrees)

6.4.12 reflect: reflection from neutral material

Convolution model for reflection from neutral material according to the method of Magdziarz & Zdziarski (1995, MNRAS, 273, 837). This is a generalization of the **pexrav** and **bexrav** models. The reflection component alone can be obtained for $|rel_{refl}| < 0$. Then the actual reflection normalization is $|rel_{refl}|$. Note that you need to change then the limits of $|rel_{refl}|$ excluding zero (as then the direct component appears). If $E_c = 0$, there is no cutoff in the power law. The metal and iron abundance are variable with respect to those set by the command **abund**. The opacities are those set by the command **xsect**. As expected in AGNs, H and He are assumed to be fully ionized.

When using this model it is essential to extend the energy range over which the model is calculated because photons at higher energies are Compton downscattered into the target energy range. The energy range can be extended using the **energies** extend command. The upper limit on the energies should be set above that for which the input spectrum has significant flux. To speed up the model, calculation of the output spectrum can be limited to energies below a given value by using **xset** to define REFLECT_MAX_E (in units of keV). For instance, suppose that the original data extends up to 100 keV. To accurately determine the reflection it may be necessary to extend the energy range up to 500 keV. Now to avoid calculating the output spectrum between 100 and 500 keV use the command **xset** REFLECT_MAX_E 100.0.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting REFLECT_PRECISION

eg **xset** REFLECT_PRECISION 0.05. The default precision is 0.01 (ie 1%).

par1	reflection scaling factor (1 for isotropic source above disk)
par2=z	redshift
par3	abundance of elements heavier than He relative to the solar abundances
par4	iron abundance relative to the above
par5	cos i, the inclination angle

6.4.13 rfxconv: angle-dependent reflection from an ionized disk

This convolution model from Chris Done combines the ionized disk table model from Ross and Fabian with the Magdziarz & Zdziarski Compton reflection code. It is described in Kolehmainen, Done & Diaz Trigo (2011) and is a modification of the model first described in Done & Gierlinski (2006).

The algorithm used is as follows.

1. Determine the average power-law index of the input spectrum between 2 and 10 keV. For this index and the other input parameters interpolate on the table models to generate the reflected spectrum from the ionized disk.
2. Estimate the average power-law index of the reflected spectrum over the range 12 - 14 keV.
3. Iterate over the Compton reflection models changing the cross-section at 10 keV until a match is found with the index calculated in the previous step.
4. Renormalize the reflection spectrum calculated in step 1 to match the Compton reflection calculated in step 3 at 14 keV.
5. Calculate the final reflection spectrum by using the renormalized ionized disk spectrum below 14 keV and the Compton reflection spectrum above 14 keV.

When using this model it is essential to extend the energy range over which the model is calculated because photons at higher energies are Compton downscattered into the target energy range. The energy range can be extended using the extend command. The upper limit on the energies should be set above that for which the input spectrum has significant flux. To speed up the

model, calculation of the output spectrum can be limited to energies below a given value by using xset to define RFXCONV_MAX_E (in units of keV). For instance, suppose that the original data extends up to 100 keV. To accurately determine the reflection it may be necessary to extend the energy range up to 500 keV. Now to avoid calculating the output spectrum between 100 and 500 keV use the command xset RFXCONV_MAX_E 100.0.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting RFXCONV_PRECISION eg xset RFXCONV_PRECISION 0.05. The default precision is 0.01 (ie 1%).

To use different ionized disk table model files than those installed change the directory searched for these files using xset RFXCONV_DIR.

The model parameters are as follows.

par1 = rel_{refl}	the relative reflection normalization. If rel_{refl} is negative tl the reflected component is returned.
par2 = $redshift$	
par3 = Fe_{abund}	the iron abundance relative to Solar. All other elements sumed to have Solar abundance.
par4 = $cosIncl$	cosine of the inclination angle (degrees).
par5 = $log(xi)$	the ionization parameter used by the table models.

6.4.14 simpl: comptonization of a seed spectrum

The SIMple Power Law model: An empirical model of Comptonization in which a fraction of the photons in an input seed spectrum is scattered into a power-law component (Steiner et al. 2009, PASP, 121, 1279). It is designed for use with soft thermal spectra that are not Compton thick and that have a photon index Gamma >1. **simpl** offers the advantage of operating in a self consistent manner, linking the seed spectrum to the generated power law. Compared to **powerlaw**, **simpl** gives equally good fits while also employing just two parameters, and **simpl** has the virtue of eliminating the divergence of **powerlaw** at low energies. Because **simpl** redistributes input photons to higher (and lower energies), for detectors with limited response matrices (at high or low energies), or with poor resolution, the sampled energies should be extended to adequately cover the relevant energy range (for details and an example, see the appendix in Steiner et al. 2009).

par1	Gamma, the photon power law index.
par2	The scattered fraction (between 0 and 1).
par3	A flag to switch between up-scattering only (> 0) and both up- and down-scattering (≤ 0)

6.4.15 xilconv: angle-dependent reflection from an ionized disk

This convolution model from Chris Done combines an ionized disk table model from the XILLVER model of Garcia et al. (2013) with the Magdziarz & Zdziarski Compton reflection code. It is a modification of the **rfxconv** model described in Kolehmainen, Done & Diaz Trigo (2011) which is a modification of the model first described in Done & Gierlinski (2006).

The algorithm used is as follows.

1. Determine the average power-law index of the input spectrum between 2 and 10 keV. For this index and the other input parameters interpolate on the table models to generate the reflected spectrum from the ionized disk.
2. Estimate the average power-law index of the reflected spectrum over the range 12 - 14 keV.
3. Iterate over the Compton reflection models changing the cross-section at 10 keV until a match is found with the index calculated in the previous step.
4. Renormalize the reflection spectrum calculated in step 1 to match the Compton reflection calculated in step 3 at 14 keV.
5. Calculate the final reflection spectrum by using the renormalized ionized disk spectrum below 14 keV and the Compton reflection spectrum above 14 keV.

When using this model it is essential to extend the energy range over which the model is calculated because photons at higher energies are Compton down-scattered into the target energy range. The energy range can be extended using the extend command. The upper limit on the energies should be set above that for which the input spectrum has significant flux. To speed up the model, calculation of the output spectrum can be limited to energies below a given value by using xset to define XILCONV_MAX_E (in units of keV). For instance, suppose that the original data extends up to 100 keV. To accurately determine the reflection it may be necessary to extend the energy range up to 500 keV.

Now to avoid calculating the output spectrum between 100 and 500 keV use the command xset XILCONV_MAX_E 100.0.

The core of this model is a Greens' function integration with one numerical integral performed for each model energy. The numerical integration is done using an adaptive method which continues until a given estimated fractional precision is reached. The precision can be changed by setting XILCONV_PRECISION eg xset XILCONV_PRECISION 0.05. The default precision is 0.01 (ie 1%).

To use different ionized disk table model files than those installed change the directory searched for these files using xset XILCONV_DIR.

The model parameters are as follows.

par1 = rel_{refl}	the relative reflection normalization. If rel_{refl} is negative th the reflected component is returned.
par2 = $redshift$	
par3 = Fe_{abund}	the iron abundance relative to Solar. All other elements sumed to have Solar abundance.
par4 = $cosIncl$	cosine of the inclination angle (degrees).
par5 = $log(xi)$	the ionization parameter used by the table models.
par6 = $cutoff$	the exponential cut-off energy (keV).

6.4.16 vashift: velocity shift an additive model

This convolution model velocity shifts an additive model. It takes the calculated model and shifts energies $-Ev/c$.

The **energies** command must be to used to extend the energy range over which the model.

The parameter is:

par1=v velocity (km/s)

An example model use is:

```
XSPEC>model phabs*vashift*(pow+ga)
```

which velocity shifts the power-law and gaussian then multiplies by local absorption.

6.4.17 vmshift: velocity shift a multiplicative model

This convolution model velocity shifts a multiplicative model. It takes the calculated model and shifts energies by $-Ev/c$.

The **energies** command must be to used to extend the energy range over which the model is being calculated.

The parameter is:

par1=v	velocity (km/s)

An example model use is:

```
XSPEC>model phabs*(vmshift*phabs)*pow
```

which multiplies the power-law by both a velocity shifted and local absorption.

6.4.18 zashift: redshift an additive model

This convolution model redshifts an additive model. It takes the calculated model and shifts energies by 1/(1+z) then applies an additional 1/(1+z) factor to the model values.

The **energies** command must be to used to extend the maximum energy over which the model is being calculated to (1+z) times the maximum energy in the response.

The parameter is:

par1	redshift

An example model use is:

```
XSPEC>model phabs*zashift*(pow+ga)
```

which redshifts the power-law and gaussian then multiplies by local absorption.

6.4.19 zmshift: redshift a multiplicative model

This convolution model redshifts a multiplicative model. It takes the calculated model and shifts energies by 1/(1+z).

The **energies** command must be to used to extend the maximum energy over which the model is being calculated to (1+z) times the maximum energy in the response.

The parameter is:

par1	redshift

An example model use is:

```
XSPEC>model phabs*(zmshift*phabs)*pow
```

which multiplies the power-law by both a redshifted and local absorption.

6.5 Pile-Up Model Components

6.5.1 pileup: CCD pile-up model for Chandra

CCD pile-up model used for brightish point sources observed by Chandra. This is an implementation of the fast pile-up algorithm proposed by John Davis (see http://space.mit.edu/~davis/papers/pileup2001.pdf). The frame time and maximum number of photons to pile up should be fixed. The grade morphing is expressed through a single parameter, alpha, which should be left as a free parameter. This model should be considered in beta test. Note that to calculate fluxes etc. for the model you must remove the **pileup** component. The pile-up model is similar to the operation of the convolution models, differing only in the treatment of the detector efficiency during the convolution. Note that **renorm** will not work with **pileup** since increasing the normalization does not linearly increase the predicted count rate. Therefore you should set **renorm none** prior to doing a fit with **pileup**.

par1	frame time (in seconds)
par2	maximum number of photons to pile up
par3	grade correction for single photon detection
par4	grade morphing parameter (good grade fraction is assumed proportional to par4$^{(p-1)}$ where p is the number of piled photons)
par5	PSF fraction. Only this fraction will be treated for pile-up. Note that this is **not** the fraction of the PSF included in the extraction region but is the fraction of counts in the region which are from the point source whose pile-up is being modeled. For this model to work well the extraction region should be large enough to contain essentially all the PSF.
par6	Number of regions. The counts to be piled-up will be distributed among par6 regions, which will be piled-up independently.
par7	Value of FRACEXPO keyword in ARF.

6.6 Mixing Model Components

Mixing models perform transformations on the available spectra. The spectra must be assigned to more than one data group in order have any effect. Each data group is a "region" of the observation, and the "mixing transformation" allows the model flux from one such "region" to influence another region. Thus, these models, unlike all the others, can be two-dimensional in effect. It follows that they differ from all the other models in that the data must be read before the model can be defined. In most cases also, the input data must contain additional information in order to use the model. This additional information can be in the form of the OGIP standard XFLT keys, which allow a set of scalar real values, or in some cases additional files to be read containing spatial information.

XSPEC will return error messages if the data are not loaded, not compatible with the model (i.e. do not contain the required additional information), loaded inconsistently with the use of the model (the division of data into regions is incorrect, or data within a given region fail consistency checks). It will also return error messages if the data are subsequently changed to a set that violates these consistency checks, and additionally remove the model definition.

6.6.1 ascac: ASCA surface brightness model

Mixing model for ASCA data. Written for cluster data so uses beta or two power-law surface brightness models. Includes a calculation of the telescope effective area so no arf should be applied to input files. Note that this model is very slow if any of the parameters are free.

The model is used by reading spectra in as separate datagroups. Each input file requires an XFLTnnnn keyword set to "region: N" where N is a different number for each file (eg if concentric annuli are in use then number outwards). The normalizations for each datagroup should be linked since the **ascac** model takes care of the relative normalizations based on the surface brightness model used. A maximum of five different spatial regions is allowed. The absolute normalization is not reliable so this model should not be used to derive fluxes.

par1	Alpha
par2	Beta
par3	Core (arcmin)
par4	0 ⇒ beta model, 1 ⇒ 2-power-law

6.6.2 projct: project 3-D ellipsoidal shells onto 2-D elliptical annuli

This model performs a 3-D to 2-D projection of prolate ellipsoidal shells onto elliptical annuli. The annuli can have varying ellipticities and position angles but must have the same center. The user should extract spectra in a series of annuli. Each spectrum needs three additional keywords (as XFLTnnnn) in the spectrum extension. These should be "major: x", "minor: y", "orient: z" where x, y, z are the semi-major axis, semi-minor axis, and position angle (in degrees), respectively, for the outer boundary of the annulus. It is assumed that the inner boundary is specified by the outer boundary of the previous annulus. The lengths can be in any consistent units although for numerical accuracy they should have reasonable values. Optional pairs of extra XFLTnnnn keywords can be used to specify start and end angles for a partial annulus. These have keys angleNl and angleNu so typical keyword values might be "angle1l: 30.0" and "angle1u: 45.0". These angles should be given relative to the same zero as the position angle.

The user reads in the spectra as separate datagroups and sets model parameters for each datagroup. The model for datagroup J will be the model in the shell

whose outer boundary is a prolate ellipsoid of semi-major and semi-minor axes given by the semi-major and semi-minor axes in the XFLT keywords for dataset J. The **projct** model sums up the appropriate fractions of each ellipsoid model to make the projected spectrum.

For example, suppose we extract spectrum from three elliptical regions defined by (1,0.5,0), (2,1,0), (3,1.5,0). That is the first region is in an ellipse of semi-major axis 1 and semi-minor axis 0.5. The second region is an elliptical annulus whose inner boundary has semi-major axis 1 and semi-minor axes 0.5 and whose outer boundary has semi-major axis 2 and semi-minor axis 1. The third region is defined similarly. The model fit has a temperature of 2 keV for the first datagroup, 3 keV for the second, and 4 keV for the third. The actual model fit to the first dataset has contributions from all three temperatures, the second only from the 3 and 4 keV components, and the third only from the 4 keV component. The weighting is the fraction of the ellipsoidal volume intersected by the elliptical annular cross-section. Thus the normalizations correspond to the emission measure in each ellipsoidal shell.

If multiple observations are to be analyzed, data sets from different observations corresponding to the same annulus should be part of the same data group. For example, given the following 4 data files:

Data sets for obs 1: obs1_an1, obs1_an2
Data sets for obs 2: obs2_an1, obs2_an2

The proper data loading command is:

```
XSPEC12>data 1:1 obs1_an1 1:2 obs2_an1 2:3 obs1_an2 2:4 obs2_an2
```

The **projct** model has 3 (fixed) parameters, which can be used to define the inner ellipse of the region being analyzed. For instance, in the example above we could have only read in spectra for the outer two regions but then set the **projct** parameters to (1.0,0.5,0.0). This would have allowed us to determine the temperatures and emission measures of the outer two annuli without having to worry about fitting a model to the central region.

par1	semi-major axis of inner boundary ellipse
par2	semi-minor axis of inner boundary ellipse
par3	position angle of inner boundary ellipse

6.6.3 suzpsf: suzaku surface brightness model

Mixing model for Suzaku data. Mixes the spectra between datagroups based on the PSF overlap between selected regions. A surface brightness model is required to calculate the mixing and this can be supplied in several ways. If SUZPSF-IMAGE has been set to some image file (using **xset**) then this image will be used for the surface brightness distribution. If SUZPSF-IMAGE has not been set then either a beta or two power-law model is used. In this case the model parameters determine the shape of the surface brightness distribution. If SUZPSF-RA and SUZPSF-DEC are set they are used as the center of the distribution. They should be specified either in decimal degrees or as hh:mm:ss.s and dd:mm:ss.s. If SUZPSF-RA and SUZPSF-DEC are not set then the centroid of the wmap will be used as the center of the surface brightness distribution.

The PSF used is an empirical model of a sum of two exponentials and a Gaussian with coefficients determined from an observation of MCG-6-30-15 performed early in the mission.

The model works by calculating the mixing factors. It will recalculate these factors if any of the SUZPSF-* or any of the model parameters are changed. Calculating the mixing factors is very slow so should be avoided as much as possible. To speed things up, it is possible to save the mixing factor array to a FITS file and re-use it during a later calculation. To save a mixing factor calculation, prior to loading the mixing model (using the **model** command), use **xset** to set the variable SUZPSF-MIXFACT-OFILEn to the name of the output FITS file, and where n is an integer corresponding to the observation number:

```
XSPEC12> xset SUZPSF-MIXFACT-OFILE1 fact_obs1.fits
```

Conversely, a saved factor array can be read in by setting SUZPSF-MIXFACT-IFILEn:

```
XSPEC12> xset SUZPSF-MIXFACT-IFILE1 fact_obs1.fits
```

Multiple observations can be fit simultaneously. In this case the observations should be read in each datagroup in the same order, e.g.

```
XSPEC12> data 1:1 obs1reg1 1:2 obs2reg1 1:3 obs3reg1 2:4 obs1reg2 2:5 obs2reg2
```

par1	Alpha (not used if Switch=0)
par2	Beta
par3	Core (arcmin)
par4	Switch (0 = beta model, 1 = 2-power-law)

6.6.4 xmmpsf: xmm surface brightness model

Mixing model for XMM data. Mixes the spectra between datagroups based on the PSF overlap between selected regions. A surface brightness model is required to calculate the mixing and this can be supplied in several ways. If XMMPSF-IMAGE has been set to some image file (using **xset**) then this image will be used for the surface brightness distribution. If XMMPSF-IMAGE has not been set then either a beta or two power-law model is used. In this case the model parameters determine the shape of the surface brightness distribution. If XMMPSF-RA and XMMPSF-DEC are set they are used as the center of the distribution. They should be specified either in decimal degrees or as hh:mm:ss.s and dd:mm:ss.s. If XMMPSF-RA and XMMPSF-DEC are not set then the centroid of the wmap will be used as the center of the surface brightness distribution.

The model works by calculating the mixing factors. It will recalculate these factors if any of the XMMPSF-* or any of the model parameters are changed. Calculating the mixing factors is very slow so should be avoided as much as possible. To speed things up, it is possible to save the mixing factor array to a FITS file and re-use it during a later calculation. To save a mixing factor calculation, prior to loading the mixing model (using the **model** command), use **xset** to set the variable XMMPSF-MIXFACT-OFILEn to the name of the output FITS file, and where n is an integer corresponding to the observation number:

```
XSPEC12> xset XMMPSF-MIXFACT-OFILE1 fact_obs1.fits
```

Conversely, a saved factor array can be read in by setting XMMPSF-MIXFACT-IFILEn:

```
XSPEC12> xset XMMPSF-MIXFACT-IFILE1 fact_obs1.fits
```

Multiple observations can be fit simultaneously. In this case the observations should be read in each datagroup in the same order, e.g.

```
XSPEC12> data 1:1 obs1reg1 1:2 obs2reg1 1:3 obs3reg1 2:4 obs1reg2 2:5 obs2reg2
```

par1	Alpha (not used if Switch=0)
par2	Beta
par3	Core (arcmin)
par4	Switch (0 = beta model, 1 = 2-power-law)

Appendix A

The User Interface

A.1 Introduction

All communication with the user in XSPEC is performed through the tcl user interface. When XSPEC starts, a tcl interpreter is initialized, and the XSPEC commands are added to it so that the tcl interpreter understands them. The XSPEC commands, which are C++ functions, define the syntax through a new built-in library of utility functions. The parser used in earlier versions of XSPEC has been discontinued: however the syntax understood by XSPEC12 is much the same as before.

A.2 XSPEC and tcl/tk

Because tcl is a full scripting language, users can write complex scripts with loops, branching, etc., which utilize XSPEC commands. Here we describe how to use those features of tcl necessary to give the user similar functionality to that available in previous versions of XSPEC, and to give information on the details of our tcl implementation that may be useful to experienced tcl users. For a description of tcl, see, for example, Practical Programming in Tcl and Tk, B. Welch, (1997, Prentice Hall).

Tk, tcl's companion graphical user interface (GUI) toolkit, is also loaded by XSPEC on startup. It is planned that future versions of XSPEC will provide an optional GUI side-by-side with the command line interface (CLI). Although XSPEC does not currently use tk, its presence allows users to write XSPEC

scripts with graphical interfaces using Tk commands.

A.3 A note on command processing

To emulate the performance of the former XSPEC parser, the command functions are programmed to react similarly to some of its features.

The # sign is used for comments in tcl, but may appear only at the beginning of a command. tcl and XSPEC both ignore carriage returns on a new line, but XSPEC also ignores the skip character /. The character sequence /* entered during a command exits that command, sets any responses to the default response, and returns the user to the prompt. The \ character is used in tcl for continuing a command onto the next line.

Additionally, note that in tcl, commands and their arguments are delimited by white space. They are terminated by a newline or semicolon, unless there is an open set of parentheses " " constituting a loop or test structure. In other words, in tcl the following starts a loop:

```
while { condition } {
...
}
but

while { condition }
{

}
```

is incorrect, since in the latter case the while command terminates at the end of the first line.

A.4 Command Recall/Editing

The XSPEC/tcl interface also uses gnu readline for command input, which allows command line editing and interactive command recall. On most systems, the left and right arrow keys and the backspace/delete key can be used to navigate and edit the command line. The up and down arrow keys can be used to step thru the command history list. Gnu readline is highly customizable,

and many more editing/recall functions are available. Readline documentation can be generated in either postscript or html format from the files in the xanadu/readline/doc directory distributed with the source.

The default implementation of tcl also supports a C-shell like command recall mechanism. The history command gives a numbered list of the most recently entered commands. Any command in the list can be re-executed by entering !n, where n is the number of the command in the history list. The previous command can be re-executed by entering !!. The most recent command that begins with a string can be re-executed by entering !prefix, where prefix is the string the command begins with.

Note that command recall is implemented using the tcl unknown procedure, part of which is a script file loaded by tcl at run time. See the section on the unknown command for more details on how it is implemented in XSPEC.

A.5 Logging

The log command can be used to open a log file to which all input and and output to tcl will be written. Reading these log files can potentially be confusing when logging tcl flow control commands such as while or for. This is because tcl treats the body of these commands as an argument of the command. Thus when the command is echoed to the log file, the entire body of the command is echoed with it.

In order to make this situation less confusing, before commands are echoed to the command file, all newline characters are replaced by semicolons, and the resulting command line is truncated to 80 characters. Then any commands executed with in the body of a flow control command are echoed as they are executed.

Consider the following sequence of tcl commands within XSPEC:

```
XSPEC12> log
Logging to file: xspec.log
XSPEC12> set i 1 ; set product 1
1
XSPEC12> while {$i <= 5} {
XSPEC12> set product [expr $product * $i]
XSPEC12> incr i
XSPEC12> }
XSPEC12> set product
120
```

```
 XSPEC12>
```

This would produce the following output in the file xspec.log:

```
Logging to file: xspec.log
 XSPEC12> set i 1
 set product 1
 1
 XSPEC12> while {$i <= 5} {;set product [expr $product * $i];incr i;}
   expr $product * $i
  set product [expr $product * $i]
  incr i
   expr $product * $i
  set product [expr $product * $i]
  incr i
   expr $product * $i
  set product [expr $product * $i]
  incr i
   expr $product * $i
  set product [expr $product * $i]
  incr i
   expr $product * $i
  set product [expr $product * $i]
  incr i
 XSPEC12> set product
 120
 XSPEC12>
```

A.6 Command Completion

tcl attempts to match the name of any entered command as an abbreviation of a valid command (either a tcl or XSPEC command). If the entered command matches more than one valid command, tcl then lists the possible choices, but does not execute the command. For XSPEC commands, aliases have been constructed matching the command to its minimum abbreviation, as listed when typing **?** at the XSPEC prompt (see under Aliases).

For example, the minimum abbreviation for the **plot** command is **pl**. Thus, typing **pl** will execute the plot command, even though this would otherwise be ambiguous with the tk command **place**. Command completion is also implemented using the tcl **unknown** procedure, part of which is a script file loaded by tcl at run time, and may be different or not exist on your system. See the

section in this help file on the **unknown** command for more details on how it is implemented in XSPEC.

N.B. tcl explicitly switches off command completion for scripts. Because of the way scripts are implemented in XSPEC, however, command abbreviations nevertheless do work in scripts entered with the **@** command, but not when entered from the command line or using the source command. See below for more details about tcl scripting.

A.7 Unknown Procedure

tcl provides a facility whereby if it cannot match an entered command to its list of known commands, it calls the **unknown** procedure, with the unmatched command (along with its arguments) as its argument. The version of `init.tcl` distributed with tcl contains a version of the **unknown** procedure. When tcl initializes, it looks in several standard places for a script file named `init.tcl`, which it executes if found. The **unknown** procedure is where tcl does command completion and automatic shell command execution.

At start up time, XSPEC loads its own **unknown** procedure, which it uses to intercept script processing requests of the form

```
XSPEC12>@<script>
```

and renames the previously defined **unknown** procedure to **tclunknown**. If XSPEC is not doing any special processing, it simply passes any unmatched commands on to **tclunknown**, which then processes them as usual. XSPEC has its own special version of the **unknown** procedure

These factors need to be taken into consideration for programmers writing tcl scripts for use within XSPEC. For example, if after initialization, users wishing to load a different version of the standard tcl **unknown** procedure should name that procedure **tclunknown**, rather than **unknown**.

A.8 Aliases

Command name aliases can be constructed using the tcl **interp** command:

```
interp alias {} <command_alias> {} <xspec_command>
```

where <xspec_command> is the name of the command you wish to make an alias for, and <command_alias> is the name of the alias you wish to set for the command. The **{}** are required syntax.

To delete the alias <command_alias>, simply nullify it with:

```
interp alias {} <command_alias> {}
```

A.9 Initialization Script

When running interactively, the user has the option of providing an initialization script, which will be executed after XSPEC completes its startup procedure, ie. just before it begins prompting for commands. The file should be named xspec.rc and located in the directory `$HOME/.xspec`. If one runs XSPEC in batch mode, by specifying a script on the command line, this initialization script is not executed. When multiple users are accessing a single system-wide XSPEC build, the installer can also provide additional initialization that will apply to all users. See the section "XSPEC Overview and Helpful Hints: Customizing XSPEC" for more details.

A.10 XSPEC Command Result

After being executed, many tcl commands return a result string, which is echoed to the terminal when the command is entered on the command line. When writing complex tcl scripts, this result can be stored and/or used as a test in loops, etc. When XSPEC commands are executed, they write information to the terminal by writing directly to the appropriate output channel. However, when running interactively, the tcl result string is also written to the terminal after the command is executed. The **tclout** command (see command description) creates tcl variables from XSPEC calculations.

A.11 Script Files

XSPEC/tcl script files can be executed in three different ways, as follows:

```
xspec - <script>
```

executing script on initialization

```
XSPEC12>@ <script>
```

executing script from within the program

```
XSPEC12>source tclscript
```

use tcl's source command from within the program.

Each of these usages does something slightly different. In the first form, XSPEC will execute a file called <`script`>. One may execute a series of script files at startup with the following command syntax:

```
unix> xspec - file1 file2 file3 ...
```

Note that the space following the "-" is required.

The second form is **@**<`name`>, where <`name`> is the name of the script file to be executed. Here the default extension of `.xcm` is assumed. Scripts containing valid tcl or XSPEC commands will be executed using this form, and (unless the script ends in **quit** or **exit**) will return to the interactive prompt after completion.

The final form, using tcl's **source** command, is intended for the special case where the script contains the implementation of a new command written in tcl/tk. See the section on writing custom commands for more details. In current tcl versions it compiles the script into bytecode representation for more efficient execution, and adds any procedures defined in the script to the set of commands understood by the interpreter. It will not work for general scripts containing XSPEC/tcl commands, for example those produced by XSPEC's **save** command. These should rather be executed using the **@** form.

Note that only in the second case (**@**) is there a default filename suffix (i.e. `.xcm`), for both the other methods of script execution the filename must be given in full.

tcl internally switches off the mechanism that expands command abbreviations when scripts are executed. If this were not done, the user could specify command abbreviations that change the behavior of the tcl command **set** (e.g. **set** for the **setplot** command would redefine tcl's command for setting variables). This behavior can be overridden with the statement

```
set tcl_interactive 1
```

near the beginning of the script, but it is not recommended to do so. Instead, we strongly recommend spelling out command names in full within XSPEC scripts.

By default, when XSPEC is executing a script file, it echoes each command to the terminal before it is executed. This can be controlled using the tcl variable **xs_echo_script**, whose default value is 1. If this variable is set to 0, the commands from the script file will not be echoed to the terminal.

In summary, we suggest the following convention:

- Running an xspec script from the unix command prompt is intended to be used for background processing or overnight batch jobs. Using the unix **at** command, one can arrange to receive the log file by e-mail.
- The **@** usage is intended for processing previously run xspec command sequences, such as are produced by the **save** command.
- The **source** usage, as well as executing the commands in the script, performs the equivalent of pre-compiling the script for later invocation. Its most appropriate use is in preparing new custom XSPEC command procedures. Once the script is working correctly, it can be placed in the user script directory and become part of the user's standard command set. For examples, see the scripts `addline.tcl` and `modid.tcl` in the directory `$SPECTRAL/scripts` that implement the commands **addline** and **modid**. These also show how to make commands self-documenting.

A.12 Unix Shell Commands

Shell commands can be executed within XSPEC using the tcl **exec** command. When running interactively, if Tcl cannot find a command that matches that entered on the command line, it will search for a shell command that matches the entered command. If it finds a match, it automatically executes the shell command via **exec**. Note that this feature is implemented using the tcl **unknown** procedure, part of which is a script file loaded by tcl at run time, and may be different or not exist on your system. See Section A.7 for more details on how **unknown** is implemented in XSPEC.

Note that the tcl **exec** command executes the given command directly, without first passing it on to the shell. Thus no globbing (ie. expansion of wildcards such as `*.pha`) is performed. If you wish to pass your command through a shell for wildcard expansion, etc, use the **syscall** command.

If you want to start a subshell from within XSPEC, simple type the command for starting that shell, ie. type

```
XSPEC12>csh
```

in order to start a C-shell. Note that typing

```
XSPEC12>exec csh
```

will not work properly. Giving the

```
XSPEC12>syscall
```

command with no arguments will start a subshell using your current shell (csh, tcsh, bash, sh, etc).

A.13 Writing Custom XSPEC commands

XSPEC commands can be written by users as tcl procedures, which have similarities with fortran or C/C++ subroutines. Within XSPEC, tcl procedures can take arguments and execute XSPEC and tcl commands. The syntax for specifying arguments to a tcl procedure is as follows:

```
proc my_proc {arg1 arg2}{
...

data 1:1 ${arg1}_s0_20
data 2:2 ${arg2}_s1_20
...
}
```

Here, <`arg1`>, <`arg2`> are values supplied by the user (here, part of a filename) from the command line, and substituted wherever <`${arg1}`>, <`${arg2}`> appear within the script. One may also give an argument a default value, so that the command so created may be invoked even without needing to specify the argument:

```
proc my_proc {arg1    {arg2 file2} }  {
...
}
```

Note that the parentheses enclosing both <`arg2`> and <`file2`> in this expression distinguish this from the case where 3 arguments are required for **my_proc**. Once this file is created, it needs to be **source**'d once, which compiles the script into an internal bytecode representation (this is similar to the way Java operates). Alternatively, one may place it in the user script directory and create an index in that directory, after which case it will be found automatically and compiled the first time it is invoked.

The user script directory is given by the line

```
USER_SCRIPT_DIRECTORY:
```

in the `Xspec.init` file that is copied into `$HOME/.xspec` when the user starts XSPEC for the first time (the supplied default value for this directory is the `$HOME/.xspec` directory itself). After the script is placed there, perform the following command

```
%xspec2
XSPEC12>cd <USER_SCRIPT_DIRECTORY>
XSPEC12>auto_mkindex .
XSPEC12>exit
```

This will instruct XSPEC to build an index of scripts to be loaded on xspec startup. On the next invocation of XSPEC, the script will be sourced on startup and will appear in the list of commands XSPEC understands.

The **my_proc** procedure is then defined such that one may type:

```
XSPEC12>my_proc eso103 eso104
```

and the data statement in the above example will be executed as if the following had been entered:

```
data 1:1 eso103_s0_20
data 2:2 eso104_s1_20
```

The tcl **info** command can be used to show which procedures have been defined:

```
XSPEC12>info commands <procedure name>
```

This will return <`procedure name`> if that procedure has been compiled (source'd) already or is a built-in command, or nothing if it has not (yet) been invoked or defined.

A.14 Scripting commands that prompt the user

The commands **model**, **editmod**, **addcomp**, **newpar**, and **fakeit** may prompt the user for more information when used interactively. In order to write scripts that use these commands, one must know how to force XSPEC to enter the information that would be prompted for. The technique is exemplified as follows. Suppose we defined a procedure **xmodel** that makes a model with certain predefined parameter values:

```
set p1  {1.5 0.001 0 0 1.E05 1.E06}
set p2  {1 0.001 0 0 1.E05 1.E06}
proc xmodel {modelString param1 param2 args} {
  model $modelString & $param1 & param2 & /*
}
```

Here the **&** character is taken by XSPEC as a carriage return, delimiting the model string and parameter arguments into separate input lines.

The procedure **xmodel** may be compiled with the command

```
XSPEC12> source xmodel.tcl
```

This creates **xmodel** as a command with two arguments which sets subsequent parameters to their default values. It can be invoked e.g. by

```
XSPEC12>xmodel {wa(po + peg)} $p1 $p2
```

Note that the model string, which contains spaces, needs to be entered within **{}** or double quotes. Note also that tcl understands a single string argument, args, as in

```
proc  tclscript { args } {
...
}
```

to mean a variable number of arguments to a procedure (it is supplied as a tcl list, which can be split within the procedure into separate strings for digestion by xspec if present).

A.15 Script Example

In the directory `$HEADAS/../spectral/session` is a script file called `tclex.xcm`. This script gives an example of how one might use the power of tcl's scripting language in an XSPEC session. This script should be executed with

```
XSPEC12> @tclex
# This script gives an example of how one might use the power of tcl's
# scripting language in an XSPEC session.  In this example, XSPEC loops
# thru 3 data files (file1, file2 and file3) and fits them each to the
# same model 'wabs(po+ga)'.  After the fit the value of parameter 4 (the
# line energy for the gaussian) for each data set is saved to a file.

# Keep going until fit converges.
query yes

# Open the file to put the results in.
set fileid [open fit_result.dat w]

for {set i 1} {$i < 4} {incr i} {

# Set up the model.
  model wabs(po+ga) & /*

# Get the file.
  data file$i

# Fit it to the model.
  fit

# Get the values specified for parameter 4.
  tclout param 4
  set par4 [string trim $xspec_tclout]

# Turn it into a Tcl list.
  regsub -all { +} $par4 { } cpar4
  set lpar4 [split $cpar4]

# Print out the result to the file.  Parameter value is
```

```
# the 0th element of the list ‘lpar4’.
  puts $fileid "$i [lindex $lpar4 0]"

}

# Close the file.
close $fileid
end
```

The user is encouraged to read the voluminous on-line documentation and literature available about tcl in order to benefit fully its flexible command processing, graphical interfacing, and scripting capabilities. See http://www.tcl.tk for much more information and extensive bibliography.

Appendix B

Statistics in XSPEC

B.1 Introduction

There are two operations performed in XSPEC that require statistics. The first is parameter estimation, which comprises finding the parameters for a given model that provide the best fit to the data and then estimating uncertainties on these parameters. The second operation is testing whether the model and its best-fit parameters actually match the data. This is usually referred to as determining the goodness-of-fit.

Which statistics should be used for these two operations depends on the probability distributions underlying the data. Almost all astronomical data are drawn from one of two distributions: Gaussian (or normal) and Poisson. The Poisson distribution is the familiar case of counting statistics and is valid whenever the only source of experimental noise is due to the number of events arriving at the detector. This is a good approximation for modern CCD instruments. If some other sort of noise is dominant then it is usually described by the Gaussian distribution. A common example of this is detectors that require background to be modeled in some way, rather than directly measured. The uncertainty in the background modeling is assumed to be Gaussian.

In the limit of large numbers of counts the Poisson distribution can be well approximated by a Gaussian so the latter is often used for detectors with high counting rates. In most cases this will cause no errors and does simplify the handling of background uncertainties however care should be exercised that no systematic offsets are introduced.

A fuller discussion of many of the issues discussed in this appendix can be found in Siemiginowska (2011).

B.2 Parameter Estimation

The standard statistic used in parameter estimation is the maximum likelihood. This is based on the intuitive idea that the best values of the parameters are those that maximize the probability of the observed data given the model. The likelihood is defined as the total probability of observing the data given the model and current parameters. In practice, the statistic used is twice the negative log likelihood.

Gaussian data (chi)

The likelihood for Gaussian data is

$$L = \prod_{i=1}^{N} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left[\frac{-(y_i - m_i)^2}{2\sigma_i^2}\right] \tag{B.1}$$

where y_i are the observed data rates, $_i$ their errors, and m_i the values of the predicted data rates based on the model (with current parameters) and instrumental response. Taking twice the negative natural log of L and ignoring terms which depend only on the data (and will thus not change as parameters are varied) gives the familiar statistic :

$$S^2 = \sum_{i=1}^{N} \frac{(y_i - m_i)^2}{\sigma_i^2} \tag{B.2}$$

commonly referred to as χ^2 and used for the **statistic chi** option.

Gaussian data with background (chi)

The previous section assumed that the only contribution to the observed data was from the model. In practice, there is usually background. This can either be included in the model or taken from another spectrum file (read in using the back command). In the latter case the y_i become observed data rates from the source spectrum subtracted by the background spectrum and the σ_i are the source and background errors added in quadrature. Since the difference of two Gaussians variables is another Gaussian variable, the S^2 statistic can still be used in this case.

Poisson data (cstat)

The likelihood for Poisson distributed data is:

$$L = \prod_{i=1}^{N} (tm_i)^{S_i} \mathrm{e}^{-tm_i} / S_i! \tag{B.3}$$

where S_i are the observed counts, t the exposure time, and m_i the predicted count rates based on the current model and instrumental response. The maximum likelihood-based statistic for Poisson data, given in Cash (1979), is :

$$C = 2 \sum_{i=1}^{N} (tm_i) - S_i \ln (tm_i) + \ln (S_i!) \tag{B.4}$$

The final term depends only on the data (and hence makes no difference to the best-fit parameters) so can be replaced by Stirling's approximation to give :

$$C = 2 \sum_{i=1}^{N} (tm_i) - S_i + S_i (\ln (S_i) - \ln (tm_i)) \tag{B.5}$$

which provides a statistic which asymptotes to S^2 in the limit of large number of counts (Castor, priv. comm.). This is what is used for the **statistic cstat** option. Note that using the S^2 statistic instead of C is not recommended since it can produce biassed results even when the number of counts is quite large (see e.g. Humphrey et al. 2009).

Poisson data with Poisson background (cstat)

This case is more difficult than that of Gaussian data because the difference between two Poisson variables is not another Poisson variable so the background data cannot be subtracted from the source and used within the C statistic. The combined likelihood for the source and background observations can be written as:

$$L = \prod_{i=1}^{N} \frac{(t_s(m_i + b_i))^{S_i} \mathrm{e}^{-t_s(m_i+b_i)}}{S_i!} \times \frac{(t_b b_i)^{B_i} \mathrm{e}^{-t_b b_i}}{B_i!} \tag{B.6}$$

where t_s and t_b are the exposure times for the source and background spectra, respectively, B_i are the background data and b_i the predicted rates from a model for the expected background. If there is a physically motivated model for the

background then this likelihood can be used to derive a statistic which can be minimized while varying the parameters for both the source and background models.

As a simple illustration suppose the source spectrum is source.pha and the background spectrum back.pha. The source model is an absorbed apec and the background model is a power-law. Further suppose that the background model requires a different response matrix to the source, backmod.rsp say. The fit is set up by:

```
XSPEC12> data 1:1 source.pha 2:2 background.pha
XSPEC12> resp 2:1 backmod.rsp 2:2 backmod.rsp
XSPEC12> model phabs(apec)
XSPEC12> model 2:backmodel pow
```

where the normalization of the apec model is fixed to zero for the second data group (i.e. the background spectrum) and the parameters of the background model are linked between the data groups.

If there is no appropriate model for the background it is still possible to proceed. Suppose that each bin in the background spectrum is given its own parameter so that the background model is $b_i = f_i$. A standard XSPEC fit for all these parameters would be impractical however there is an analytical solution for the best-fit f_i in terms of the other variables which can be derived by using the fact that the derivative of L will be zero at the best fit. Solving for the f_i and substituting gives the profile likelihood:

$$W = 2\sum_{i=1}^{N} t_s m_i + (t_s + t_b) f_i - S_i \ln(t_s m_i + t_s f_i) - B_i \ln(t_b f_i) - S_i(1 - \ln S_i) - B_i(1 - \ln B_i) \tag{B.7}$$

where

$$f_i = \frac{S_i + B_i - (t_s + t_b) m_i + d_i}{2(t_s + t_b)} \tag{B.8}$$

and

$$d_i = \sqrt{[(t_s + t_b) m_i - S_i - B_i]^2 + 4(t_s + t_b) B_i m_i} \tag{B.9}$$

If any bin has S_i and/or B_i zero then its contribution to W (W_i) is calculated as a special case. So, if S_i is zero then:

$$W_i = t_s m_i - B_i \ln\left(t_b/(t_s + t_b)\right) \tag{B.10}$$

If B_i is zero then there are two special cases. If $m_i < S_i/(t_s + t_b)$ then:

$$W_i = -t_b m_i - S_i \ln\left(t_s/(t_s + t_b)\right) \tag{B.11}$$

otherwise:

$$W_i = t_s m_i + S_i(\ln S_i - \ln\left(t_s m_i\right) - 1) \tag{B.12}$$

This W statistic is used for **statistic cstat** if a background spectrum with Poisson statistics has been read in. In practice, it works well for many cases but for weak sources can generate an obviously wrong best fit. It is not clear why this happens although binning to ensure that every bin contains at least one count often seems to fix the problem. In the limit of large numbers of counts per spectrum bin a second-order Taylor expansion shows that W tends to :

$$\sum_{i=1}^{N} \left(\frac{[S_i - t_s m_i - t_s f_i]^2}{t_s(m_i + f_i)} + \frac{[B_i - t_b f_i]^2}{t_b f_i} \right) \tag{B.13}$$

which is distributed as χ^2 with $\mathrm{N} - \mathrm{M}$ degrees of freedom, where the model m_i has M parameters (include the normalization).

Poisson data with Gaussian background (pgstat)

Another possible background option is if the background spectrum is not Poisson. For instance, it may have been generated by some model based on correlations between the background counts and spacecraft orbital position. In this case there may be an uncertainty associated with the background which is assumed to be Gaussian. In this case the same technique as above can be used to derive a profile likelihood statistic :

$$PG = 2\sum_{i=1}^{N} t_s(m_i + f_i) - S_i \ln\left(t_s m_i + t_s f_i\right) + \frac{1}{2\sigma_i^2}(B_i - t_b f_i)^2 - S_i(1 - \ln S_i) \tag{B.14}$$

where

$$f_i = \frac{-(t_s\sigma_i^2 - t_b B_i + t_b^2 m_i) \pm d_i}{2t_b^2} \tag{B.15}$$

and

$$d_i = \sqrt{[t_s\sigma_i^2 - t_b B_i + t_b^2 m_i]^2 - 4t_b^2[t_s\sigma_i^2 m_i - S_i\sigma_i^2 - t_b B_i m_i]} \tag{B.16}$$

There is a special case for any bin with S_i equal to zero:

$$PG_i = t_s m_i + B_i(t_s/t_b) - \sigma_i(t_s/t_b)^2/2 \tag{B.17}$$

This is what is used for the **statistic pgstat** option.

Poisson data with known background (pstat)

Another possible background option is if the background spectrum is known. Again the same technique as above can be used to derive a profile likelihood statistic :

$$P = 2\sum_{i=1}^{N} t_s(m_i + B_i/t_b) - S_i \ln\left[t_s(m_i + B_i/t_b\right] - S_i(1 - \ln S_i) \tag{B.18}$$

This is what is used for the **statistic pstat** option.

Bayesian analysis of Poisson data with Poisson background (lstat)

An alternative approach to fitting Poisson data with background is to use Bayesian methods. In this case instead of solving for the background rate parameters we marginalize over them writing the joint probability distribution of the source parameters as :

$$P = p\left(\{\theta_j\}|\{S_i\},\{B_i\},I\right) = \int ... \int \{db_k\} p\left(\{\theta_j\},\{b_k\}|\{S_i\},\{B_i\},I\right) \tag{B.19}$$

where $\{\theta_j\}$ are the source parameters, $\{b_k\}$ the background rate parameters and I any prior information. Using Bayes theorem, that the $\{\theta_j\}$ and independent

of the $\{b_k\}$, that the $\{b_k\}$ are individually independent and that the observed counts are Poisson gives :

$$P = \frac{p\left(\{\theta_j\}|I\right)}{p\left(\{S_i\}|I\right)p\left(\{B_i\}|I\right)} \prod_{k=1}^{N} \frac{t_s^{S_k} t_b^{B_k} \mathrm{e}^{-m_k t_s}}{S_k! B_k!} J_k \tag{B.20}$$

where :

$$J_k = \int db_k p(b_k|I)(m_k + b_k)^{S_k} b_k^{B_k} \mathrm{e}^{-b_k(t_s+t_b)} \tag{B.21}$$

To calculate J_k we need to make an assumption about the prior background probability distribution, $p(b_k|I)$. We follow Loredo (1992) and assume a uniform prior between 0 and b_i^{max}. Expanding the binomial gives :

$$J_k = \frac{1}{b_k^{max}} \sum_{j=0}^{S_k} m_k^j \frac{S_k!}{j!(S_k-j)!} \frac{\gamma\left(S_k + B_k - j + 1, b_k^{max}(t_s+t_b)\right)}{(t_s+t_b)^{S_k+B_k-j+1}} \tag{B.22}$$

where :

$$\gamma(\alpha,\beta) = \int_0^\beta x^{(\alpha-1)} \mathrm{e}^{-x} dx \tag{B.23}$$

Again, following Loredo we assume that $(t_s + t_b)b_k^{max} >> B_k$ and using the approximation $\gamma(\alpha,\beta) \sim (\alpha-1)!$ when $\alpha >> \beta$ gives :

$$J_k = \frac{S_k!(t_s+t_b)^{-(S_k+B_k+1)}}{b_k^{max}} \sum_{j=0}^{S_k} m_k^j \frac{(S_k+B_k-j)!}{j!(S_k-j)!} (t_s+t_b)^j \tag{B.24}$$

Note that for $m_k = 0$ only the $j = 0$ term in the summation is non-zero. Now, we define lstat by calculating $-2\ln P$ and ignoring all additive terms which are independent of the model parameters :

$$\text{lstat} = -2\ln p\left(\{\theta_j\}|I\right) + 2\sum_{k=1}^{N} \left(m_k t_s - \ln \sum_{j=0}^{S_k} m_k^j \frac{(S_k+B_k-j)!}{j!(S_k-j)!} (t_s+t_b)^k \right) \tag{B.25}$$

Power spectra from time series data (whittle)

XSPEC has been used by a number of researchers to fit models to power spectra from time series data. In this case the x-axis is frequency (in Hz) and not keV so plots have to be modified appropriately. The correct fit statistic is that due to Whittle as discussed in Vaughan (2010) and Barret & Vaughan (2012) :

$$S = 2\sum_{i=1}^{N}\left(\frac{y_i}{m_i} + \ln m_i\right) \tag{B.26}$$

B.3 Parameter confidence regions

Fisher Matrix

XSPEC provides several different methods to estimate the precision with which parameters are determined. The simplest, and least reliable, is based on the inverse of the second derivative of the statistic with respect to the parameter at the best fit. The first derivative must be zero by construction and the second derivative provides a measure of how rapidly the statistic increases away from the best-fit. The faster the statistic increases, i.e. the larger the second derivative, the more precisely the parameter is determined. The matrix of second derivatives is often referred to as the Fisher information. Its inverse is the covariance matrix, written out at the end of an XSPEC fit.

The +/- numbers provided for each parameter in the standard fit output are estimates of the one-sigma uncertainty, calculated as the square root of the diagonal elements of the covariance matrix. As such, these ignore any correlations between parameters. Whether correlations are important can be seen by comparing with the off-diagonal elements of the covariance matrix. In general, these estimates should be considered lower limits to the true uncertainty.

Correlation information is also given in the table of variances and principal axes which also appears at the end of a fit. Each row in this table is an eigenvalue and associated eigenvector of the Fisher matrix. If the parameters are independent then each eigenvector will have a contribution from only one parameter. For instance, if there are three independent parameters then the eigenvectors will be (1,0,0), (0,1,0), and (0,0,1). If the parameters are not independent then each eigenvector will show contributions from more than one parameter.

Delta Statistic

The next most reliable method for deriving parameter confidence regions is to

find surfaces of constant delta statistic from the best-fit value, i.e. where :

$$\text{Statistic} = \text{Statistic}_{\text{best-fit}} + \Delta \tag{B.27}$$

This is the method used by the **error** command, which searches for the parameter value where the statistic differs from that at the best fit by a value (Δ) specified in the command. For each value of the parameter being tested all other free parameters are allowed to vary. The results of the **error** command can be checked using **steppar**, which can also be used to find simultaneous confidence regions of multiple parameters. The specific values of Δ which generate particular confidence regions are calculated by assuming that it is distributed as χ^2 with the number of degrees of freedom equal to the number of parameters being tested (e.g. when using the **error** command there is one degree of freedom, when using **steppar** for two parameters followed by **plot contour** there are two degrees of freedom). This assumption is correct for the S^2 statistic and is asymptotically correct for other statistic choices.

Monte Carlo

The best but most computationally expensive methods for estimating parameter confidence regions are using two different Monte Carlo techniques. The first technique is to start with the best fit model and parameters and simulate datasets with identical properties (responses, exposure times, etc.) to those observed. For each simulation, perform a fit and record the best-fit parameters. The sets of best-fit parameters now map out the multi-dimensional probability distribution for the parameters assuming that the original best-fit parameters are the true ones. While this is unlikely to be true, the relative distribution should still be accurate so can be used to estimate confidence regions. There is no explicit command in XSPEC to use this technique however it is easy to construct scripts to perform the simulations and store the results.

The second technique is Markov Chain Monte Carlo (MCMC) and is of much wider applicability. In MCMC a chain of sets of parameter values is generated which describe the parameter probability distribution. This determines both the best-fit (the mode) and the confidence regions. The **chain** command runs MCMC chains which can be converted to probability distributions using **margin** (which takes the same arguments as **steppar**). The results can be plotted in 1- or 2-D using **plot margin** and **plot integprob** to plot the probability density and integrated probability. If MCMC chains are in use then the **error** command will use them to estimate the parameter uncertainty.

B.4 Goodness-of-fit

Parameter values and confidence regions only mean anything if the model actual fits the data. The standard way of assessing this is to perform a test to reject the null hypothesis that the observed data are drawn from the model. Thus we calculate some statistic T and if $T_{obs} > T_{critical}$ then we reject the model at the confidence level corresponding to $T_{critical}$. Ideally, $T_{critical}$ is independent of the model so all that is required to evaluate the test is a table giving $T_{critical}$ values for different confidence levels. This is the case for χ^2 which is one of the reasons why it is used so widely. However, for other test statistics this may not be true and the distribution of T must be estimated for the model in use then the observed value compared to that distribution. This is done in XSPEC using the **goodness** command. The model is simulated many times and a value of T calculated for each fake dataset. These are then ordered and a distribution constructed. This distribution can be plotted using **plot goodness**. Now suppose that T_{obs} obs exceeds 90% of the simulated T values we can reject the model at 90% confidence.

It is worth emphasizing that goodness-of-fit testing only allows us to reject a model with a certain level of confidence, it never provides us with a probability that this is the correct model.

Chi-square (chi)

The standard goodness-of-fit test for Gaussian data is χ^2 (as defined above). At the end of a **fit**, XSPEC writes out the reduced χ^2 ($= \chi^2/\nu$, where ν is the number of degrees of freedom = number of data bins minus number of free parameters). A rough rule of thumb is that the reduced χ^2 should be approximately one. If the reduced χ^2 is much greater than one then the observed data are likely not drawn from the model. If the reduced χ^2 is much less than one then the Gaussian sigma associated with the data are likely over-estimated. XSPEC also writes out the null hypothesis probability, which is the probability of the observed data being drawn from the model given the value of χ^2 and the number of degrees of freedom.

Pearson chi-square (pchi)

Pearson's original (1900) chi-square test was not for Gaussian data but for the case of dividing counts up between cells. This corresponds to the case of Poisson data with no background.

$$\chi^2_P = \sum_{i=1}^{N} \frac{(y_i - m_i)^2}{m_i} \tag{B.28}$$

Kolmogorov-Smirnov (ks)

There are a number of test statistics based on the empirical distribution function (EDF). The EDF is the cumulative spectrum :

$$Y_i = \left(\sum_{j=1}^{i} y_J\right) / \left(\sum_{j=1}^{N} y_j\right) \tag{B.29}$$

for the data and

$$M_i = \left(\sum_{j=1}^{i} m_J\right) / \left(\sum_{j=1}^{N} m_j\right) \tag{B.30}$$

for the model.

The EDF can be plotted using **plot icounts**. The best known of these tests is Kolmogorov-Smirnov whose statistic is simply the largest difference between the observed and model EDFs :

$$D = supremum|Y_i - M_i| \tag{B.31}$$

The XSPEC **statistic test ks** option returns $\log D$. The significance of the ks value can be determined using the **goodness** command.

Cramer-von Mises (cvm)

The Cramer-von Mises statistic is the sum of the squared differences of the EDFs :

$$w^2 = \sum_{i=1}^{N} (Y_i - M_i)^2 \tag{B.32}$$

The XSPEC **statistic test cvm** option returns $\log w^2$ and its significance should be determined using the **goodness** command.

Anderson-Darling (ad)

Anderson-Darling is a modification of Cramer-von Mises which places more weight on the tails of distribution :

$$w^2 = \sum_{i=1}^{N} \frac{(Y_i - M_i)^2}{M_i(1 - M_i)} \tag{B.33}$$

Runs (runs)

The Runs (or Wald-Wolfowitz) test checks that residuals are randomly distributed above and below zero and do not cluster. Suppose N_p is the number of channels with +ve residuals, N_n the number of channels with negative residuals, and R the number of runs then the Runs statistic is :

$$Runs = (R - \mu)/\sqrt{[(\mu - 1)(\mu - 2)/(N - 1)]} \tag{B.34}$$

where :

$$N = N_p + N_n \tag{B.35}$$

and

$$\mu = \frac{2N_pN_n}{N} + 1 \tag{B.36}$$

The hypothesis that the residuals are randomly distributed can be rejected if abs(Runs) exceeds a critical value. For large sample runs (where N_p and N_n both exceed 10) the critical value is drawn from the Normal distribution. For instance, for a test at the 5% significance level, the hypothesis can be rejected if abs(Runs) exceeds 1.96.

References

- Barret, D. & Vaughan, S., 2012. "Maximum likelihood fitting of X-ray power density spectra: application to high-frequency quasi-periodic oscillation from the neutron star X-ray binary 4U1608-522", ApJ 746, 131.
- Cash, W., 1979. "Parameter estimation in astronomy through application of the likelihood ratio", ApJ 228, 939.
- Humphrey, P.J., Liu, W. and Buote, D.A., "χ^2 and Poissonian Data: Biases Even in the High-Count Regime and How to Avoid Them.", ApJ 693, 822.

- Loredo, T., 1992. In "Statistical Challenges in Modern Astronomy", eds. Feigelson, E.D. and Babu, G.J., pp 275-297.
- Siemiginowska, A., 2011. In "Handbook of X-ray Astronomy" eds. Arnaud, K.A., Smith, R.K. and Siemiginowska, A., Cambridge University Press, Cambridge.
- Vaughan, S. 2010, "A Bayesian test for periodic signals in red noise", MNRAS 402, 307.

Appendix C

Adding Models to XSPEC

XSPEC includes a large collection of standard models that can be fit to data. However, sometimes these are not enough and a new model might be required. In order of increasing complexity the ways to do this are: use the **mdefine** command; create a table model; load a model function created by someone else; create and load your own model function. The **mdefine** command can be used for a model which can be described using a simple formula and is documented under the commands section of the manual so we do not discuss it further. This appendix describes the other three methods then finishes with a note about the more complex issue of mixing models.

C.1 Table models

A very simple way of fitting with user-defined models is available for a particular class of models. These are models that can be defined by a grid of spectra, with the elements of the grid covering the range of values of the parameters of the model. For instance, for a one-parameter model, a set of model spectra can be tabulated for different values of the parameter (P1, P2, P3, etc.) The correct model spectrum for a value P is calculated by interpolation on the grid. The generalization to more parameters works in the obvious way. The table is specified in the **model** command by the special strings **atable**, **mtable**, or **etable** with the filename following in brackets – see the entries in the models section of the manual. Any number of table model components can be used simultaneously.

Table model components can be much slower than most standard models if there

are significant numbers of parameters. The memory requirements increase as 2^n where n is the number of parameters in the model. A table model with more than 3 or 4 fitting parameters is not recommended. Additionally, the interpolation is linear, which implies that the second derivatives used by the default Levenberg-Marquadt algorithm may not be accurate. If the fit does not work well it may be worth trying the migrad (minuit library) algorithm which makes no assumptions about the second derivative.

As with standard models, the spectra should be in terms of flux-per-bin and not flux-per-keV. Any set of energy bins can be used, and XSPEC will interpolate the model spectra onto the appropriate energy bins for the detectors in use. It is therefore a good idea to choose energy bins such that the spectrum is well-sampled over the range of interest. The file structure for these models is a FITS format described at :
http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/general/ogip_92_009/ogip_92_009.html
or:
ftp://legacy.gsfc.nasa.gov/fits_info/fits_formats/docs/general/ogip_92_009

The HEASP C++ library and Python module available in HEAsoft can be used to make table model files. The documentation at :
http://heasarc.gsfc.nasa.gov/docs/software/lheasoft/headas/heasp/heasp_guide.html
includes examples for both C++ and Python.

C.2 Loading a new model function

New model functions either downloaded from the XSPEC additional models webpage at :
http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/newmodels.html
or acquired privately are added using the two commands **initpackage**, which prepares and compiles a library module containing them, and **lmod**, which actually loads them into the program. These commands are described in the XSPEC commands section of the manual, to which the user is referred. Any number of different user model packages may be added to XSPEC from the user prompt, and the user has control over the directory from which models are loaded.

Note that the **lmod** command requires write-access to the particular directory specified. This is because lmod uses the Tcl **make package** and **package require** mechanisms for automatic library loads and these require Tcl write an index file (`pkgIndex.tcl`) to the directory. Consequently we recommend using

the Tcl **load** command instead of **lmod** if the library is being used by a number of users on a local network. Note that such a library can be loaded automatically by placing the command in the `global_customize.tcl` script (see the section "Customizing XSPEC" in the Overview (Chapter 3)).

C.3 Writing a new model function

A model function is a subroutine that calculates the model spectrum given an input array of energy bins and an array of parameter values. The input array of energy bins gives the boundaries of the energy bins and hence has one more entry than the output flux arrays. The energy bins are assumed to be contiguous and will be determined by the response matrix in use. The subroutine should thus make no assumptions about the energy range and bin sizes. The output flux array for an additive model should be in terms of photons/cm^2/s (not photons/cm^2/s/keV) i.e. it is the model spectrum integrated over the energy bin. The output array for a multiplicative model is the multiplicative factor for that bin. Convolution models are operators on the output from additive or multiplicative models. Model subroutines can be written in fortran, either in single or double precision, in C++ using either C++-style arguments or C style arguments, and in C.

The `model.dat` entry

In addition to the subroutine, XSPEC requires a text file describing the model and its parameters. The standard models are specified in the `model.dat` file so we usually refer to this text file by that name. A sample `model.dat` entry has the following form:

```
modelentry        5  0.    1.e20      modelfunc    add  0 0
lowT    keV     0.1   0.0808  0.0808 79.9      79.9       0.001
highT   keV     4.    0.0808  0.0808 79.9      79.9       0.001
Abundanc " "    1.    0.      0.      5.        5.        0.01
*redshift " "   0.0
```

The first line for each model gives the model name, the number of parameters, the low and high energies for which the model is valid, the name of the subroutine to be called and the type of model (add, mul, mix, or con, or acn). The final two arguments are flags: the first should be set to 1 if model variances are calculated by modelfunc and the second should be set to 1 if the model should be forced to perform a calculation for each spectrum. This final flag is necessary because if multiple spectra have the same energy bins, the default behavior is to perform the model calculation for just one spectrum and copy the results for each of

the others. However if a model depends on information about the spectrum in addition to its energy ranges, it must be forced to perform a calculation for each spectrum.

The remaining lines in the text file specify each parameter in the model. For regular model parameters the first two fields are the parameter name followed by an optional units label. If there is no units label then there must be a quoted blank (“ ”) placeholder. The remaining 6 numerical entries are the default parameter value, hard min, soft min, soft max, hard max, and fit delta, which are described in the **newpar** command section.

There are three special types of parameter which can be used. If the name of the parameter is prefixed with a “*” the parameter is a “scale” parameter and cannot be made variable or linked to any kind of parameter other than another scale parameter. Since the parameter value can never vary only the initial value need be given. If the name of the parameter is prefixed with a “$” the parameter is a “switch” parameter which is not used directly as part of the calculation, but switches the model component function’s mode of operation (i.e. calculate or interpolate). Switch parameters only have 2 fields: the parameter name and an integer value.

Finally, If a P is added at the end of the line for a parameter then the parameter is defined to be periodic. During a fit, a periodic parameter will not be pegged if it tries to exceed its hard limits. Instead it will be assigned a value within its limits: f(max + delta) = f(min + delta), f(min-delta) = f(max-delta). The soft min and max settings are irrelevant for period parameters and will be ignored.

The model subroutine function

The following table lists the function arguments required for the different language options. The second column is the way the function name should be included in the `model.dat` entry.

Call Type and Specification	Arguments and Type	Meaning
Single precision fortran	real*4 ear(0:ne)	Energy array
	integer ne	Size of flux array
	real*4 param(*)	Parameter values. (Dimension must be specified inside the function)
funcName	integer ifl	The spectrum number being calculated
	real*4 photar(ne)	Output flux array
	real*4 photer(ne)	Output flux error array (optional)
Double precision fortran	real*8 ear(0:ne)	As above
	integer ne	As above
	real*8 param()	As above
F_funcName	integer ifl	As above
	real*8 photar(ne)	As above
	real*8 photer(ne)	As above
C/C++, C-style	const Real* energy	Energy array (size Nflux+1)
	int Nflux	Size of flux array
	const Real* parameter	Parameter values
c_funcName	int spectrum	Spectrum number of model component being calculated
	Real* flux	Output flux array
	Real* fluxError	Output flux error array (optional)
	const char* init	Initialization string (see below)
C++, C++ style	const RealArray& energy	Energy array
	const RealArray& parameter	Parameter values
	int spectrum	Spectrum number of model component being calculated
C_funcName	RealArray& flux	Output flux array
	RealArray& fluxError	Output flux error array (optional)
	const string& init	Initialization string (see below)

For example, a model component in double precision fortran is specified by:

```
modelentry        5  0.    1.e20     F_funcName    add  0 0
```

XSPEC then picks out the right function definition, and calls the Fortran function `funcName` which expects double precision arguments. The C-style call can clearly be compiled and implemented by either a C or a C++ compiler: however we recommend using the C++ call if the model is written in C++ as it will reduce overhead in copying C arrays in and out the XSPEC internal data structures. To prevent unresolved symbol linkage errors, we also recommend prefacing C++ local model function definitions with the `extern ``C'' directive`.

Example C/C++ function definitions:

```
/* C -style */
extern ``C''
void modelfunc(const Real* energy, int Nflux, const Real* parameter,
               int spectrum, Real* flux, Real* fluxError,
               const char* init)
{
/* Model code:  Do not allocate memory for flux and fluxError arrays.
XSPEC's C-function wrapper will allocate arrays prior to calling the
function (and will free them afterwards). */
}
// C++
extern ``C''
void modelfunc(const RealArray& energy, const RealArray& parameter,
               int spectrum, RealArray& flux, RealArray& fluxError,
               const string& init)
{
// Model code:  Should resize flux RealArray to energy.size()-1.
// Do the same for fluxError array if calculating errors, otherwise
// leave it at size 0.
}
```

Note on type definitions for (C and C++): XSPEC provides a typedef for Real, in the `xsTypes.h` header file. The distributed code has

```
typedef Real double;
```

i.e. all calculations are performed in double precision. This is used for C models and C++ models with C-style arguments.

The type RealArray is a dynamic (resizeable) array of Real. XSPEC uses the std::valarray template class to implement RealArray. The internal details of

XSPEC require that the RealArray typedef supports vectorized assignments and mathematical operations, and indirect addressing (see C++ documentation for details). However, we do not recommend using specific features of the std::valarray class, such as array slicing, in case the typedef is changed in future.

The input energies are set by the response matrices of the detectors in use. IFL is an integer which specifies to which response (and therefore which spectrum) these energies correspond. It exists to allow multi-dimensional models where the function might also depend on eg pulse-phase in a variable source. The output flux array should not be assumed to have any particular values on input. It is assumed to contain previously calculated values only by convolution/pileup models, which have the nature of operators. The output flux error array allows the function to return model variances.

The C and C++ call types allow one extra argument, which is a character string that can be appended to the top line of the model component description. This string is read on initialization and available to the model during execution. An example of its use might be the name of a file with specific data used in the model calculation: this allows different models to be implemented the same way except for different input data by specifying different names and input strings.

Function utility routines

There are a number of internal XSPEC routines which may be useful for local models written in either C++ or Fortran. These are defined in `FunctionUtility.h` and `xsFortran.cxx`, respectively, in the directory `Xspec/src/XSUtil/FunctionUtils`. Documentation is available in the file `XSUtil_guide.pdf` in `Xspec/src/help`.

These functions allow the model to find out the Solar abundances in use, get any information defined using the **xset** command, find the cosmological parameters currently set, get the contents of the XFLT#### keywords in the spectrum files, and load information into an internal database which can be accessed using **tclout**.

There are also C++ classes and associated methods for equilibrium and non-equilibrium collisional plasmas, photo-electric opacity from neutral and ionized materials, and Compton scattering. Descriptions of these can be found in `XSFunctions_guide.pdf` in `Xspec/src/help`.

Multi-dimensional models

Typically an XSPEC model depends only on the energy (and the model parameters) but more complicated models are sometimes required. The model might depend on energy and, for a variable source, the time at which the spectrum was obtained or, for an extended source, its spatial position. The extra information about the spectrum must be stored in the XFLT#### keywords in the spec-

trum files. For instance, if the model has a dependence on time then for one spectrum XFLT0001 could be set to "start: 1345.54" and XFLT0002 to "end: 1356.43" and another have XFLT0001 set to "start: 1356.43" and XFLT0002 to "end: 1368.21". An example C++ routine might then include the code

```
#include <XSUtil/FunctionUtils/FunctionUtility.h>
...
extern ``C''
void mymodel(const RealArray& energyArray, const RealArray& params,
             int spectrum, RealArray& flux, RealArray& fluxErr,
             const string& initString)
...
    Real tstart = FunctionUtility::getXFLT(spectrum,``start'');
    Real tend = FunctionUtility::getXFLT(spectrum,``end'');
...
```

C.4 Third-Party Libraries In Local Models Build

The Makefile that initpackage creates for building your local models library is based on the template file

```
heasoft-[ver]/Xspec/src/tools/initpackage/xspackage.tmpl
```

If you need to add a path to a third-party library's header files, add:

```
-I/path/to/your/3rdParty/library/include
```

to the HD_CXXFLAGS setting. Then type **hmake** and **hmake install** from the

```
heasoft-[ver]/Xspec/src/tools/initpackage
```

directory.

To make sure the linker pulls in the library on Mac OS X further edit the `xspackage.tmpl` file by adding a "-l" flag for the library (e.g. -lgsl) in the HD_SHLIB_LIBS settings. Then reinstall `xspackage.tmpl` as mentioned above. On Linux/Unix the XSPEC executable itself should be relinked with the new library included. So, edit the file

```
heasoft-[ver]/Xspec/src/main/Makefile
```

by adding a "-l" flag for the library to the HD_CXXLIBS setting. Then from the same directory do:

```
rm xspec
hmake local
hmake publish
hmake install
```

After these modifications, you should be able to use **initpackage** and **lmod** in the normal way to build and load your local models library.

C.5 Writing new mixing models

Mixing models are fundamentally different from the other kinds of models since they apply a transformation to a set of modeled fluxes (as enumerated by the spectra in the fit), rather than modify the flux designed to fit a single spectrum. The need to store temporary results, as well as the requirements of the model calculation, lead to many workspace arrays: further, the transformations applied are often fixed during a fit, or can be split to avoid redundant calculations into parts that are fixed and parts that change during iteration in order. XSPEC's internal organization (data structures) can be mapped straightforwardly to the requirements of these models so to implement them efficiently and handle memory allocation, we recommend that mixing models be written in C++ or C. At present only a C++ implementation is available. Users considering adding new mixing model types should contact the developers of XSPEC at xspec12@athena.gsfc.nasa.gov.

Appendix D

Overview of PLT

XSPEC uses the PLT library, which is in turn based on PGPLOT[1], to implement its plotting capabilities.

Extensive documentation for the PLT graphics routine is available in the The QDP/PLT Users's Guide and from PLT's interactive help. This appendix is intended to provide information to assist in using PLT from within the XSPEC program.

Within XSPEC, it is possible to set your graphics device using the **cpd** command. Any PGPLOT device supported by your local version of PGPLOT is accepted. The **cpd** command can also be used to display a list of all PGPLOT devices. If you fail to enter a device name, you will be prompted for a PGPLOT device every time you generate a new plot.

From XSPEC, there are two ways to call the PLT routine. Both have the same syntax which is described in the corresponding manual section.

The **plot** <`plot mode`> command will produce a graph and control will return immediately to XSPEC.

The **iplot** <`plot mode`> command will put XSPEC into interactive plot mode. The prompt

```
PLT>
```

[1]PGPLOT is the name of a Graphics Subroutine Library written by T. J. Pearson at the California Institute of Technology

will appear after XSPEC has finished producing the graph. At this point, you can enter PLT commands to inspect interesting parts of the graph, add labels, or make a hardcopy file for later printing.

D.1 Getting started with PLT

In the following description of PLT commands, the full command is described. Capital letters denote the shortest abbreviation of the command that will be recognized. Here is a brief guide to some of the PLT commands that can be entered when **iplot** is invoked.

- **HElp** will provide you with descriptions of the PLT commands.
- **Plot** redraws the display using all of the commands that change the graph entered since the last plot.
- **Rescale** followed by two numbers, will set the minimum and maximum of the plotted x-range to the numbers specified. Without further arguments, **Rescale X** or **Rescale Y** will reset the minimum and maximum values to their default values. Rescale also updates the screen immediately. Other commands allow you to make several changes to the the graph without having to wait for the screen to be updated after every change.
- **LAbel** <`Top,X,Y`> [<`string`>] adds labels to various locations on the graph. For example, typing **LA Top EXOSAT was great** will cause the message "EXOSAT was great" to appear at the top of the graph the next time the display is redrawn. Without the string argument the current label for Top, X, or Y is set to the empty string.
- **Hardcopy** [**?, PGPLOT plot device** creates a file that can later be printed. Since it redraws the graph and sends it to a file, it does not reproduce what currently is visible on the graphics display, but rather what you would see if you re-issued the **Plot** command. With the optional "?" argument, **Hardcopy** returns the current hardcopy plotting device. This can be overridden with **Hardcopy** [**PGPLOT device name**].
- **EXit** returns control to XSPEC. Any changes you have made to the plot will be lost.

D.2 PLT Command summary

CLear	Immediately clear the graphics device

COlor	Change the default colour index
CONtour	Produce a contour plot
CPD	Change the plotting device
CQuit	Clear the graphics device and return control to XSPEC
CSize	Change the default character size
Error	Control whether errors are displayed and used in fitting
EXit	Exit PLT and return control to XSPEC
Fit	Fit the PLT model to the data
FNy	Evaluate the model at the specified location
FOnt	Change the default text font
Freeze	Freeze a parameter value
GAp	Change the default gap size between the data and the edge
Grid	Control the location of the major and minor tic marks
Hardcopy	Make a file that can later be printed
HElp	Obtain help on any PLT command
Imodel	Numerically integrate the model over specified range
LAbel	Add or remove labels from the plot
LIne	Control whether a line is used to connect data points
LOg	Control whether data is plotted using a log10 scale
LStyle	Change the default style of the line connecting the data points
LWidth	Change the default line width
MArker	Control whether the data points are plotted with markers
MOdel	Define a PLT model
Newpar	Change a parameter value associated with the model
PLot	Immediately re-plot the data
PRompt	Change the PLT⟩ prompt
Rescale	Reset the minimum and maximum plot range
SCr	Change the color representation of the specified color index
SHow	Display the values of PLT internal variables
SKip	Control how PLT divides data into vectors
STatistics	Compute various statistical properties of the data
THaw	Allow a parameter value to vary during a fit
Time	Control whether the time stamp is plotted
Uncertainty	Compute the uncertainty in a parameter value
VErsion	Display date of the most recent modification to PLT

Viewport	Control the size of the viewport plotting area
WData	Write a QDP data file to disk
WEnviron	Write both QDP data and header files to disk
WHead	Write a QDP header file to disk
WModel	Write a model file to disk
Xaxis	Define the method used to calculate the x-variable
Yaxis	Define the y-axis scale for a contour plot
$	Execute operating system commands
@filename	Read commands from a PLT command (.pco) file

Appendix E

Associated Programs

E.1 Introduction

The HEAsoft package provides a number of programs and subroutine libraries to manipulate the FITS files used by XSPEC. A description of most tasks can be obtained by typing **fhelp taskname** or to get a complete list **fhelp ftools**.

E.2 HEAsoft reading tasks

ftlist	Prints the contents of a FITS file to the screen or to a file
dmprmf	Prints the contents of a FITS RMF file to the screen or to a file.This tool prints the RMF file in a more legible fashion than ftlist

E.3 HEAsoft manipulation tasks

fparkey	Changes the value of a keyword in a FITS extension header
grppha	Defines (or redefines) and/or displays the grouping and quality flags, the important keywords, and the fractional systematic errors.
rbnpha	Compresses a FITS PHA file to a user-defined number of channels. The output is a new file containing the revised PHA extension plus a direct copy of any other extensions in the original file.
mathpha	Performs arithmetical operations on PHA files.
cmppha	Convert a type II pha file to a type I pha file.
rbnrmf	Bins a FITS RMF file (the detector response matrix) in channel or energy space.
cmprmf	Compress an RMF by removing all response below a threshold value.
addarf	Adds together ARFs.
addrmf	Adds together RMFs.
marfrmf	Multiplies an RMF file by an ARF file.
genrsp	A generic spectral response generator.
fv	A general GUI for manipulating FITS files.

E.4 HEAsoft subroutines

The heasp library in `heacore/heasp` is a C++ class library for manipulating all the filetypes used by XSPEC. This provides both C++ and C routines as well as a Python package. A complete description is available at: http://heasarc.gsfc.nasa.gov/docs/software/lheasoft/headas/heasp/heasp_guide.html

Appendix F

Using the XSPEC Models Library in Other Programs

For those who wish to incorporate the standard XSPEC model functions library into their own programs, XSPEC provides a set of functions and wrappers that can be called from external C, C++ or Fortran programs.

F.1 Calling Model Functions From C And Fortran

An increasing number of XSPEC model functions are written in C++, and have the C++-style function interface described in Appendix C. XSPEC provides function wrappers for each of these to make them callable from Fortran or C programs. The wrappers are stored in the files `funcWrappers.h` and `funcWrappers.cxx` in the `XSFunctions` directory.

For each C++ model function there are 2 wrappers: one for passing single precision arrays and one for double precision, with the interfaces as shown in Appendix C for single precision Fortran-style and C-style respectively. The single precision wrapper function name will be the original C++ function name appended with a "f_" prefix, while the double precision wrapper will have a "C_" prefix.

For example, XSPEC's `model.dat` entry for the power law model lists the function name C_powerLaw. This shows that the actual function name is

"powerLaw" and the "C_" indicates it has a C++ interface inside XSPEC. `funcWrappers.cxx` defines the following 2 wrappers:

```
void f_powerLaw(const float* energy, int nFlux, const float* params,
                int spectrumNumber, float* flux, float* fluxError)
void C_powerLaw(const double* energy, int nFlux, const double* params,
                int spectrumNumber, double* flux, double* fluxError,
                const char* initStr)
```

The second function is intended to be called from C programs, while Fortran programs may call either (`funcWrappers.cxx` also includes CERN <`cfortran.h`> definitions to make these accessible to Fortran).

F.2 Interface Routines

XSPEC also provides a set of functions for accessing some of the model functions' internal data. The C++ functions are listed in the file `FunctionUtility.h` in the `XSUtil/FunctionUtils` directory. For C and Fortran access, equivalent wrapper functions are listed in the same directory in `xsFortran.h`. The wrapper functions have C-style function declarations, and are also made available to Fortran calling routines via the CERN <`cfortran.h`> interface.

The currently provided C/Fortran wrapper functions are (see `xsFortran.h` for the function signatures):

FNINIT	Initializes data directory locations needed by the models. See below for a fuller description.
FGABND	Get an element abundance.
FGCHAT	Get current chatter level setting for model functions' output verbosity.
FPCHAT	Set the chatter level. Default is 10, higher chatter levels produce more output.
FGDATD	Get the model .dat files path.
FPDATD	Set the model .dat files path.
FGMODF	Get the model ion data path.
FGMSTR	Get a model string value (see XSPEC xset command).
FPMSTR	Set a model string value.
FPSLFL	Load values of a file solar abundance table (see **abund** command).
FGSOLR	Get the solar abundance table setting.
FPSOLR	Set the solar abundance table.
FGXSCT	Get the cross section table setting.
FPXSCT	Set the cross section table.
RFLABD	Read abundance data from a file, then load and set this to be the current abundance table. (Essentially this combines a file read with the FPSLFL and FPSOLR functions.)
csmgh0	Get the cosmology H_0 setting (see the cosmo command).
csmph0	Set H_0.
csmgl0	Get Λ_0.
csmpl0	Set Λ_0.
csmgq0	Get q_0.
csmpq0	Put q_0.
fzsq	Computes the luminosity distance, (c/H_0)*fzsq. The function is valid for small values of q_0*z for the case of no cosmological constant and uses the approximation of Pen (1999 ApJS 120, 49) for the case of a cosmological constant and a flat Universe. The function is not valid for non-zero cosmological constant if the Universe is not flat.
DGFILT	Get a particular XFLT keyword value from a data file.
DGNFLT	Get the number of XFLT keywords in a data file.
xs_getVersion (or xgvers)	Retrieve XSPEC's version string.

F.3 Initializing the Models Library

The external program should always call the `FNINIT` routine prior to any other call into the models library. This initializes the locations of the various data files needed by the models, and also sets the abundance and cross-section tables. Unless the user has overridden the model ion data directory location with the XSPEC_MDATA_DIR environment variable, the initial settings are:

Model ion data location	$HEADAS/../spectral/modelData
Abundance .dat files location	$HEADAS/../spectral/manager
Cross-section .dat files location	$HEADAS/../spectral/manager
Solar abundance table	angr
Photoelectric cross-section table	bcmc

F.4 Building with the Models Library

The XSFunctions library depends on three lower-level XSPEC libraries, `XS`, `XSUtil`, and `XSModel`, and also the `CCfits` and `cfitsio` libraries distributed with HEASOFT. A Makefile for a small Fortran program linking with the models library therefore may look like this on Linux:

```
myprog : myprog.o
        g77 -g myprog.o -o myprog \
                -L/path/to/headas/installed/location/lib \
                -lXSFunctions -lXSModel -lXSUtil -lXS -lCCfits_2.1 -lcfitsio_3.11
myprog.o: myprog.f
        g77 -g -c myprog.f
```

Appendix G

Adding a Custom Chain Proposal Algorithm

When running a Monte Carlo Markov Chain with the chain command, XSPEC provides several built-in proposal options from which to draw trial parameter values for the next step in the chain. A built-in proposal is selected prior to the chain run with the command:

```
chain proposal <distribution> <source>
```

where <`distribution`> is the statistical distribution used to randomize the parameter values (e.g. gaussian, cauchy), and <`source`> refers to the source of the applied covariance information (see the **chain** command for details).

It is also possible for the user to create an arbitrary new proposal scheme and add it to the options available under the chain proposal command. This is done in a way similar to the adding of local models described in Appendix C, though in this case the code can only be written in C++. Essentially three steps are involved, each described in greater detail below:

- Create a small text file named `randomize.dat`.
- Write a class which inherits from XSPEC's abstract base class RandomizerBase.
- Run XSPEC's **initpackage** and **lmod** commands to build and load the shared library containing the new proposal class(es).

G.1 The `randomize.dat` Initialization File

This file must be placed in the same directory as the user's proposal code files, and plays a role similar to the local models' .dat initialization files, though it has a much simpler structure. It also MUST be named `randomize.dat`, for this name is the only clue **initpackage** has to distinguish between creating a chain proposal or local models library.

All that needs to be entered into this file is a line of the form:

```
<class name>  [<opt string arg1> <opt string arg2> ... <opt string argN>]
```

for each proposal class that will go in the library. <`class name`> must be a case-sensitive match to the actual C++ class name, and is the only required entry on the line. The class should also be stored in code files <`class name`>.h and <`class name`>.cxx.

Any additional arguments on the line will be placed in a single C++ string (including any separating whitespace), and passed to the class constructor. This is to allow the option of setting initialization parameters at the class construction stage. Therefore if optional arguments are included, the class must have a constructor which takes a single string argument. Otherwise, the constructor should contain no arguments.

For example, a `randomize.dat` file declaring two classes might contain:

```
MyProposal1
MyProposal2    1.4773     on      false
```

In the code files, the constructor declarations corresponding to this would then be:

```
MyProposal1.h
class MyProposal1 : public RandomizerBase
{
public:
// ...
MyProposal1();
// ...
};

MyProposal2.h
```

```
class MyProposal2 : public RandomizerBase
{
public:
// ...
MyProposal2(const string& initArgs);
// ...
};
```

G.2 Writing a Chain Proposal Class

All user proposal classes must inherit from XSPEC's abstract class RandomizerBase, whose interface is defined in the file:

```
headas-<version>/Xspec/src/XSFit/Randomizer/RandomizerBase.h.
```

The proposal class must declare a constructor as described in the previous section, and which explicitly calls the RandomizerBase constructor, passing it a lower-case name string. This name will become the proposal identifier when making a selection using the **chain proposal** option during an XSPEC session. For example:

```
MyProposal1.cxx
MyProposal1::MyProposal1()
: RandomizerBase(``myprop1'')
{
}
```

In XSPEC:

```
XSPEC12> chain proposal myprop1 [<optional initializing args>]
```

The RandomizerBase class contains 5 private virtual functions: doRandomize, doInitializeLoad, doInitializeRun, doAcceptedRejected, and getCovariance.

doRandomize

doRandomize is the only pure virtual function and therefore is the only one which must be overridden in the inheriting class. Its signature is:

```
virtual void doRandomize(RealArray& parameterValues, const Fit* fit)
```

where RealArray is a typedef for std::valarray¡double¿ and is defined in `src/main/xsTypes.h`. This function is called by XSPEC for each chain iteration, and XSPEC passes in the current variable model parameter values. The overridden doRandomize function performs the necessary parameter modifications and sends them back in the same array.

The function's second argument is a const pointer to XSPEC's global Fit class object. For those willing to further explore XSPEC's internals, this pointer provides access to various fit and chain information (such as covariance matrices), which may be necessary for the user's proposal scheme.

doInitializeLoad and doInitializeRun

doInitializeLoad and doInitializeRun may be optionally overridden to perform initialization tasks at different stages during runtime. The default versions of these functions in RandomizerBase do nothing. doInitializeLoad is called by XSPEC immediately after the proposal is selected with the **chain proposal** command. Therefore one may find it useful to have this function process any additional arguments which may be entered on the command line:

```
chain proposal myprop [<optional initializing args>]
```

XSPEC automatically bundles [<`optional initializing args`>] into a single string and places it in the m_initString data member of RandomizerBase, to which the inheriting class has access. doInitializeRun is called once at the start of a chain run, and is useful for any tasks which must be performed one time immediately after the **chain run** command is entered.

doAcceptedRejected is called after each iteration in the chain. Its first argument is an array filled with the most recently attempted model parameter values, and its second argument is a boolean true or false indicating whether the attempt was accepted or rejected. The base class function does nothing with this, but an inherited class may want to use this information in an overriden function.

In its simplest form, a proposal class may be declared and defined as in the following example. This doesn't actually do anything since the doRandomize function is empty and the parameterValues array is left unchanged.

```
MyProposal.h
#ifndef MYPROPOSAL_H
#define MYPROPOSAL_H

#include <xsTypes.h>
#include <XSFit/Randomizer/RandomizerBase.h>
```

```
class Fit; // only a forward declaration is required for Fit

class MyProposal : public RandomizerBase
{
public:
MyProposal();
virtual ~MyProposal();
private:
virtual void doRandomize(RealArray& parameterValues, const Fit* fit);
};

#endif

MyProposal.cxx
#include ``MyProposal.h''
#include <XSFit/Fit/Fit.h>

MyProposal::MyProposal()
: RandomizerBase(``myprop'')
{
}

MyProposal::~MyProposal()
{
}

void MyProposal::doRandomize(RealArray& parameterValues, const Fit* fit)
{
// This is where the proposal algorithm should modify the variable
// model parameters in the parameterValues array.
}
```

G.3 Building and Loading the Proposal Class Library

Once the `randomize.dat` file and the class(es) have been written, the library can be built and loaded during an XSPEC session using the same **initpackage** and **lmod** sequence that is used for local model libraries. To create a `Makefile` and build the library:

```
XSPEC12> initpackage <name> randomize.dat <directory>
```

To load the new proposal(s) into XSPEC:

```
XSPEC12> lmod <name> <directory>
```

where <`name`> is the name you choose for the package collection of proposal classes. It will also become the library file name. The only differences from the local models case are that here the initializer file MUST be named `randomize.dat`, and that the directory path to the proposal classes (either relative or absolute) must be provided on the command line. If this is left off XSPEC will default to looking in the directory set by LOCAL_MODEL_DIRECTORY, and these classes should NOT be stored in the same directory as local models. If the building and loading has successfully completed, you should see the proposal name (the same name string that was passed to the RandomizerBase constructor) appear in the chain proposal list displayed by typing **chain proposal** with no other arguments.

Appendix H

Changes between v11 and v12

In 1998 we decided to re-engineer XSPEC using modern computer science methods so it could continue fulfilling its role as a mission-independent X-ray spectral fitting program.

The program's internal design, layout, and data structures have largely been rewritten in ANSI C++ using object oriented design techniques, generic programming techniques, and design patterns. The thoroughgoing reanalysis has also allowed a number of improvements in overall design and, at robustness, as well as maintainability, without changing the familiar syntax. With a few exceptions here and there, the new program syntax is fully backward-compatible with that of v11: most of the exceptions support new features that are enhancements (and can be ignored if not relevant to the user's problems). Some features of v11 previously declared to be deprecated have been removed.

At the same time, the core of the XSPEC calculation scheme has been retained, in particular the models library, written almost exclusively in fortran77.

Model implementation has been rewritten to support models written not only in single precision fortran, but double precision fortran, C, and C++. Further, XSPEC can now be used as a development environment for local models by allowing recompilation from the command prompt.

In v12, spectra can be fit with more than one distinct model simultaneously, provided separate model components can be assigned distinct response functions. This is particularly useful for spectra from coded aperture masks.

A new internal dynamic expression implementation allows more complex (multiply-nested) models, and also allows parameter links to be polynomial functions of one or more parameters.

Great care has been taken to optimize the program for memory usage and execution speed. A revision of the numerical derivative algorithm has reduced the number of convolution operations required during fitting. On the other hand, v12 performs its calculations in double precision (apart from the models library), and this with the more complex model expression evaluations reduces execution speed. Taken together, v12 should outperform v11 when the number of channels is large and the model to be fitted is relatively simple and should be comparable in other circumstances.

The default fitting algorithm (Levenberg-Marquadt) has been retained intact. New fitting algorithms and objective functions (statistics) may be added to the program at runtime. The CERN Minuit/migrad algorithm has been better integrated into the code and its documentation is now directly accessible during XSPEC sessions.

Type II (multi-spectrum) OGIP files are now fully supported. Multiple ranges can be selected in the **data** command, and support is present for Type II background and arf files. Observation simulations (the **fakeit** command) now operate on Type II inputs.

The online documentation scheme is now implemented using pdf or html files, replacing the older VMS-style help system. The help scheme can be configured to use external applications such as Adobe Acrobat or the xpdf readers as well as web browsers. Users can document their own local models and tcl-scripted procedures in pdf and html files and add them to the help system.

Plotting within v12 is backward compatible with a few small extensions. Although it is currently implemented using PLT, explicit dependence on the plot library has been removed. This will allow alternative plotting libraries to be used in future. The PLT plotting package is described briefly in Appendix D and in more detail in the "QDP/PLT User's Guide" (Tennant, 1989).

v12 communicates with the user through the familiar command line interface. The input/output streams, however, can in future be easily redirected to communicate with the user through a graphical user interface (GUI).

Finally, the design implements a new error handling system can return the program safely to the user prompt when an error occurs and leave the program in a state from which the user can continue working. Also, for the first time there is now an **undo** command.

H.1 Integral Spectrometer/Coded Mask Instrument Support

The INTEGRAL Spectrometer (SPI) is a coded-mask telescope, with a 19-element Germanium detector array. There are several complications regarding the spectral analysis of coded-aperture data. For XSPEC the most obvious problem is the source confusion issue; as there may be multiple sources in the FoV leading to different degrees of shadowing on different detectors. Thus, a separate instrumental response must be applied to a spectral model for each possible source, for each detector. If there are multiple sources in the FoV, then additional spectral models can be applied to an additional set of response matrices, enumerated as before over detector and dither pointing. This capability – to model more than one source at a time in a given minimization procedure – did not exist in XSPEC prior to v12. The other unique aspect of the INTEGRAL analysis is that the background is modeled along with the source(s).

XSPEC analysis of INTEGRAL/SPI data is very different from other instruments is the manner in which the response matrices are handled. Since there are a large number of responses involved in the de-convolution problem, memory use becomes a concern. To load the required response matrices (as XSPEC normally does), would require $\sim (N_{ch})^2 \times N_p \times N_d$ floating-point memory locations per source. This could become quite large for high-spectral resolution and/or long observation scenarios. To address this problem, a methodology has been developed to reconstruct the required 2-D response matrices from a basis set, consisting of a small number (3) of 2-D objects (template RMFs), and a larger number of 1-D objects (component ARFs). The full matrices can then be reconstructed "on the fly" at the minimization step of the calculation, and discarded after each use. This, in principle, occurs all very transparently to the user.

A fuller description of Integral data analysis appears in section 2 of this manual and a walkthrough example is given in 4.6.

H.2 Current Exclusions

The v11 commands and features not provided in v12 are:

Feature	Rationale for exclusion
recornorm	With version 12.5.0, this has been replaced and improved upon by the recorn mixing model.
thleqw	Rarely used command not yet implemented.
extend	Beginning with version 12.3.0, this has been replaced by the more flexible energies command.
background models	This has been replaced by v12's multiple source modeling techniques

Additionally, we have withdrawn seldom-used fitting methods `anneal` and `genetic`. Future development will add new techniques.

Appendix I

Older Release Notes

I.1 v12.9.0 Jul 2015

New features

- Mix model components are no longer heavily restricted in their usage. Previously mix components could only be placed at the start of a model expression, to operate on the results of the rest of the expression. Now they can be inserted throughout the expression in the same manner as convolution components.
- Models with mix components can also be assigned to source numbers other than 1. A corollary of this is that a mix model is no longer required to operate on all currently loaded spectra.
- To allow more than one mix model at a time the syntax for XFLT#### keywords has been generalized. These keywords are now strings of form "key: value" with xspec maintaining an internal database of key, value pairs which mixing model code can access. Backward compatibility with the old version of XFLT#### containing only a value is maintained.
- Expanded the types of expressions allowed in parameter linking. It can now handle the same mathematical functionality supported in mdefine equations. Related to this, parameters can also be specified as **p#** and **modelName:p#** in link expressions, as well as the old **#** and **modelName:#**. The link expressions are now displayed using the **p#** format.
- The output of the **plot margin** command is now independent of the grid size. There is a new **plot integprob** command which also must be run

after the **margin** command and which plots the integrated probability within the contours shown.

- The data and back commands now handle files with multiple SPECTRUM extensions. There are a number of ways of specifying a particular extension: filenameN will use the Nth SPECTRUM extension; filenameSPECTRUM2 will look for an extension called SPECTRUM2; filenameback will use the first SPECTRUM extension with the HDUCLAS2 keyword set to "BKG"; filenamename=value will look for a SPECTRUM extension with keyword name set to value (in this case there must be no spaces within the). If the BACKFILE keyword in a source filename is set to the filename itself then XSPEC will get the background from the extension with HDUCLAS2 set to "BKG" (this, finally, makes Chandra level 3 files work seemlessly in XSPEC).

- The **error** command now has a `nonew` option. This determines whether to repeat the calculation when a new minimum is found. Previously it was relying on the **query** setting to make this decision.

- Added grey-scale image as background to to contour plots (to remove them use the `image off` command in PLT. **plot dem** now displays a histogram. The new **plot eqw** option shows the distribution of equivalent width values after running the **eqwidth** command with the `err` option.

- New model **nlapec**. There is also an APECNOLINES **xset** option which will turn off lines for all apec models.

- The models using AtomDB files now require the line files to be sorted in increasing order of energy within each element and ion. This allows a speed-up in the model calculation.

- Replaced the rescalecov script with the **chain rescale** command.

- The **plot delc** command now works for cstat, plotting (obs-mod)/error, however it is important to remember that this is not plotting contributions to the statistic.

Enhancements previously released as patches to 12.8.2

- A new -udmget64 option has been added to initpackage. This is intended for Mac platform users with local Fortran models which still make use of the udmget memory allocation function.

- **tclout steppar** can now retrieve values of the variable parameters as well has the fixed parameters which are being stepped through.

Bug fixes

All bug fixes to v12.8.2 released as patches are included in v12.9.0. In addition the following problems have been corrected:

- For the case where counting statistics are not applied, **fakeit** now outputs the spectrum in a RATE column rather than COUNTS. This is to remove bias caused by truncation errors. It also now outputs a STAT_ERR column and sets the POISSERR flag to false.
- Special characters have been removed from all model parameter names and (generally) replaced with underscores. This is to make their name strings more easily accessible in PyXspec.
- When calculating **tclout peakrsid** with Cstat, it should not make use of correction files.
- Redundant model fold calculations have been removed during certain update contexts.
- Fix to vequil model array assignment error.
- The default filenames for fakeit output should always be in the current run directory, even when the response file is in a different directory.
- A file-write race condition has been eliminated for the case of running **lmod** in parallel XSPEC sessions.
- the vvnei model had the tau and meankT parameters flipped.
- the AtomDB input files were being reopened unnecessarily often.

Changes to PyXSpec

New Features:

- Local models may now be written in Python and inserted into XSPEC's models library with new AllModels.addPyMod() function.
- The plot array retrieval interface (ie. Plot.x(), Plot.y()) has been expanded to allow retrieval from secondary plot panels in a mulit-panel plot.
- New Parameter.index attribute.
- New backscale attribute for Spectrum and Background classes.
- Added new function Fit.stepparResults() for retrieving results of most recent steppar run. (Previously available as a patch)

- New noWrite option added to AllData.fakeit (Previously available as patch)

Fixes:

- The Model.__call__ function now returns Parameter objects by reference rather than by value. This is to allow the returned object to retain any custom attributes the user may have added.
- Improved handling of Ctrl-c breaking in several prompting contexts.

I.2 V12.8.2 Jul 2014

New features

- The code underlying the NEI models has been rewritten to accommodate the new AtomDB 3.0 files, the beta version of which are included as the default when using any of the NEI models. Older versions can be recovered using **xset NEIVERS**. The CEI models (apec etc) use some of the same code but there should be no change in their behavior except for a slight speed-up.
- New models:
  - agauss – Gaussian model with units in Angstrom and a Gaussian shape in wavelength space.
  - nsx – Neutron star non-magnetic atmosphere model from Wynn Ho and collaborators.
  - nsmaxg – Neutron star magnetic atmosphere model.
  - rnei – NEI recombination model assuming an initial temperature.
  - vrnei – rnei with variable abundances for the common elements.
  - vvrnei – rnei with variable abundances for all elements (Z ¡= 30).
  - zagauss – The agauss model with a redshift parameter.
- The goodness command has added options fit—nofit. The fit option runs a fit on each simulation before calculating the test statistic.
- The cstat fit statistic can be modified by appending an integer eg. statistic cstat5. In this case the statistic will be evaluated as if the group min N had been used in grppha. This is not as efficient because the rebinning has to be done every statistic evaluation however it is useful for testing using simulations.

- The **model** command now checks that parameters are all within a reasonable order of magnitude ($\sim 10^{10}$) of each other, and issues a warning if they are not.
- Improved the handling of new-best-fit cases when running the **error** command on parallel processes. The global new-best-fit is now automatically retained. Previously the user was prompted to re-run **error** in a single process.
- Models can now store (keyword,value) pairs using loadDbValue (C++)or PDBVAL (Fortran). These values can be recovered using the **tclout** <`modkeyval`> option. This provides a way to get internal information from the last invocation of a model. There is a manual `XSFunctions_guide.pdf` in `Xspec/src/help` describing some of the C++ classes used in the XSFunctions library. This is intended to make it easier to use these classes in other models.

Enhancements previously released as patches to 12.8.1

- PyXspec: added Fit.testStatistic attribute for retrieving the test statistic value from the most recent fit.
- PyXspec: added compiler macros for switching to ¡Python/Python.h¿ include paths when building on Mac platforms.

Bug fixes

All bug fixes to v12.8.1 released as patches are included in v12.8.2. In addition the following problems have been corrected:

- Fix for a particular case of non-associative expression evaluation. Models of the form C((M1*A)M2) or C(M1*A)M2, where C is a convolution component, were being wrongly evaluated as (C(M1*A))M2.
- For users with external programs linking to the XSPEC model library, the FNINIT function now properly initializes the AtomDB version to v2.0.2.
- Fix for an **ignore** command syntax evaluation that was causing a crash on OS X Mavericks.

I.3 V12.8.1 Aug 2013

New features

- New models:
  - cpflux - a variant of cflux for photon flux.
  - heilin - Voigt absorption profiles for the HeI series
  - lyman - Voigt absorption profiles for the HI or HeII series
  - zbabs - EUV ISM attenuation
- A new statistic, pgstat, has been added for the case of Poisson-distributed data with a Gaussian-distributed background. The whittle statistic can now be used when fitting averaged power density functions by appending an integer (so eg whittle5 is the statistic to use when fitting a pdf constructed by averaging those from 5 observations).
- The old CERN Minuit library, which is used for the **migrad**, **minim**, **monte**, and **simplex** fitting methods and the **improve** command, has been replaced by the new version. The **minim** and **monte** methods are no longer supported and the new version does not include an **improve** command. The output from the **migrad** and **simplex** fitting methods now looks the same as that from the **leven** method. Note however that the rules for when to write intermediate fit results are not directly comparable so do not provide a measure of the relative speed of the methods.
- **fakeit** now has a <`nowrite`> option to generate fake spectra without producing output files. This is also now available in the `multifake.tcl` script command.
- Parallel processing capability has been added to the **steppar** command and can be invoked using the **parallel** command.
- Markov Chain Monte Carlo (the **chain** command) now uses the Goodman-Weare algorithm by default. Previously the default was Metropolis-Hastings.
- After a chain run, the best-fit parameters and statistic are now displayed with **chain info**, and are available through the **tclout chain** option.
- The default AtomDB version used in apec models may now be modified with the ATOMDB_VERSION keyword in the user's `Xspec.init` file.
- **steppar** now has a <`delta`> option for performing grids centered on the best-fit parameters.
- The **setplot delete** option has been enhanced to allow removal of all or a range of commands.
- For external programs calling XSPEC, new wrapper functions have been added for retrieving XFLT keywords from data files.
- Norm parameters are now set with a default "soft" upper limit below their "hard" upper limit.

- In PyXspec, the Fit.statMethod and statTest attributes can now be set for individual spectra.

Enhancements previously released as patches to 12.8.0

- AtomDB has been upgraded to version 2.0.2.
- The **tclout** <`stat`> and <`statmethod`> options can now retrieve the test statistic as well as the fit statistic.
- The **simftest** Tcl script command now takes an optional filename argument for output.
- Attributes added to PyXspec classes: Xset.parallel, Fit.statTest.

Bug fixes

All bug fixes to v12.8.0 released as patches are included in v12.8.1. In addition the following problems have been corrected:

- The command history file `xspec.hty` (in the user's ~`/.xspec` directory) is now updated when exiting XSPEC with the **quit** command. Previously it was only updated when exiting with **exit**.
- The **chain** command can now read/write files in ASCII format when running in the default Goodman-Weare mode. Previously this feature was only available for Metropolis-Hastings chains.
- Fix to an array access error in the nthcomp model.
- PyXspec fix removes error messages generated when accessing response parameters in Python versions 2.6.x.

I.4 v12.8.0 Dec 2012

New features

- Parallel-processing capability for specific tasks has been added by way of the new **parallel** command. This allows the user to set a maximum number of processes to spawn when running certain XSPEC commands. Currently the options for parallel-processing are limited to use with the

fit and **error** and commands, though we plan to implement more in the near future. This first implementation is considered to be a BETA release, and we look forward to hearing your comments and suggestions.

- XSPEC now distinguishes between the fit statistic and the test statistic. The fit statistic is used to find the best-fit parameter values and errors while the test statistic is used to provide a goodness-of-fit. Consequently, the **goodness** command now uses the test statistic. Separating these two classes of statistic has allowed us to add a number of new test statistics. These include Kolmogorov-Smirnov and the related Anderson-Darling and Cramer-von Mises. Also new are the runs statistic, based on the number of runs of consecutive positive (or negative) residuals, and the Pearson Chi-Square (pchi) statistic, Pearson's original test statistic.

- The basic Levenberg-Marquardt fit algorithm has undergone a number of changes. The most visible is an additional column in the output during the fit. —beta—/N is the norm of the vector of derivatives of the statistic with respect to the parameters divided by the number of parameters. At the best fit this should be zero so provides another measure of how well the fit is converging. —beta—/N can also be used as the criterion to stop the fit instead of the statistic delta although this is still considered experimental.

- Other internal changes to the fit algorithm are to treat the first iteration as a special case where only normalizations are allowed to change and to add the option of using delayed gratification, which can speed up convergence.

- New models: compmag, comptb, rgsxsrc. The latter is reinstated from it use in XSPEC v11.

- New plotting command **plot goodness**, for plotting a histogram of the most recent **goodness** simulation.

- New **tclout** option <`goodness sims`>.

- Added the option of using the Goodman-Weare algorithm instead of Metropolis-Hastings when using the chains command to run MCMC.

- Added the Whittle statistic for fitting models to power density spectra.

- If a data file is read which has RESPFILE="[]" then the response extensions (MATRIX, EBOUNDS) are read from the same file.

- Added support for a new type of model component: a mixing component which operates on the model pre-multiplied by the effective area. This is similar to the pile-up model type but allows for a models which require mixing between multiple spectra.

- Table model files can now have interpolated and additional parameters intermixed provided that additional parameters are indicated by METHOD=-1 in the PARAMETERS extension.

Enhancements previously released as patches to 12.7.1

- The "plot chain" has a new "thin" option for thinning the display of chain points.
- Added a "tclout ignore" option for easy retrieval of currently ignored channels. Similar capability added to PyXspec's Spectrum class.

Bug fixes

All bug fixes to v12.7.1 released as patches are included in v12.8.0. In addition the following problems have been corrected:

- The eqpair model used wrongly-sized arrays when multiple spectra were used with RMFs having varying numbers of photon energy bins. Eqpair (plus eqtherm and compth) also needed a new default value for its optical depth parameter, which fixes the problem if parameter 4 is zero.
- A fit parameter which started close to the minimum/maximum could under some circumstances be incorrectly pegged at the maximum/minimum.
- When the **error** command was run on a model with soft limits (ie. soft limits for some parameters set to a narrower range than hard limits), and the best fit value fell within the soft limit region, the displayed differences (in parentheses) were not consistent with the reported ± limits.
- Problem occurred in apec model when zero width lines were in the final energy bin.
- Fix to a potential normalization problem occurring in lorentz model with low-energy large-width lines.
- When running **initpackage** and **lmod** from PyXspec on certain Linux builds, local models calling functions in XSPEC's `xslib` library producing unresolved symbol link errors.

I.5 v12.7.1 March 2012

New features

- New models: gadem, vgadem, eplogpar, logpar, optxagn, optxagnf, and pexmon.
- The convolution models rdblur, rdblur2, kdblur and kerrconv have been sped up. They are now O(N) instead of O(N^2) where N is the number of energy bins in the response.
- Continued rationalization of Compton reflection routines. eqpair, eqtherm, compth, compps, ntee now all use the same routines as reflect and ireflct. For models with ionized reflector there may be changes in results because the new code uses the actual input spectrum to calculate the ionization fractions while the old code assumed a power-law.
- The parameter simulation arrays used for the calculation of **eqwidth** and **flux** error are now accessible through **tclout**.
- **initpackage** now works on Cygwin. The static_initpackage work-around is no longer needed and has been removed.
- New Fortran interface wrapper function RFLABD, for reading new abundance tables into external programs using the XSPEC models library.
- Added an xsetbl function for use in external programs. This provides access to XSPEC's internal exponential table model routines, similar to what xsatbl and xsmtbl do for additive and multiplicative table models.
- **bayes** command is now supported for response parameters (ie. gain).
- New **show version** option.
- Improved **error** command output messaging. The error results now have lower chatter level (5) than most of the warning messages (10), thus making it easier to filter out the warnings.
- PyXspec beta version upgraded to v1.0. See the PyXspec release notes for details.

Enhancements previously released as patches to 12.7.0

- New **tclout** options: <`nullhyp`>, <`rerror`>.

Bug fixes

All bug fixes to v12.7.0 released as patches a-u are included in v12.7.1. In addition the following problems have been corrected:

- Program aborted when attempting to fit with gain parameters attched to dummy responses.
- Program aborted when removing a spectrum with a response containing gain parameters, AND while that response was temporarily replaced with a dummy response.
- The **save** command did not add the default `.xcm` extension for file names that included a path.
- A crash could occur if the EBOUNDS array wasn't the right size.
- It was not possible to plot 2 or more models in a multi-panel **plot model** display.

I.6 v12.7.0 May 2011

New features

- The primary new feature of 12.7.0 is the addition of the Python module, PyXspec v0.9 beta. PyXspec is built and installed by default on most platforms along with the regular XSPEC build, and simply requires an **import xspec** statement to load into Python. XSPEC can now be run from object-oriented Python scripts, or interactively from a Python shell prompt. Detailed instructions can be found in `PyXspec.pdf`. While most features of standard XSPEC are already supported in this beta release, some still remain to be implemented. Please let us know if any missing feature is of particular importance to you, or if you have suggestions and ideas for improvement.
- New models:
  - cplinear: Piecewise linear non-physical background model for low-count spectra developed for Chandra by Patrick Broos.
  - eqpair, eqtherm, compth: Paolo Coppi's hybrid hot plasma emission models.
  - vvapec, bvvapec: APEC models allowing all 30 elemental abundances to vary (for use with AtomDB 2.0).
  - zigm: Multiplicative model, computes the mean attenuation of the optical/UV spectrum by the intergalactic medium.
  - zashift, zmshift: Convolution models for applying redshifts to additive and multiplicative models respectively.

- Also note that the default APEC model data files have been updated to AtomDB 2.0. This version of AtomDB includes contributions from more elements than earlier versions. When using the apec and vapec models these extra elements have Solar abundance by default. To change this use **xset** **APEC_TRACE_ABUND**.
- The **statistic** command may now be applied to individual spectra. This makes it possible to simultaneously fit spectra which require different fit statistics.
- Added the `pgstat` option to the **statistic** command. This is similar to using `cstat` with a background file except that the background is assumed to have Gaussian statistics (not Poisson) read from a STAT_ERR (and optionally SYS_ERR) column.
- Modified the pileup model for consistency with Sherpa and ISIS. A new parameter fracexpo is added which should be set to the FRACEXPO keyword value in the ARF.
- It is now possible to choose proportional or fixed fit deltas from the startup `Xspec.init` file. The initial default setting is now proportional deltas rather than fixed deltas.
- Improvements made to the output generated during a fit. The parameter names are listed at the top of columns, not just their numbers. Column alignment has been improved, and is no longer limited to a maximum width of 5 columns.
- Added reporting of the Bayesian contribution (if any) to the fit statistic output.

Enhancements previously released as patches to 12.6.0

- Initpackage now recognizes and builds files with .f03 extensions for Fortran 2003, and .f90 extensions for Fortran 90.
- Added a new **fakeit** option for setting the fake background exposure time.
- Added a new **tclout version** option for returning the XSPEC version string.
- Improved XSPEC's internal update mechanism to reduce the number of model calculations.

Bug fixes

All bug fixes to v12.6.0 released as patches a - ab are included in v12.7.0. In addition the following problems have been corrected:

- A crash occurred if **tclout notice energies** was performed on a spectrum containing only a dummy response with no channels.
- The 5 redundant **xset** options (those which merely duplicate other existing XSPEC commands) weren't passing their arguments to the command handlers correctly.
- Fit error messages were misleading for the case where the data was missing a suitable response, or when the only existing models were inactive.
- A fatal error could occur in **fakeit** when attempting to generate a background file while only a dummy was used for the response.
- The **bayes** command handler was not properly handling the case where the prior type option string was abbreviated.
- A crash occurred when **flux** was run in error mode and the specified energy was entirely outside the range of one or more spectra.

I.7 v12.6.0 March 2010

New features

- The main improvements in version 12.6.0 are to XSPEC's plotting capabilities:
  - Multi-panel plotting is now supported for all combinations except contour plots. For example, **plot data model resid ratio** will produce a 4-panel plot on a single page. Up to 6 panels can be plotted in this manner.
  - There are many choices for axis units. These can be selected using the **setplot energy** and **setplot wave** commands. For example, **setplot energy GeV** uses GeV on the x- and y-axes. **setplot wave** also has a new `perhz` option for displaying the Y-axis in 1/Hz units.
  - The **setplot** command has a new `redshift <z>` option for shifting displayed energies to the source frame.
- New models: ireflect is a convolution model based on the pexriv code. sirf is a multiblackbody self-irradiated funnel
- The normalizations on all power-law models (ie. powerlaw, bknpow, bkn2pow, cutoffpl)can be changed to a flux over an energy range by setting POW_EMIN and POW_EMAX keywords in **xset**. The powerlaw model then becomes equivalent to the pegpwlw model.

- The Compton reflection models (b/p)exr(a/i)v and (i)reflect have been restructured to use adaptive Gauss-Kronrod quadrature for the Greens' function integrals. The precision to which the integrals are calculated can be set allowing a trade-off between speed and precision.
- The wrapper functions additiveTable and multiplicativeTable give external C++ models access to XSPEC's table model interpolation routines (equivalent to the xsatbl and xsmtbl functions for Fortran models).
- The display of link expressions has been simplified to show only the parameter numbers and not the extraneous component information. Also **show model** will now only display the model components and not the individual parameters. The parameters can be seen with **show parameters**.

Additional enhancements previously released as patches to 12.5.1

- Added the solar abundance data set of Asplund, Grevesse, and Sauval (2006) to the list of available tables accessed with the **abund** command.
- New **tclout nchan** option for returning the number of channels in a spectrum. The **save** command now stores relative rather than absolute paths to allow easier porting to other machines.
- The recorn model component has been converted from a mixing to a multiplicative type. This allows a model to define multiple recorn components.
- A warning message is now issued if a user attempts to load a response for a source n when there are still slots to fill for source n-1. This is intended to catch cases where a user mistakenly reverses the source and spectrum number input to the **response** command.

Bug fixes

All bug fixes to v12.5.1 released as patches a - o are included in v12.6.0. In addition the following problems have been corrected:

- It was possible for the addition of a systematic model error to actually decrease the overall variance, when it was applied to a zero-variance bin that was artificially increased by XSPEC for chi-square fitting.
- Bug in **plot ratio** when using **setplot wave** with Hz units. Y-axis model values $\langle 10^{-20}$ were not displayed in plot.
- The comptt model no longer stops and prompts the user when it fails during its incomplete gamma calculation.

- The powerlaw model has been modified to avoid a numerical instability that could occur if the index were within 10^{-12} to 10^{-15} of 1.0.

I.8 v12.5.1 Aug. 2009

New features

- Gain parameters can now be used in the **error**, **freeze**, **newpar**, **thaw**, and **untie** commands by prefixing the command name with the letter "r" (for "response parameter", the more general category to which gain parameters belong). **steppar** can now also handle gain parameters. Gain parameters can be displayed either with **show parameters** or the new **show rparameters** option.
- The **gain** command syntax has changed when using multiple sources. To better conform with the rest of XSPEC, it now requires <`source number`>:<`spectrum number`> rather than the reverse.
- Gain parameter limit values can be stored in response files, using the keywords GSLOP_MIN, GSLOP_MAX, GOFFS_MIN, and GOFFS_MAX.
- All input and output data filenames can now include CFITSIO/FTOOLS extended-syntax for specifying particular HDUs. As a result, XSPEC can now handle files which contain spectra, ARFs and RMFs in multiple extensions.
- Partial derivative calculations during fitting can now be performed numerically rather than with an approximated analytical expression. This option is chosen in the `Xspec.init` initialization file.
- If a new minimum is found during a **steppar** run, **steppar** now prompts the user for acceptance of the new values. Also the delta statistic column of a **steppar** run is now obtainable with the **tclout steppar delstat** option.
- The output warning message has been improved in the case where Levenberg-Marquardt fitting runs into a zero diagonal element in the second derivative matrix. Similarly, the more frequent pegged-parameter messages (due to running into hard limits) is now output at higher chatter levels only.
- All calls to the xanlib dynamic memory allocation function udmget have been removed from the Fortran models in XSPEC's models library. The relevant code has been converted to C++. If a user's local models library still requires the udmget code, they'll need to run **initpackage** with the new -udmget option.

Additional enhancements previously released as patches to 12.5.0

- Setplot wave x-axis units can be toggled from Hz to angstroms through WAVE_PLOT_UNITS entry in `Xspec.init` file.
- New **tclout gain** and **tclout sigma** options.
- New xs_getVersion function available for those linking their own programs to the XSPEC models library.
- The **show parameters** option can now take a range of parameters for displaying subsets.

Bug fixes

All bug fixes to v12.5.0 released as patches a - an are included in v12.5.1. In addition the following problems have been corrected.

- After running the **arf** command, any gain previously applied to the associated RMF will be removed. Previously it was erroneously applying the gain to the new ARF.
- Additional header file inclusions needed in code files to compile with g++-4.4.0
- Extra line-feed characters removed from Ascii text files in the `modelData` directory. These were causing problems on Solaris 10 w/f90.
- The nthcomp model's internal arrays were hardcoded to a maximum size of 5000 energy bins. The size is now dynamically allocated. (This also affects the diskir model.)
- A Levenberg-Marquardt fit now immediately stops if the fit statistic becomes NaN due to an erroneous model calculation.
- C++-style comments have been removed from `xsFortran.h` for the benefit of users compiling their own C programs with the models library.
- Plotting fix for case where **setplot area** is selected and no models are currently loaded.
- Model parsing fix for case of nested parentheses with no '+' operator, ie. A(B(C*D)).