Duo: a general program for the calculation of spectra of diatomic molecules[*]

User's manual

Yurchenko, Sergey N.
`s.yurchenko@ucl.ac.uk`

Lodi, Lorenzo
`l.lodi@ucl.ac.uk`

Tennyson, Jonathan
`j.tennyson@ucl.ac.uk`

Stolyarov, Andrey V.
`avstol@phys.chem.msu.ru`

October 31, 2018

1 Introduction

Duo is a computer program for the spectroscopy of diatomic molecules. Its main functionalities belong to one of these three tasks:

1. Given a set of potential energy curves (PECs) Duo can solve the corresponding one-dimensional Schrödinger equation, which for $^1\Sigma$ states is

$$-\frac{\hbar^2}{2\mu}\frac{\mathrm{d}^2}{\mathrm{d}r^2}\psi_{\upsilon J}(r) + \left[V_{\rm state}(r) + \frac{J(J+1)}{2\mu r^2}\right]\psi_{\upsilon J}(r) = E_{\upsilon J}\psi_{\upsilon J}(r) \ , \qquad (1)$$

   and find the bound-state energies and wave functions. PECs may be coupled to one another by a variety of coupling terms, in which case the relevant coupling curves should be also provided. Supported couplings include spin-orbit, spin-electronic, spin-rotational, L-uncoupling and S-uncoupling.

2. Given a set of PECs, coupling curves and dipole moment curves Duo can compute line intensities for rotational, vibrational and electronic transitions.

3. Given a set of reference energy levels or line positions (e.g., obtained from experiment) Duo can find the PECs and coupling curves which best reproduce the given data (empirical refinement of PECs or 'fitting').

[*]See Ref. [YLTS15]

DUO inputs can be broken down into three sections:

1. Calculation setup.
2. Specification of the Hamiltonian (PECs and couplings).
3. Calculation of spectra *or* fitting of PECs and couplings (optional).

This manual is organized as follows. In Section 2 we introduce the main functionalities of DUO and illustrate how to compute energy levels, which is a prerequisite step to computing spectra or fitting. In Section 3 we discuss the calculation of spectra and in Section 4 fitting. Section 5 contains a list of the function forms implemented in DUO to represent curves. Finally, Section 7 contains technical information on how to compile and run the program under Linux, Windows and OS X.

1.1 Getting started

DUO runs from the command line (see Section 7) and uses as input a plain text input file; DUO is run with a command of the type

```
./duo.exe < input.inp > output.txt
```

The input file is organized in self-contained input lines (e.g., `masses 1.00000 1.00000` specifies the masses of the two atoms in Daltons) or in input sections beginning with a specific keyword (e.g., `grid`) and ending with the keyword `end`. The input is not case sensitive, so `masses`, `MASSES`, `Masses` or any other combinations of uppercase and lowercase letters work in exactly the same way. A comma, a space or a hyphen (minus sign) can all be used as delimiters, so, e.g., one can also write `masses 1.00000, 1.00000`. Sometimes keywords have several aliases, which are all equivalent. Lines delimited by parentheses (i.e., round brackets) are ignored and can be used for comments. If in the input there is a line with one of the keyword `END`, `STOP` or `FINISH` all lines after it are ignored.

2 Computing energy levels

In the following we present all keywords and options relevant to the calculations of energy levels; a commented input is reported starting from Section 2.8.

2.1 Calculation setup

In the calculation setup we specify global information about the molecule.

- `atoms` defines the chemical symbols of the two atoms. For example:

```
atoms Na-23 H-2
```

specifies the ^{23}NaD diatomic. DUO includes an extensive database of atomic properties (atomic masses, nuclear spins, isotopic abundances and other quantities) and will use the appropriate values when required. The database should cover all naturally-occurring nuclei as well as all radioactive ones with a half-life greater than one day and is based on the AME2012 and NUBASE2012 databases [WAW$^+$12, AKW$^+$12]. Each atom should be specified by its chemical symbol, a hyphen (minus sign) and the atomic mass number, like in the example above. Atomic masses will be used, which is generally the most appropriate choice unless one is explicitely including non-adiabatic corrections. The hydrogen isotopes deuterium and tritium can also be optionally specified by the symbols `D` and `T`.

The atomic mass number can be omitted, like in the following example:

```
atoms Li F
```

In this case DUO will use the most-abundant isotopes (^{7}Li and ^{19}F in the example above) or, for radioactive nuclei not naturally found, the longest lived one. For example

```
atoms Tc H
```

selects for technetium the isotope ^{97}Tc, which is the longest-lived one. A few nuclides in the database are nuclear metastable isomers, i.e. long-lived excited states of nuclei; these can be specified with a notation of the kind

```
atoms Sb-120m H
```

In the example above the radioactive isotope of antimony ^{120m}Sb is specified (and hydrogen). Another example

```
atoms Sc-44m3 H
```

specifies the scandium radioactive isotope ^{44m3}Sc (and hydrogen).

- `masses` is an optional keyword which specifies explicitely the masses of the two atoms (in Daltons, i.e. unified atomic mass units), overriding the values from the internal database if the keyword `atoms` is also specified. For example, the masses for the CaO molecule would be:

```
masses 39.9625906 15.99491463
```

The masses may be atomic masses (the recommended choice if one does not include adiabatic or non-adiabatic corrections), nuclear masses.[1] An up-to-date reference of atomic masses is provided by the AME2012 catalogue [WAW$^+$12] Duo can also make use of position-dependent masses (which is a practical way to account for non-adiabatic effects), which are described in Section 2.5.

- `nstates` is the number of potential energy curves (PECs) included in the calculation. For example, if the ground state and four excited states of a molecule are to be included:

  ```
  nstates 5
  ```

  Note that if `nstates` is set to a number different from the actual number of PECs included in the input file no error message is issued; if more than `nstates` PECs are included in the input file then the PECs with `state` > `nstates` will be ignored.

  Note also that, consistently with the way Duo works internally, `nstates` is the number of unique PECs in absence of spin-orbit couplings.

- `jrot` specifies the set of total angular momentum quantum numbers to be computed. These must be integers or half-integers, depending on whether there is an even or odd number of electrons. One can directly specify the values (separated by spaces or commas), specify a range of values (a minimum and a maximum values separated by a hyphen; note than the hyphen must be surrounded by at least by one space on each side). The values do not have to appear in ascending order. For example, the following line

  ```
  jrot 2.5, 0.5, 10.5 - 12.5,  20.5
  ```

  specifies the set J =0.5, 2.5, 10.5, 11.5, 12.5, 20.5.

  The first J in the `jrot` list will be used to define the reference zero-point-energy (ZPE) value for the run.

  Note that in the optional sections specifying calculation of spectra (see Section 3) or specifying fitting (section 4) is necessary to specify again a list of J values by `J` and `Jlist` respectively, which are completely independent from the `jrot` value specified for energy level calculation.

[1]Actual, physical nuclear masses are obtained [WAW$^+$12] by $m_{\text{nuclear}} = m_{\text{atomic}} - Zm_e + B_e(Z)/c^2$, i.e. by subtracting the mass of the electrons and adding the mass-equivalent of the binding energy of the electrons; a fit of the Dirac-Hartree-Fock atomic energies Ref.[VD97] gives, within a fitting error of ±5 %, $B_e(Z)/c^2 \approx (Z/1729)^{2.42}$ Da. In practice B_e is a tiny correction and can be neglected, which is also arguably a more consistent choice in a non-relativistic context.

- `symmetry` (or `Symgroup`) is an optional keywork which specifies the molecular permutation-inversion symmetry group, which is `Cs(M)` for heteronuclear diatomics and `C2v(M)` for homonuclear diatomics. For example:

  ```
  symmetry Cs(M)
  ```

  Instead of `C2v(M)` one can write equivalently `C2h(M)` or `G4(M)`, as these groups are isomorphic; the only difference will be in the labels used for the energy levels. The short-hand notations `Cs`, `C2v`, `C2h` and `G4` can also be used and are equivalent to the ones with `(M)`. The energy calculations are done using C_s(M), which is also the default, while for the intensities the `C2v(M)` group can be also used. Note that this keyword refers to the symmetry of the exact total (electronic, vibrational and rotational) Hamiltonian and *not* to the $C_{\infty v}$ or the $D_{\infty h}$ point groups, which are relative to the clamped-nuclei electronic Hamiltonian.

- `DO_NOT_SHIFT_PECS` suppresses shifting the PEC to the minimum of potential 1 (assumed to the lowest). The default is to do the shift of the PECs to the minimum of `poten 1`. In order to suppress shifting energies to ZPE, use

  ```
  ZPE 0.0
  ```

  see also the description of the keyword `ZPE`.

- `SOLUTIONMETHOD` defines the DVR basis set and thus the DVR solution method for the vibrational problem. Possible methods include `5POINTDIFFERENCES`

  ```
  SOLUTIONMETHOD  5POINTDIFFERENCES
  ```

  for the 5 points stencil finite differences to derive the kinetic energy operator. A more efficient method is Sinc DVR (default), which is switched on with

  ```
  SOLUTIONMETHOD  SINC
  ```

  Since Sinc is also currently the default method, this does not have to be specified.

2.2 Defining the grid

`grid` specifies an input section with the specifications of the grid of points used for the solution of the vibrational problem. Example:

```
grid
  npoints 501
  unit angstroms
  range 1.48 , 2.65
```

```
   type 4
   alpha  1.0
   re 1.80
end
```

Keywords:

- `npoints` is the number of grid points N_p. Typical runs use 100 to 500 points.
- `units` is optional and specifies the unit of measure of the grid specifications; possible values are `angstroms` (default) or `bohrs`.
- `range` specifies the range of the grid in terms of $r_{\rm min}$ and $r_{\rm max}$, the lower and upper bond lengths. $r_{\rm min}$ should be strictly greater than zero and $r_{\rm max}$ strictly greater than $r_{\rm min}$. As elsewhere in the program, the value may be separated by a space or a comma.
- `type` is an integer number ≥ 0 which specifies the type of grid. Duo support not only uniformely spaced grids (default), which correspond to `type 0`, but also various kind on non-uniformely spaced ones, which are particularly useful for near-dissociation, very weakly bound states [MSLR08]. Example:

  ```
  type 0
  ```

  In the case of uniformely-spaced grids the mesh points r_j, $j = 0, N_p - 1$ are given by

  $$r_j = r_{\rm min} + \Delta j \qquad \text{where} \qquad \Delta = \frac{r_{\rm max} - r_{\rm min}}{N_p - 1} \tag{2}$$

  Non-uniformely spaced grids are based on a change of variables from r to $z = f(r)$; it is then the transformed variable z that is uniformely sampled. The transformed variables z are parametrised by two parameters, r_e and α, which have to be specified for the grid types > 0 (see below).

  Transformed variable currently implemented are [MSLR08]:
  `type 1` $z = \exp(-e^{-\alpha(r-r_e)})$
  `type 2` $z = 1 - \left(1 + e^{\alpha(r-r_e)}\right)^{-1}$
  `type 3` $z = \arctan(\alpha(r - r_e))$
  `type 4` $z = (y-1)/(y+1)$ with $y = (r/r_e)^\alpha$
  All the transformed grids have the property of decreasing the density of points for large r, so that one does not 'waste' too many points in regions where the potential is almost constant and the corresponding vibrational wave function slowly varying, see figure 1 for an example.

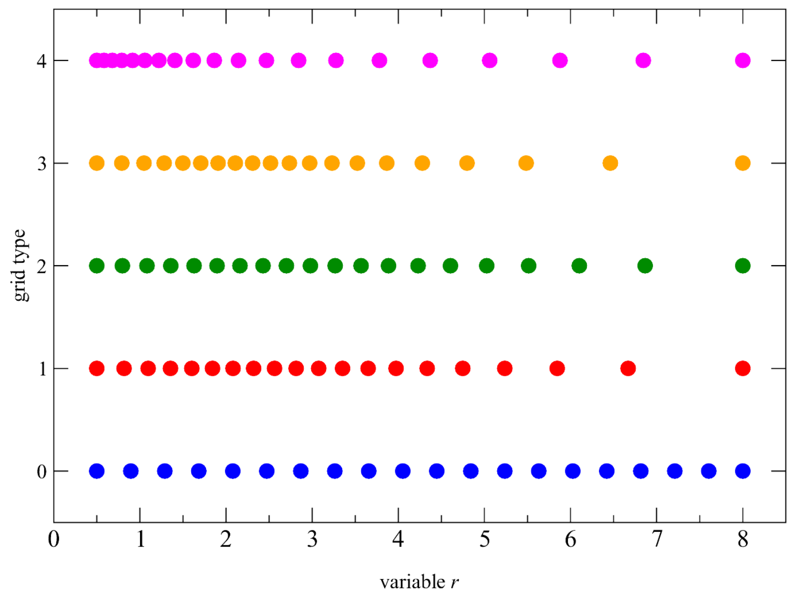


Figure 1: Example of the non-uniformly spaced grids implemented in Duo, see Section 2.2. `type 1` corresponds to a transformation to the variable $z = \exp(-e^{-\alpha(r-r_e)})$; `type 2` to $z = 1 - \left(1 + e^{\alpha(r-r_e)}\right)^{-1}$; `type 3` to $z = \arctan(\alpha(r - r_e))$; `type 4` to $z = (y - 1)/(y + 1)$ with $y = (r/r_e)^\alpha$. All grids contain 20 points and span from $r = 0.5$ to $r = 8.0$. We set the parameters to $r_e = 2.0$ and $\alpha = 0.5$ in all cases.

- `re` (alias: `ref`) Reference bond length used for `type` > 0 (see above).
- `alpha` Parameter α for `type` > 0 (see above).

2.3 Vibrational basis set

The keyword `vibrationalbasis` (aliases: `vibrations`, `contraction`) specifies the size of the vibrational basis set. As explained in Ref. [YLTS15] Duo uses a 'contraction' scheme to construct the rovibronic basis set used for the solution of the coupled problem. As a first step the $J = 0$ vibration problem is solved for each electronic state, in which the corresponding Schrödinger equation (Eq. (1)) is solved in the grid representation of `npoints`. Then a certain number of the resulted vibrational eigenfunctions $|v\rangle$ with $0 \leq v \leq$`vmax` and $\tilde{E} \leq$ `EnerMax` is selected to form the vibrational part of the basis set

$$|J\Omega S\Sigma\Lambda v\rangle = |J\Omega\rangle|S\Sigma\rangle|\Lambda\rangle|v\rangle$$

where $|J\Omega\rangle$ and $|S\Sigma\rangle$ are the rigid rotor functions and $|\Lambda\rangle$ are the electronic wavefunctions implicitly taken from the *ab initio* calculations.

Example:

```
vibrationalbasis
 vmax 30
 enermax 25000
end
```

2.3.1 Keywords

- `vmax` (alias: `vibmax`) specifies the value of the maximum vibrational quantum number to be computed and kept for the solution of the coupled problem. For example

  ```
  vmax 15
  ```

  specifies to compute for each PEC the lowest-energy 15 vibrational levels; it is also possibile to specify different values of `vmax` for each PEC, in which case the values must be given as a list; for example

  ```
  vmax 10 15 8
  ```

  specifies that for the PEC identified as `poten 1` (see Section 2.5) Duo should use $v_{\max} = 10$, for `poten 2` $v_{\max} = 15$ and for `poten 3` $v_{\max} = 8$. If there are more PEC (`poten 4` etc.) they will use for `vmax` the last value specified ($v_{\max} = 8$ in this example).

- `enermax` Alternatively or complementary to `vmax` one can select the vibrational energy levels to compute by specifying an upper energy threshold (in cm^{-1}). Similarly to `vmax`, one can specify a different value of `enermax` for each PEC by writing a list of values; for example

  ```
  enermax 30000.0 25000.0
  ```

  selects a threshold of 30 000 cm^{-1} for `poten 1` and one of 25 000 cm^{-1} for `poten 2` and any other potential present. Note that by default Duo will shift the PECs so that the lowest point of the lowest-lying PEC has zero energy, and that the energy used for the `enermax` threshold are 'total' vibrational energies including the zero point energy. One can prevent Duo from shifting the PECs by writing in the input (anywhere but not within an input section) the option `do_not_shift_pecs`.

  If both `enermax` and `vmax` are specified only levels which satisfy both criteria are kept for the solution of the coupled problem. If neither of them is specified (or the `vibrationalbasis` input section is missing altogether) then `vmax` is taken equal to `npoints` for all PECs and there is a hard-coded limit of 10^8 cm^{-1} for `enermax`.

2.4 Options for the coupled problem

The input section `EigenSolver` (aliases: `FinalStates`, `diagonalizer`, `FinalStates`) specifies various options relative to the $J > 0$ and/or the coupled problem; it also specifies the LAPACK routine which should be used for matrix diagonalization (both for the solution of the vibrational problem and for the solution of the coupled problem). Example:

```
Eigensolver
  enermax 25000.0
  nroots 500
  ZPE 1200.0
  SYEVR
END
```

2.4.1 Keywords

- `nroots` is the number of energy levels of the coupled problem to be computed (for any of the specified values of `jrot`). Example:

  ```
  nroots 500
  ```

- `enermax` (aliases: `uplimit`, `enercut`) is an energy threshold (cm^{-1}) selecting the energy levels of the coupled problem to be computed. For example:

  ```
  enermax 15000.
  ```

  If both `nroots` and `enermax` are specified then only levels satisfying both criteria are selected. Note that the present `enermax` threshold is distinct from the homonymous one in the `vibrationalbasis` input section, as the latter refers to the solution of the $J = 0$ uncoupled problem while the one being discussed at present refers to the solution of the full (rotationally excited and/or coupled) problem.

- `ZPE` allows to explicitly input the zero-point energy (ZPE) of the molecule (in cm^{-1}). This affects the value printed, as Duo always prints energy of rovibronic levels by subtracting the ZPE. Example:

  ```
  ZPE 931.418890
  ```

  If `ZPE` is not included Duo will define the ZPE value as the lowest computed energy for the first value of J listed next to the `jrot` keyword (`jlist` in the fitting), *from the positive parity block*. Currently it is not possible to take an automatic ZPE from the negative parity block (it is however possible in the intensity and fitting parts of the output). Thus ZPE does not necessarily have to be from the ground electronic state. This ZPE taken from the `eigensolver/diagonalizer` section changes the energies in the main, standard Duo output.

The ZPE shift can be suppressed by setting the ZPE value to zero. This should be done either in the `Diagonalizer`, `Fitting` or `Intensity` sections, depending on the current task:

```
ZPE 0.0
```

- `SYEVR` or `SYEV` This optional keywords permits to specify which routine from the LAPACK library should be used for matrix diagonalization. At the moment only the two options quoted are implemented. Example:

  ```
  SYEVR
  ```

  The SYEV routine (default) first reduces the matrix to diagonalize to tridiagonal form using orthogonal similarity transformations, and then the QR algorithm is applied to the tridiagonal matrix to compute the eigenvalues and the eigenvectors. The SYEVR routine also reduces the matrix to diagonalize to tridiagonal form using orthogonal similarity transformations but then, whenever possible, computes the eigenspectrum using Multiple Relatively Robust Representations (MR). SYEVR might give better performance, although exact timings are system- and case-dependent. See also the comments in Ref. [YLTS15]

2.5 Specification of curves and couplings

Once the main global parameters have been specified as described in the previous sections, it is necessary to introduce the PECs and the various coupling curves defining the Hamiltonian. Dipole moment curves (DMCs), which are necessary for calculating spectral line intensities, are also discussed in this section, as well as some special objects which are used for fitting. Each object specification consists in a first part in which keywords are given and a second one (starting from the `values` keyword) in which numerical values are given; the order of the keywords is not important, except for `values`. Each object specification is terminated by the `end` keyword.

Objects of type `poten` (i.e., PECs, discussed in more detail below) begin with a line of the kind '`poten` N' where N is an integer index number counting over potentials and identifying them. It is recommended that PECs are numbered progressively as $1, 2, 3, \ldots$, although this only restriction is that the total number $N_{\max}$ of PECs should be not less than the total number of states specified by the keywork `nstates` (section 2.1).

Most other objects (e.g., `spin-orbit`) are assumed to be matrix elements of some operator between electronic wave functions and after the keyword identifying their type require two integer numbers specifying the two indexes of the two electronic states involved (bra and ket). The indexes are the numbers specified after the `poten` keyword.

Currently Duo supports the following types of objects:

- `poten` (alias: `potential`) Objects of type `poten` represent potential energy curves (PECs) and are the most fundamental objects underlying each calculation. From the point of view of theory each PEC is the solution of the electronic Schrödinger equation with clamped nuclei, possibly complemented with the scalar-relativistic correction and with the Born-Oppenheimer Diagonal correction (also known as adiabatic correction). Approximate PECs can be obtained with well-known quantum chemistry methods such as Hartree-Fock, coupled cluster theory etc. Objects of type `poten` should always appear before all other objects as they are used to assign to each electronic states its quantum numbers. Here is an example for a PEC showing the general structure:

```
poten 1
name "a 3Piu"
symmetry u
type  EMO
lambda 1
mult   3
values
V0          0.82956283449835E+03
RE          0.13544137530870E+01
DE          0.50061051451709E+05
RREF       -0.10000000000000E+01
PL          0.40000000000000E+01
PR          0.40000000000000E+01
NL          0.20000000000000E+01
NR          0.20000000000000E+01
B0          0.20320375686486E+01
B1         -0.92543284427290E-02
B2          0.00000000000000E+00
end
```

- `L2` (alias: `L**2`) These objects represent matrix elements between electronic states of the molecule-fixed angular momentum operator $\hat{L}^2 = \hat{L}_x^2 + \hat{L}_y^2 + \hat{L}_z^2$. See Section 2 of Ref. [YLTS15] for more information.

- `L+` (aliases: `Lplus`, `LxLy`) and `Lx` represent matrix elements between electronic states of the molecule-fixed angular momentum operator $\hat{L}_+ = \hat{L}_x + i\hat{L}_y$ and $\hat{L}_x$ in the Λ- and Cartesian-representations, respectively. See Section 2 of Ref. [YLTS15] for more information.

- `spin-orbit` and `spin-orbit-x` These objects are matrix elements of the

Breit-Pauli spin-orbit Hamiltonian in the Λ- and Cartesian-representations, respectively (see Section 2.2 of Ref. [YLTS15]).

Example:

```
spin-orbit  1 3
name "<0,S=0 (X1Sigma+)|LSY|+1 (a3Pi),S=1> SO1"
spin   0.0 1.0
lambda 0 -1
sigma 0.0 -1.0
type   grid
factor sqrt(2)  (1 or i)
units bohr  cm-1
values
   2.80     17.500000
   2.90     15.159900
   3.00     12.347700
   3.10      9.050780
   3.20      5.391190
   3.30      1.256660
   3.40     -3.304040
   3.50     -8.104950
   3.60    -12.848400
   3.70    -17.229100
   3.80    -21.049000
   3.90    -24.250400
   4.00    -26.876900
   4.10    -29.014700
   4.20    -30.756100
   4.30    -32.181900
   4.50    -34.335500
   5.00    -37.348300
end
```

For the `spin-orbit-x` case (Λ-representation), the value of the matrix elements of the $\hat{L}_z$ operator nust be specified using the `<x|Lz|y>` keyword. This representation is designed to work with e.g., the MOLPRO outputs. For $\Lambda \neq 0$, the diagonal SO-matrix element (e.g. between to Π-components of $\Lambda = 1$) should be specified using the $\langle \Pi_x | LSZ | \Pi_y \rangle$ component (e.g. ⟨1.2|LSZ|1.3⟩).

- `spin-spin-p` and `spin-spin-o` Parametrised phenomenological spin-spin operator (diagonal and off-diagonal. See Section 2.2 of Ref. [YLTS15]).
- `spin-rot` Matrix elements of the spin-rotational operator (see Section 2.2 of Ref. [YLTS15]).
- `bob-rot` (alias: `bobrot`) Specifies the rotational g factor (rotational Born-Oppenheimer breakdown term), which can be interpreted as a position-dependent modification to the rotational mass (see Section 2.2 of Ref. [YLTS15]).
- `diabatic` (alias: `diabat`) Non-diagonal coupling of potential energy functions in the diabatic representation.

- `lambda-opq`, `lambda-p2q`, and `lambda-q` These objects are three Lambda-doubling objects which correspond to $o^{\rm LD} + p^{\rm LD} + q^{\rm LD}$, $p^{\rm LD} + 2q^{\rm LD}$, and $q^{\rm LD}$ couplings, see Ref. [YLTS15].

  Example:

```
lambda-p2q  1 1
name "<X,2Pi|lambda-p2q|X,2Pi>"
lambda     1 1
spin   0.5 0.5
type  BOBLEROY
factor    1.0
N 8
values
RE           0.16200000000000E+01
RREF        -0.10000000000000E+01
P            0.10000000000000E+01
NT           0.20000000000000E+01
B0           0.98500969657331E-01
B1           0.00000000000000E+00
B2           0.00000000000000E+00
BINF         0.00000000000000E+00
end
```

- `abinitio` Objects of type `abinitio` (aliases: `reference`, `anchor`) are reference, ‘abinitio’ curves which may be specified during fitting. When they are used they constrain the fit so that the fitted function differs as little as possible from the *ab initio* (reference) one (see Ref. [YLTS15]). The reference curve is typically obtained by *ab initio* methods. For any Duo object one can specify a corresponding reference curve as in the following example:

```
abinitio spin-orbit 1 2
name "<3.1,S=0,0 (B1pSigma)|LSX|+1 (d3Pig),S=1,1>"
spin   0.0 1.0
type   grid
units bohr cm-1
values
     2.3       -3.207178925    13.0
     2.4       -3.668814404    24.0
     2.5       -4.010985122    35.0
     2.6       -4.271163495    46.0
     2.7       -4.445721312    47.0
     2.8       -4.468083270    48.0
end
```

- `dipole` (aliases: `dipole-moment`, `TM`) and `dipole-x` Diagonal or transition dipole moment curves (DMCs), necessary for computing (dipole-allowed) transition line intensities and related quantities (Einstein A coefficients etc.). `dipole-x` is related to the Cartesian-representation.

  At the moment Duo cannot compute electric-quadrupole or magnetic dipole transition line intensities.

In the following we give information on the keywords used during the object characterisation.

2.6 Keywords used in the specification of objects

This is a list of keywords used to specify various parameters of Duo objects.

- `name` is a text label which can be assigned to any object for reference in the output. The string must appear within quotation marks. Examples:

  ```
  name "X 1Sigma+"
  name "<X1Sigma\|HSO\|A3Pi>"
  ```

- `lambda` specifies the quantum number(s) Λ, i.e. projections of the electronic angular momentum onto the molecular axis, either for one (PECs) or two states (couplings). It must be an integral number and is allowed to be either positive or negative. Examples:

  ```
  lambda 1
  lambda 0 -1
  ```

  The last example is relative to a coupling-type object and the two numbers refer to the bra and ket states.

- `sigma` specifies the quantum number(s) Σ, i.e. the projections of the total spin onto the molecular axis, either for one (diagonal) or two states (couplings). These values should be real ($-S \leq \Sigma \leq S$) and can be half-integral, where S is the total spin. `sigma` is currently required for the spin-orbit couplings only. Example:

  ```
  sigma 0.5 1.5
  ```

  where two numbers refer to the bra and ket states.

- `mult` (alias: `multiplicity`) specifies the multiplicity of the electronic state(s), given by $(2S + 1)$, where S is the total spin. It must be an integer number and is an alternative to the `spin` keyword. Examples:

  ```
  mult 3
  mult 1 3
  ```

  The last example is relative to a coupling-type object and the two numbers refer to the bra and ket states.

- `spin` The total spin of the electronic state(s), an integer or half-integer number. Example:

  ```
  spin 1.0
  spin 0.5 1.5
  ```

The last example is relative to a coupling-type object and the two numbers refer to the bra and ket states.

- `symmetry` This keyword tells Duo if the electronic state has gerade `g` or ungerade `u` symmetry (only for homonuclear diatomics) and whether it has positive (`+`) or negative `-` parity (only for Σ states, i.e. states with $\Lambda = 0$, for which it is mandatory). Examples:

```
symmetry +
symmetry + u
symmetry g
```

  The `g/u` or `+/-` can appear in any order.

- `type` selects the parametrised analytical function used for representing the objects or selects the interpolation type to be used. The function types supported by Duo are listed in Section 5. Examples:

```
type grid
type polynomial
type morse
```

  In the examples above `grid` selects numerical interpolation of values given on a grid, `polynomial` selects a polynomial expansion and `morse` selects a polynomial expansion in the Morse variable. See Section 5 for details.

- `Interpolationtype` is used only for `type grid` and specifies the method used for the numerical interpolation of the numerical values. The currently implemented interpolation methods are `Cubicsplines` and `Quinticsplines` (default). Example:

```
Interpolationtype Cubicsplines
Interpolationtype Quinticsplines
```

- `factor` This optional keyword permits to rescale any object by an arbitrary multiplication factor. At the moment the accepted values are any real number, the imaginary unit i, the square root of two, written as `sqrt(2)`, or products of these quantities. To write a product simply leave a space between the factors, but do not use the `*` sign. All factor can have a $\pm$ sign. The default value for `factor` is 1. This keyword is useful, for example, to temporarily zero a certain object without removing it from the input file. Examples:

```
factor 1.5
factor -sqrt(2)
factor -2 sqrt(2) i
```

  In the last example the factor is read in as $-2\sqrt{2}i$. Note that imaginary factors make sense only in some cases for some coupling terms (in particular, spin-orbit) in the Cartesian-representation, see Section 2.7.

- `units` This keyword selects the units of measure used for the the object in question. Supported units are: `angstroms` (default) and `bohr` for the bond lengths; `cm-1` (default), `hartree` (aliases are `au`, `a.u.`, and `Eh`), and `eV` (electronvolts) for energies; `debye` (default) and `ea0` (i.e., atomic units) for dipoles; units can appear in any order. Example:

```
units angstrom cm-1 (default for poten, spin-orbit, lambda-doubling etc)
units bohr cm-1
units debye  (default)
units ae0 bohr
```

- `ASSIGN_V_BY_COUNT` The vibrational quantum number v is assigned by counting the rovibronic states of the same 'State', Λ, Σ arranged by increasing energy. The corresponding 'State', Λ, Σ labels are defined using the largest-contribution approach (the quantum labels corresponding to the basis set contribution with the largest expansion coefficient). The keyword should appear anywhere in the body of the input file. The default is to use the largest-contribution approach also to assign the vibrational quantum number (no `ASSIGN_V_BY_COUNT`).

- `values` This keyword starts the subsection containing the numerical values defining the object. For one of the `type`'s corresponding to an analytical function (Section 5), the input between `values` and `end` contains the values of the parameters of the function. The input consists in two columns separated by spaces containing *(i)* a string label identifying the parameter and *(ii)* the value of the parameter (a real number).

  In case of `fitting` (see Section 4) a third column should also be provided; the parameters which are permitted to vary during fitting must have in the third column the string `fit` or, alternatively, the letter `f` or the number 1. Any other string or number (for example, the string `nofit` or the number 0) implies the parameter should be kept at its initial value. In the case of fitting, the keyword `link` can be also appear at the end of each the line; this keyword permits to cross-reference values from different objects and is explained below in this section.

  In the case of objects of type `grid` only two columns are normally needed, a first containing the bond length r_i and a second with the value of the object. Only in the case of object of the `abinitio` (`reference`) type and specified as `grid` a third column should be present specifying the fitting weights (see Section 4).

- `<x|Lz|y>`, `<z|Lz|xy>` (aliases `<a|Lz|b>` and `<1|Lz|2>`) This keyword is sometimes needed when specifying coupling curves between electronic states with $|\Lambda| > 0$ in order to resolve ambiguities in the definition of the degenerate components of each electronic state, see Section 2.7. This keyword specifies the matrix element of the $\hat{L}_z$ operator between the degenerate components of the electronic wave function. Quantum chemistry programs such as Molpro choose the degenerate components so that they transform like the x or y functions (for states with odd $|\Lambda|$, i.e. Π, Φ, $\cdots$, corresponding to symmetry species b_1 and b_2 in the C_{2v} point group) or like z and xy (for states with even $|\Lambda|$, i.e. Δ, Γ, $\cdots$, corresponding to symmetry species a_1 and a_2 in the C_{2v} point group). In this keyword we specify matrix elements of the type $\langle \Pi_x | \hat{L}_z | \Pi_y \rangle$ or $\langle \Delta_z | \hat{L}_z | \Delta_{xy} \rangle$ for the bra and ket states. Examples:

```
<x|Lz|y>   i  -i
<z|Lz|xy> -2i  i
```

These matrix elements are pure imaginary number in the form $\pm|\Lambda|i$. It is the overall $\pm$ sign which Duo needs and cannot be otherwise guessed. As shown in the examples above, each factor should be written in the form $\pm|\Lambda|i$ without any space or `*` sign.

- `link` This special keyword is used in fitting to force a set of parameters (which may be relative to a different object) to have the same value. For example, in a typical situation one may want to fit a set of PECs and to constrain their dissociation (asymptotic) energy to the same value (because they are expected from theory to share the same dissociation channel). After the keyword `link` one should provide three numbers i_1, i_2, i_3 defining the parameter ID, where i_1 identifies the object type (e.g. `poten`, `spin-orbit`, `spin-rot` etc.), i_2 is the object number within the type i_1 and i_3 is the parameter number as it appears after `values`. The ID numbers i_1,i_2, i_3 are specified in the fitting outputs in the form `[i,j,k]`. Example of the input:

```
DE            0.50960000000000E+05    fit     link   1   1   3
```

Example of the corresponding output

```
DE            0.50960000000000E+05    [ 1    1    3 ]
```

- `morphing` This keyword is used for fitting and switches on the morphing method, see Ref. [YLTS15].

- `ZPE` allows to explicitly input the zero-point energy (ZPE) of the molecule (in cm^{-1}). This affects the value printed, as by default Duo prints energy of rovibronic levels by subtracting the ZPE. if not specified, the lowest energy of the first J-block (independent of parity) will be used as appear on the line `Jlist`.

- `fit_factor` This factor (d_λ) is used as a part of the reference *ab initio* curves of the `abinitio` type which (when given) is applied to the corresponding weights assigned to the corresponding values of this object, (see Section 4.3 of [YLTS15]). It is different from `fit_factor` defined within the `Fitting` section.

  Example:

```
abinitio poten 1
name "A 1Pi"
type   grid
lambda 1
mult   1
units bohr cm-1
fit_factor  1e1
values
2.00 32841.37010 0.01
2.20 17837.88960 0.10
2.40 8785.33147 0.70
2.60 3648.35520 1.00
2.70 2107.10737 1.00
2.80 1073.95670 1.00
2.90 442.52180 1.00
3.00 114.94960 1.00
3.10 0.00000     1.00
3.20 48.46120 1.00
3.30 213.34240 1.00
3.40 455.16980 1.00
3.50 739.61170 1.00
3.60 1038.82620 1.00
3.70 1332.46170 1.00
4.00 2059.31119 1.00
4.50 2619.19233 0.30
5.00 2682.84741 0.30
6.00 2554.34992 0.30
8.00 2524.31106 0.30
10.00 2561.48269 1.00
12.00 2575.09861 1.00
end
```

2.7 Representations of the electronic wave functions

As mentioned in the discussion of the `<1|Lz|2>` keyword above, quantum chemistry programs generally use real-valued electronic wave functions which transform according to the irreducible representations of the C_{2v} point group (for heteronuclear diatomics) or of D_{2h} (for homonuclear diatomics). On the other hand Duo internally assumes the electronic wave functions are eigenfunctions of the $\hat{L}_z$ operator, which implies they must be complex valued for $|\Lambda| > 0$. Converting from one representation to the other is simple, as

$$|\pm\Lambda\rangle = \frac{1}{\sqrt{2}}\left[\mp|1\rangle - i|2\rangle\right]. \tag{3}$$

2.8 Example: computing energy levels (one PEC).

Here below there is a commented, minimalistic Duo input file for a single Morse potential; note that the input is case-insensitive. In this particular example we compute the $J = 0$ energy levels of a Morse oscillator $V(r) = D_e(1-e^{-a(r-r_e)})^2$ with $D_e = 40000$ cm^{-1}, $r_e = 1$ Å and $a = 1$ Å^{-1}; the masses of both atoms are both set to 1 Dalton, so that this example is very approximately corresponds to the hydrogen molecule H_2. The exact energy levels are given by $E_n = \omega(n+1/2)\,[1-x_e(n+1/2)]$, $n = 0, \ldots, 33$, with $\omega = a\sqrt{2D_e/\mu} = 2322.593667$ cm^{-1} and $x_e = \omega/(4D_e) = 0.01451621$.

```
(DUO test input)                 comment line
masses 1.00000 1.000000          masses of the two atoms, in Daltons

nstates 1                        number of PECs in the input

jrot 0 10                        total angular momentum J

grid                             specification of the grid
 npoints 250                     number of grid points
 range 0.30, 6.50                r_min and r_max, in Angstroms
end                              end of grid specification

EigenSolver                      options for the Eigensolver
 enermax 35000.0                 print only levels up to enermax cm^-1
 nroots 10                       print only nroots lowest-energy levels
 SYEV                            use SYEV diagonalizer from LAPACK
end                              end of input section EigenSolver

VibrationalBasis                 options for the vibrational uncoupled problem
 vmax 10                         compute vmax+1 vibrational states
END                              end of vibrational specifications

poten 1                          PEC number 1 specification
name "Morse"                     label
type Morse                       functional form: (extended) Morse function
lambda 0                         quantum number Λ
mult 1                           multiplicity, 2S + 1
symmetry +                       only for Σ terms: ± symmetry
units cm-1                       unit for energies
units angstroms                  unit for distances and inverse distances
values                           beginning of specification of the parameters
v0 0.000000                      specification of global shift
r0 1.000000                      specification of r_e
a0 1.000000                      specification of a
De 40000.                        specification of D_e
end                              end of PEC number 1 specification
```

The output has this structure:

- Duo will by default echo the whole of the input file in the output between the lines `(Transcript of the input --->)` and `(<--- End of the input.)` This is useful so that the ouput file will also contain the corresponding input. To avoid echoing the input just add the keyword `do_not_echo_input` anywhere in the input file (but not within an input section).

- Duo will then print its logo, the values of the physical constants (used by the program for such things as conversions between different units) and print some of the global input parameters such as the number of grid points, extent of the grid etc.

- Duo will then print the values of all objects (PECs, dipole moment curves, couplings) on the internal grid. For PECs Duo will also compute and print quantities such as the value of the first few derivatives at the minimum, the corresponding equilibrium spectroscopic constants (harmonic frequency, rigid-rotor rotational constant etc.).

- Duo will solve the $J = 0$ one-dimensional Schrödinger equation for each of the PECs and print the corresponding 'vibrational (contracted)' energies.

- Duo will then solve the full problem (with $J > 0$ and/or all coupling terms activated). In the example above we specified two values of J, namely $J = 0$ and $J = 10$. The $J = 0$ energies will be exactly the same as the 'vibrational (contracted)' ones, as in our example there are no couplings at all.

2.9 Writing the wave functions to disk

Both the vibrational ($J = 0$, uncoupled) basis functions and the coefficients of the expansion of the final ($J > 0$ and coupled) wave functions can be written to disk by including in the Duo input a section with the following structure (see Section 2.2 of the Duo paper [YLTS15]):

```
checkpoint
 eigenfunc save
 filename CO
end
```

Two files will be produced, called in our example `CO_vib.chk` and `CO_vectors.chk`. The file `CO_vib.chk` contains the values of the vibrational basis functions at the grid points and has the following structure:

```
  1          0.000000      1   0   A_1Sigma+
  0.417251193034E-06
  0.913182486541E-06
  0.140429031525E-05
  0.191466765349E-05
  0.243955552609E-05
  0.298913870277E-05
  0.356440215967E-05
  0.417282770822E-05
  0.481737299860E-05
  0.550475969611E-05
  0.623909577848E-05
```

The first line describes the assignment of the vibrational basis function; the first number is a counter over all vibrational wave functions; the second is the energy in cm^{-1}; the third is the 'state' quantum number indicating the electronic state; the fourth is the v vibrational quantum number; finally, the label of the electronic state is reported. What follows is the value of the vibrational wave function at the grid points. The file ends with the line

```
End of contracted basis
```

The file `CO_vectors.chk` contains the values of the expansion coefficients of the final wave functions. The structure is as follows:

```
 Molecule = C-12            O-16
 masses   =      12.000000000000     15.994914504752
 Nroots   =        3
 Nbasis   =        0
 Nestates =        1
 Npoints   =      100
 range   =      0.6500000     3.0000000
Morse_   <- States
      |   # |     J  | p |            Coeff.    | St  vib Lambda  Sigma  Omega|
         1      0.0  1   0.100000000000E+01   1   1   0      0.0      0.0
         1      0.0  1   0.000000000000E+00   1   2   0      0.0      0.0
         1      0.0  1   0.000000000000E+00   1   3   0      0.0      0.0
         2      0.0  1   0.000000000000E+00   1   1   0      0.0      0.0
         2      0.0  1   0.100000000000E+01   1   2   0      0.0      0.0
         2      0.0  1   0.000000000000E+00   1   3   0      0.0      0.0
         3      0.0  1   0.000000000000E+00   1   1   0      0.0      0.0
         3      0.0  1   0.000000000000E+00   1   2   0      0.0      0.0
         3      0.0  1   0.100000000000E+01   1   3   0      0.0      0.0
End of eigenvector
```

The first seven lines are a header containing the names of the atoms, the atomic masses, the number of wave functions computed, the total dimension of the $J > 0$ or coupled Hamiltonian matrix, the number of electronic states in the calculations, the number of grid points and range of the grid (in Å). The numbers following are: # is a counter over the rovibronic wave functions; J is the total[2] angular momentum; p is the total $\pm$ parity (0 for $+$ and 1 for $-$); *Coeff.* is the value of the coefficient in the expansion; following are the quantum number of the basis function (electronic, vibrational, Λ, Σ and Ω).

3 Computing spectra

Absorption or emission spectra as well as line lists, partition functions and other related quantities can be computed by adding in the input file an `intensity` section. Here is an example of its general structure:

[2]Stricly speaking, $\mathbf{J} = \mathbf{R} + \mathbf{L} + \mathbf{S}$ is the sum of the rotational and total electronic angular momenta; it is the total angular momentum only if the nuclear angular momentum $\mathbf{I}$ is zero (or is neglected).

```
intensity
  absorption
  thresh_intens  1e-15
  thresh_coeff   1e-15
  temperature   300.0
  qstat         10.0
  gns           1.0 1.0
  ZPE  931.418890
  selection (rules) 1 1
  J,  0.5, 1.5
  freq-window  -0.001,  25000.0
  energy low   -0.001, 6000.00, upper   -0.00, 30000.0
end
```

If the keyword `intensity` is followed by `none` or `off` then the calculation of intensities is disabled and the section is ignored. This is useful to temporarily avoid the intensity calculation without removing or commenting out the relative input section from the input file. The meaning of the keywords is explained in the following.

3.1 Keywords

- `absorption, emission, partfunc` These keywords define the type of the spectra (absorption or emission) or whether Duo should only compute the partition function. This keyword should appear immediately after `intensity`. Example:

  ```
  absorption
  emission
  partfunc
  ```

- `J` (aliases `Jrot`, `Jlist`) defines the range of rotational angular momentum quantum numbers for which line transitions should be computed. Note that this parameter is independent from `jrot` specified in the general setup (Section 2.1). Example:

  ```
  J 0,10
  ```

  Using the `J` keyword the intensity production can be split into independent `J` $J_{\rm min}, J_{\rm max}$ ranges. In order to prevent overlaps, the range $J_{\rm min}, J_{\rm max}$ does not include transitions $J_{\rm min} \leftrightarrow J_{\rm min}$, except for $J_{\rm min} = 0.5$, where the transitions $0.5 \leftrightarrow 0.5$ are included[3].

- `energy low` and `upper` These keywords to restrict the calculation to transitions between levels satisfying the specified lower and upper energy thresholds (in cm^{-1}): In the following we select transitions for which the lower state is between 0 and 6000 cm^{-1} and the upper state is between 10000 and 30000 cm^{-1}:

[3]Transitions $0 \leftrightarrow 0$ are forbidden

```
energy low 0.0, 6000.00, upper 10000., 30000.0
```

Note that in this context level energies are measured by setting the energy of the lowest energy level to zero, i.e. they do not include the zero-point energy, in contrast with the threshold `enermax` specified in the general setup (Section 2.1).

- `freq-window` specifies a frequency window for line positions (in cm^{-1}). Example:

  ```
  freq-window 0.001, 25000.0
  ```

- `gns` specifies the nuclear statistical weight, which for heteronuclear diatomics is given by $g_{ns} = (2I_1+1)(2I_2+1)$, where I_1 and I_2 are the spins of the two nuclei. In the case of homonuclear diatomics four numbers are expected, one for each symmetry species of the C_{2v}(M) or $C_{2h}(M)$ symmetry groups (see Ref.[YLTS15]). Example:

  ```
  GNS 3.0
  ```

  For the C_{2v}(M) or C_{2h}(M) symmetries associated with the homonuclear molecules the $g_{\rm ns}$ values must be specified for all of the four irreducible representation in the order A_1, A_2, B_1, B_2 and A_g, A_u, B_g, B_u, respectively.

  ```
  GNS 1.0 1.0 0.0 0.0
  ```

- `overlap` allows for printing vibrational overlap integral, aka Franck-Condon factors. The default is not to print (`off`). One can also explicitly switch the overlaps off by adding `off` next to `overlap`:

  ```
  overlap off
  ```

  The format is

  ```
  < i,   v| i',   v'> = value
  ```

  where `i` and `i'` are the electronic state numbers, `v` and `v'` are the vibrational labels and `value` is the overlap:

  $$\langle i, v|i', v'\rangle.$$

- `vib-dipole` prints out vibrational transition moments $\langle i, v|\mu(r)|i', v'\rangle$. By default these values are print out whenever the `intensity` is invoked. In order to switch this option off write `off` next to `vib-dipole`:

  ```
  vib-dipole! off
  ```

  The format is

```
< i,   v| <State | mu | State'> i',   v'> = value
```

where `i` and `i'` are the electronic state numbers, `v` and `v'` are the vibrational labels, `State` is the electronic state label and `value` is the transition dipole moment.

- `Temperature` specifies the temperature (in Kelvin) to be used for the calculation of line intensities. It can be considered as a reference temperature since the Einstein coefficients as the main computational product and are temperature independent. The partition function associated with this `Temperature` should be also specified. Example:

  ```
  temperature  298.0
  ```

- `qstat` (aliases: `part-func` and `Q`). This keyword is to specify the value of the partition function Q for the reference temperature defined by `Temperature`. If not given, Q is computed by Duo.

  Example:

  ```
  qstat 10.0
  ```

- `ZPE` This keyword defines the zero point energy (cm^{-1}) used for the calculation of line intensities, overriding the value specified by the same keyword in the `EigenSolver` input section (see Section 2.4). It is important to explicitly specify `ZPE` when the ground rovibronic state (whose energy defined the ZPE) is not included in the calculation. Omitting this keyword corresponds to using as ZPE the energy of the lowest-lying level used in the calculation. Example:

  ```
  ZPE 931.418890
  ```

- `Thresh-intes` specifies a minimum intensity threshold (in cm/molecule) for printing the transition into the output file as well as into the line list. Example

  ```
  Thresh-intes  1e-35
  ```

- `Thresh-Einstein` specifies a threshold for the Einstein coefficient (in 1/s) for printing out the transition into the output file as well as into the line list. Example:

  ```
  Thresh-Einstein  1e-50
  ```

- `linelist` specifies a file name for writing a line list in the ExoMol format. Example:

  ```
  linelist ScH
  ```

In the example above two files will be written, `ScH.states`, containing a list of energy levels, and `ScH.trans`, containing the line transition data (line positions and Einstein A coefficients). See Refs.[THY13, YLTS15**?**] for the description of the ExoMol format.

4 Fitting

Duo allows the user to modify ('refine') the potential energy curves and other coupling curves by least-squares-fit to 'experimental' energy term values or wavenumbers.

Fitting is, by far, the trickiest part of Duo, both on the part of the program itself and on the part of the user. While the calculation of energy levels and spectra from given PECs, couplings and dipole curves is relatively straighforward (the most critical point being the consistency of the phases specified for the various coupling terms), fitting is often more difficult and may require a trial-and-error approach. Fitting is also the part of Duo where most improvements are to be expected in future new versions.

Example of a fitting section:

```
FITTING
JLIST 2.5,0.5, 1.5 - 11.5, 22.5 - 112.5
itmax 30
fit_factor  1e6
output alo_01
fit_type dgelss
lock      5.0
robust      0.001
energies    (J parity NN  energy ) (e-state v ilambda isigma omega  weight)
 0.5   +   1     0.0000  1  0  0   0.5   0.5    100.000
 0.5   +   2   965.4519  1  1  0   0.5   0.5      7.071
 0.5   +   3  1916.8596  1  2  0   0.5   0.5      5.774
 0.5   +   4  2854.2366  1  3  0   0.5   0.5      5.000
 0.5   +   5  3777.5016  1  4  0   0.5   0.5      4.472
 0.5   +   6  4686.7136  1  5  0   0.5   0.5      4.082
 0.5   +   7  5346.1146  2  0  1  -0.5   0.5    100.000
end
```

4.1 Keywords

- `FITTING` This keyword marks the beginning of the fitting input section. The whole section can be deactivated by putting `none` or `off` next to the keyword `FITTING`. This is useful to disable the fitting without removing the input block from the input file.

- `jlist` (aliases are `jrot` and `J`) This keyword allows the user to specify the values of the J quantum number to be used in the fit. It superseedes the corresponding `jrot` keyword specified in the general setup (Section 2.1). Individual values of J can be separated by spaces or commas, while ranges are specified by two values separated by a hyphen (hyphens should be surrounded by spaces). The first J value is used to determine ZPE. For example

```
   JLIST  1.5, 5.5, 15.5 - 25.5, 112.5
```

selects the values 1.5, 5.5, all values from 15.5 to 25.5 and the value 112.5.

- `itmax` (alias `itermax`) An integer defining the maximum number of fitting iterations. Setting `itmax` to zero implies that no fit will be performed (straight-through calculation); however, the differences between the computed energy levels (or frequences) and the reference (experimental) ones will be printed. Example:

  ```
  itmax 15
  ```

- `fit_factor` This factor is used when reference curves of the `abinitio` type are included in the fit and used to define the importance of the energy/frequency data relative to the reference `abinitio` data. This factor is applied to all energy (frequencies) weight factors $w_i^{\rm en}$(see Section 4.3 of [YLTS15]).

  When the factor is very large (e.g. 10^6, like in the example above) the penalty for differing for the reference curve is very small, so that only the 'obs. - calc.' for energy levels matter. Vice versa, if the factor is very small (e.g. 10^{-6}) the fit is constrained so that the fitted curves stay very close to the reference (`abinitio`) ones. When this number is extremely small (smaller than 10^{-16}) the experimental data are completely ignored and the fit is performed to the *ab initio* values only. Thus this feature also allows one to use the `FITTING` section for building analytical representations (see `type`-s currently available) of different objects by fitting to the corresponding *ab initio* or reference data provided in the `abinitio`-sections of the input.

  Example:

  ```
  fit_factor 1e2
  ```

- `lock` denotes the threshold (cm^{-1}) for which the quantum numbers are locked. The quantum numbers defining 'state', v, $|\lambda|$, $|\sigma|$ and $|\Omega|$ will be used to identify and lock the energy value in place of the row number within the J/parity block. When negative, the match is reconstructed based solely on the closest value within the lock-threshold given. If the match within the lock-region is not found, the row J/parity number is used to match the theoretical and experimental energies. For example to match and lock to the calculated energy to the 'experimental' one based on the quantum numbers within 20 cm^{-1} use:

```
lock 20.0
```

- `thresh_obs-calc` This keywords triggers switching off states from the fit if the obs.-calc. residuals become larger than the threshold specified. This feature is useful in case of multiple swapping of the states during the fits and even the lock `option` does not help. The default value is zero (the feature is off).

- `robust` This keyword allows the user to switch on Watson's robust fitting procedure [Wat03]: "0" is 'off', any other positive value is 'on' and defines the target accuracy of the fit as given by the weighted root-mean-square error. The `robust`-value is the targeted accuracy (obs.-calc.) of the fit. Example:

```
robust 0.01
```

- `target_rms` is to define the convergence threshold (cm^{-1}) for the total, not-weighted root-mean-squares (rms) fitting error. Example:

```
target_rms 0.1
```

- `output` is the *filename* for the files *name*.en, *name*.freq and *name*.pot, containing detailed information on the fitting, including the fitting residuals for each iteration. See Section 4.1.2 below for more details. Example:

```
output NaH_fit
```

- `energies` This keyword starts the section with the energy levels to be fit to (e.g., obtained from an analysis of the experimental line positions). Energy levels are written as in the following example:

```
energies
   0.5   +    1     0.0000 1  0 0 0.5  0.5  1.00
   0.5   +    2   965.4519 1  1 0 0.5  0.5  0.90
   0.5   +    3  1916.8596 1  2 0 0.5  0.5  0.80
end
```

where the meaning of the various quantities is as follows; col.1 is the total angular momentum quantum number J; col. 2 either the total parity $\tau = \pm$ or the e/f parity; col. 3 is a running number N couting levels in ascending order of the energy within a (J, τ) symmetry block; col. 4 is the energy term value $\tilde{E}$, in cm^{-1}; col. 5 is the electronic state index 'state', as numbered in the `poten` sections; col. 6 is the vibrational quantum number v; col. 7 is the projection of the electronic angular momentum Λ for the state in question (an integer); col. 8 is the projection of the total electronic spin Σ (integer of half integer); col. 9 is the projection of the total angular momentum Ω (integer of half integer); col. 10 is the weight W of the experimental energy in question (a real and positive number usually given by σ^{-2}, where σ is the uncertainty of the energy level).

- `frequency` (aliases are `frequencies` and `wavenumbers`) This keyword works similarly to the `energies` keyword above but starts the section specifying the wavenumbers (i.e., line positions) to be fitted to. Example:

```
frequencies
 0.0  +   2 0.0 +  1    720.0000   2  0   1  -1.0   0.5    1  0   0   0.0   0.0  1.00
 2.0  +  17 3.0 -  2   5638.1376   4  0   0   1.0   1.0    2  0  -1  -1.0  -2.0  1.00
 4.0  +  17 5.0 -  2   5627.5270   4  0   0   1.0   1.0    2  0  -1  -1.0  -2.0  1.00
 4.0  +  18 7.0 -  2   5616.7976   4  0   0   0.0   0.0    2  0  -1  -1.0  -2.0  1.00
end
```

  The meaning of the quantities in each line are the following (see the keyword `energies` above for an explanation of the symbols. The prime/double prime symbol correspond to upper/lower level): J', τ', N', J'', τ'', N''; frequency (cm^{-1}); state$'$, v', Λ', Σ', Ω'; state$''$, v'', Λ'', Σ'', Ω''; weight.

- `off`,`none` is used to switch off `Fitting`, `Intensity` or `Overlap`, when put next to these keywords.

4.1.1 Structure of the fitting output

During fitting Duo will print for each iterations the fitting residuals using the following structure (the first line with numbers 1 to 20 is not part of the output but serves as a legend):

```
1  2    3   4          5           6           7        8   9    10   11    12    13    14  15  16    17    18    19    20

1  1   0.5  +     0.0000      0.0000      0.0000  0.60E-02  1     0    1  -0.5   0.5   0.5   1   0     1  -0.5   0.5   0.5
2  2   0.5  +  1970.2743   1970.3983     -0.1240  0.59E-02  1     1    1  -0.5   0.5   0.5   1   1     1  -0.5   0.5   0.5
3  3   0.5  +  3869.6639   3869.7934     -0.1295  0.30E-02  1     2    1  -0.5   0.5   0.5   1   2     1  -0.5   0.5   0.5
4  4   0.5  +  5698.7392   5699.2951     -0.5559  0.20E-02  1     3    1  -0.5   0.5   0.5   1   3     1  -0.5   0.5   0.5
5  1   0.5  -     0.1001      0.0000      0.1001  0.60E-02  1     0   -1   0.5  -0.5   0.5   1   0    -1   0.5  -0.5   0.5
6  2   0.5  -  1970.4156   1970.3983      0.0173  0.59E-02  1     1   -1   0.5  -0.5   0.5   1   1    -1   0.5  -0.5   0.5
```

The meaning of the quantities in the various columns is as follows; col.1 is a simple line counter i counting over all lines; col.2 is a counter N counting lines within each J,τ symmetry block; col. 3 is J; col. 4 is the parity τ; col.5,6 are, respetively, the reference ('Observed') and the calculated value of the line position; col.7 is the difference between observed and computed line positions; col. 8 is the weight assigned to the transition in the fit; col. 9 to 14 are the quantum numbers of the lower state: 'state', v, Λ, Σ, Ω and S; col. 15 to 20 are the quantum numbers for the upper state (same definition as for columns 9 to 14).

4.1.2 The auxiliary files .en, .freq, .pot

The files *name*.en contains all computed term values together with the theoretical quantum numbers, compared to the experimental values, when available, along with the 'experimental' quantum numbers as specified in the `fitting` section, for all iterations of the least-squares fit. Here *name* is the file name as speficied by the `output` keyword. The output is in the same format as in the standard output file (see above) with the difference that it contains all calculated values (subject of the `nroots` keyword, see Section 2.4). An asterisk '*' at the end of the line indicates that either the theoretical and 'experimental' assignments don't agree or a residuals obs.-calc. is too large (large than the `lock` parameter).

The frequency file *name*.freq with the keyword `frequencies`. It has a similar structure as the standard output, with the difference that for each transition from the `frequency` section the program will estimate additional transition frequencies involving energies (both lower and upper) which are within `lock` cm^{-1} of the corresponding input values. This is done to facilitate the search for possible miss-assignment, which is typical for transitions. This is printed out for all iterations.

The file *name*.pot ('potential') contains the residuals between the fitted and the reference curve (if specified by an `abinitio` object). The file is overwritten at each iteration.

5 Analytical functions

This section shows examples of the definitions of the analytical functions supported in Duo as described in Ref. [YLTS15].

1. `EMO` The Extended Morse Oscillator (EMO) which is as also used by LEVEL.

$$V(r) = T_{\rm e} + (A_{\rm e} - T_{\rm e}) \left(1 \ - \ \exp\left\{-\beta_{\rm EMO}(r)(r-r_{\rm e})\right\}\right)^2, \tag{4}$$

   which has the form of a Morse potential with a exponential tail and the distance-dependent exponent coefficient

$$\beta_{\rm EMO}(r) \ = \ \sum_{i=0} a_i y_p^{\rm eq}(r)^i, \tag{5}$$

   expressed as a simple power series in the reduced variable [vRB84]:

$$y_p^{\rm eq}(r) \ = \ \frac{r^p - r_{\rm e}^p}{r^p + r_{\rm e}^p} \tag{6}$$

   with p as a parameter. This form guarantees the correct dissociation limit and allows for extra flexibility in the degree of the polynomial on the left or on the right sides of a reference position $R_{\rm ref}$ which we take at $R_{\rm ref} = r_{\rm e}$. This is specified by the parameters $N = N_l$ (N_r) and $p = p_l$ (p_r), respectively.

   Example:

```
poten 2
name "a 3Piu"
symmetry u
type  EMO
lambda 1
mult   3
values
Te          0.81769829519421E+03
Re          0.13115676812526E+01
Ae          0.50960000000000E+05
RREF       -0.10000000000000E+01
PL          4
PR          4
NL          2
NR          3
a0          0.21868146887665E+01
a1          0.88875855351916E-01
a2          0.84932592800179E-01
a3          0.23343175838290E+00
end
```

Taylor expansion around r_0:

$$V(r) = T_\mathrm{e} + (A_\mathrm{e} - T_\mathrm{e}) a_0^2 (r - r_0)^2 + (A_\mathrm{e} - T_\mathrm{e}) \left(p \frac{a_0 a_1}{r_e} - a_0^3 \right) (r - r_0)^3 + \cdots$$

2. `MLR` Morse Long-Range (MLR) function [LHJ06, LH07, Le 07, LHTL11]:

$$V(r) = T_\mathrm{e} + (A_\mathrm{e} - T_\mathrm{e}) \left(1 - \frac{u_\mathrm{LR}(r)}{u_\mathrm{LR}(r_e)} \; \exp\{ -\beta_\mathrm{MLR}(r) y_p^\mathrm{eq}(r) \} \right)^2 , \qquad (7)$$

where the radial variable y_p^eq in the exponent is given by Eq. (6), the long-range potential $u_\mathrm{LR}(r)$ by

$$V(r) \to u_\mathrm{LR}(r) = \sum_n \frac{C_n}{r^n} \qquad (8)$$

while the exponent coefficient function

$$\beta_\mathrm{MLR}(r) = y_p^\mathrm{ref}(r) \, \beta_\infty \;+\; \left[1 - y_p^\mathrm{ref}(r) \right] \; \sum_{i=0} a_i [y_q^\mathrm{ref}(r)]^i \qquad (9)$$

is defined in terms of two radial variables which are similar to y_p^eq, but are defined with respect to a different expansion center r_ref, and involve two different powers, p and q. The above definition of the function $\beta_\mathrm{MLR}(r)$ means that:

$$\beta_\mathrm{MLR}(r \to \infty) \;\equiv\; \beta_\infty \;=\; \ln[2 D_\mathrm{e} / u_\mathrm{LR}(r_\mathrm{e})]. \qquad (10)$$

Example:

```
poten 6
name "d 3Pig"
symmetry g
```

```
lambda 1
mult   3
units bohr cm-1
type  MLR
values
Te         0.20151357236994E+05
RE         0.12398935933004E+01
AE         0.50960000000000E+05         link   1   1   3
RREF      -0.10000000000000E+01
P          0.40000000000000E+01
NL         0.20000000000000E+01
NR         0.80000000000000E+01
b0         0.30652655627150E+01
b1        -0.93393246763924E+00
b2         0.45686541184906E+01
b3        -0.37637923145046E+01
b4        -0.41028177891391E+01
b5         0.00000000000000E+00
b6         0.00000000000000E+00
b7         0.00000000000000E+00
b8         0.00000000000000E+00
Binf       1.000000000000000000
a1         0.00000000000000E+00
a2         0.00000000000000E+00
a3         0.00000000000000E+00
a4         0.00000000000000E+00
a5         0.00000000000000E+00
a6         192774.
a7         0.00000000000000E+00
a8         0.00000000000000E+00
end
```

3. `Surkus` (alias `BobLeroy`) Šurkus-polynomial expansion [vRB84]:

$$V(r) = T_{\rm e} + (1 - y_p^{\rm eq}) \sum_{i \geq 0} a_i [y_p^{\rm eq}]^i + y_p^{\rm eq} a_{\rm inf}, \qquad (11)$$

where $y_p^{\rm eq}$ is the Šurkus variable (6) and $a_{\rm inf}$ is the asymptote of the potential at $r \to \infty$.

Example:

```
spin-orbit  2 2
name "<Lambda=+1,S=1 (a2Pi)|LSZ|+1 (a2Pi),S=1>"
spin   0.5 0.5
lambda 1 1
sigma  0.5 0.5
type  BOBLEROY
units  cm-1
factor    1.0   (0, 1 or i)
N 9
values
re          0.17700000000000E+01
rref       -0.10000000000000E+01
P           0.20000000000000E+01
NT          0.30000000000000E+01
a0         -0.63452015232176E+02
a1         -0.20566444179565E+01
a2         -0.13784613913938E+02
a3          0.00000000000000E+00
```

```
ainf       -0.56030500000000E+02
end
```

4. `Surkus-damp` (alias `BobLeroy_damp`) Šurkus-polynomial expansion with a damping function [vRB84]:

$$V(r) = T_\mathrm{e} + \left[(1 - y_p^\mathrm{eq}) \sum_{i\geq 0} a_i [y_p^\mathrm{eq}]^i + y_p^\mathrm{eq} a_\mathrm{inf} \right] f^\mathrm{damp} + t^\mathrm{damp}(1 - f^\mathrm{damp}), \tag{12}$$

where the damping function is defined by

$$f^\mathrm{damp} = 1 - \tanh[\alpha(r - r_0)],$$

and t^damp, r_0 and α are parameters.

Example:

```
spin-orbit  2 2
name "<Lambda=+1,S=1 (a2Pi)|LSZ|+1 (a2Pi),S=1>"
spin   0.5 0.5
lambda 1 1
sigma  0.5 0.5
type  BOBLEROY
units  cm-1
factor    1.0   (0, 1 or i)
N 9
values
re          0.17700000000000E+01
rref       -0.10000000000000E+01
P           0.20000000000000E+01
NT          0.30000000000000E+01
a0         -0.63452015232176E+02
a1         -0.20566444179565E+01
a2         -0.13784613913938E+02
a3          0.00000000000000E+00
ainf       -0.56030500000000E+02
tdamp       0.00000000000000E+00
r0          0.10000000000000E+01
alpha       0.30000000000000E+01
end
```

5. `Morse` A polynomial expansion in the Morse variable $y_\mathrm{M} = 1 - e^{-a(r-r_0)}$ is used

$$V(r) = T_\mathrm{e} + (A_\mathrm{e} - T_\mathrm{e}) y_\mathrm{M}^2 + \sum_{i=1}^{N} a_i\, y_\mathrm{M}^{i+2}. \tag{13}$$

Example

```
poten 1
name "X 1Sigmag+"
symmetry g +
type   MORSE
lambda 0
mult   1
units bohr cm-1
values
```

```
TE              0.00000000000000E+00
RE              0.12423216077595E+01
a               0.20372796052933E+01
AE              0.73955889175514E+05
A1             -0.62744302960091E+04
A2             -0.57683579529693E+04
end
```

6. `Morse_damp`

$$V_(r) = T_\mathrm{e} + (A_\mathrm{e} - T_\mathrm{e}) y_\mathrm{M}^2 + e^{-d_\mathrm{damp}(r-r_\mathrm{e})^4} \sum_{i=1} a_i \left(\frac{r-r_\mathrm{e}}{r+r_\mathrm{e}}\right)^{i+2}. \quad (14)$$

   Example:

```
poten 6
name "d 3Pig"
symmetry g
lambda 1
mult   3
units bohr cm-1
type  Morse_damp
values
Te      20121.09769
re      0.12545760270976E+01
Ae      0.50937907750000E+05        link   1   1   3
a0      0.30398932686950E+01
DAMP    0.10000000000000E-02
a1      0.11437702960146E+05
a2     -0.36585731834570E+03
a3     -0.20920472718062E+05
a4      0.90487097982036E-03
a5      0.00000000000000E+00
a6      0.00000000000000E+00
a7      0.00000000000000E+00
a8      0.00000000000000E+00
end
```

7. `Modified-Morse` (alias `MMorse`)

$$V_(r) = T_\mathrm{e} + (A_\mathrm{e} - T_\mathrm{e}) \frac{\left[1 - \exp\left(-\sum_{i=0} a_i \xi^{i+1}\right)\right]^2}{\left[1 - \exp\left(-\sum_{i=0} a_i\right)\right]^2}, \quad (15)$$

   where $\xi = (r - r_\mathrm{e})/(r + r_\mathrm{e})$.

   Example:

```
poten 8
name "Bp 1Sigmag+"
symmetry g +
lambda 0
mult   1
type  MMorse
values
Te            1.5408840263E+04
rE            1.3778208709E+00
Ae            5.0937907750E+04              link   1   1   3
```

```
a0           6.2733066935E+00
a1           1.4954972843E+01
a2           4.5160872659E+01
end
```

where the value A_e is ‘linked’ to the corresponding value of `poten 1`.

8. `Polynomial` selects a polynomial expansion in the variable $y = (r - r_0)$

$$V(r) = T_\mathrm{e} + a_1 y + a_2 y^2 + \cdots \tag{16}$$

Example:

```
spin-orbit  2 2
name "<+1,S=1 (a3Pi)|LSZ|+1  (a3Pi),S=1>"
spin   1.0 1.0
sigma  1.0 1.0
lambda 1 1
type  polynom
factor   1
values
a0          14.97
re          1.3
a1          0.0
end
```

Taylor expansion around r_0:

$$V(r) = T_\mathrm{e} + a_1(r - r_0)^2 + a_2(r - r_0)^2 + a_3(r - r_0)^3 + \cdots$$

9. `Dunham` selects a polynomial expansion in the Dunham variable $y = (r - r_0)/r_0$

$$V(r) = T_\mathrm{e} + a_0 y^2 \left(1 + a_1 y + a_2 y^2 + \cdots\right) \tag{17}$$

Example:

```
poten 1
name "X 2 Delta"
lambda 2
mult   2 type   Dunham values
Te              0.00000
Re              1.4399282269779912
a0         123727.20496894409      (= omega**2 / 4 B)
a2             -2.31
a3              3.80
a4             -6.00
a5              5.00
end
```

Comments As a function form `Dunham` is equivalent to a `Polynomial` object with the linear term absent and a redefinition of the expansion coefficients; the comments given for `Polynomial` also apply to `Dunham`.

Taylor expansion around r_0:

$$V(r) = T_\mathrm{e} + \frac{a_0}{r_0^2}(r-r_0)^2 + \frac{a_0 a_1}{r_0^3}(r-r_0)^3 + \cdots$$

10. `SPF` selects a polynomial expansion in the the so-called Simons, Parr and Finlan variable $y = (r-r_0)/r$

$$V(r) = T_\mathrm{e} + a_0 y^2 \left(1 + a_1 y + a_2 y^2 + \cdots\right) \tag{18}$$

Example:

```
poten 1
name "X 2Sigma+"
symmetry +
type   SPF
lambda 0
mult   2
values
Te         0.00000000000000E+00
RE         0.16292698613903E+01
a1         0.37922070444743E+06
a2         0.00000000000000E+00
a3        -0.53314483965665E+01
a4         0.00000000000000E+00
a5         0.19407192336518E+02
a4         0.00000000000000E+00
a5        -0.17800496953835E+03
end
```

Taylor expansion around r_0:

$$V(r) = T_\mathrm{e} + \frac{a_0}{r_0^2}(r-r_0)^2 + \frac{a_0 a_1 - 2a_0}{r_0^3}(r-r_0)^3 + \cdots$$

Behaviour for $r \to +\infty$:

$$\begin{aligned} V(r) =& T_\mathrm{e} + a_0 \left(1 + \sum_{i=1}^{N} a_i\right) \\ & - \frac{a_0 r_0}{r}\left(2 + \sum_{i=1}^{N}(i+2)a_i\right) + \cdots \end{aligned} \tag{19}$$

where N is the maximum exponent included in the expansion. For long r the potential goes to a constant value; convergence to the constant is of the $1/r$ type (correct for ions but too slow for neutral molecules).

Behaviour for $r \to 0$:

$$V(r) = a_0 a_N \left(\frac{r_0}{r}\right)^{N+2} + \cdots \tag{20}$$

The coefficient a_0 is definitely positive, but a_N can be positive and negative, so that $V(r)$ can go to $\pm\infty$ for short r.

11. `M-S` (Murrell-Sorbie) [MS74]

$$V(r) = A_\mathrm{e} - (A_\mathrm{e} - T_\mathrm{e})\left(1 + a_1\rho + a_2\rho^2 + a_3\rho^3 + \ldots\right) e^{-a_1\rho}, \tag{21}$$

where $\rho = r - r_\mathrm{e}$.

Example:

```
poten 4
name "B 2Sigma"
symmetry -
type  M-S  (Murrell-Sorbie)
lambda 0
mult   2
values
V0           21000.0
RE           1.6
DE           25653.27131
a1           2.81468
a2           1.68719
a3           0.757787
a4           -0.5963168
a5           -0.54596343
a6           0.20611664
end
```

Taylor expansion around r_e:

$$\begin{aligned} V(r) =& T_\mathrm{e} + \frac{A_\mathrm{e} - T_\mathrm{e}}{2}(a_1^2 - 2a_2)(r - r_0)^2 \\ &+ \frac{A_\mathrm{e} - T_\mathrm{e}}{3}(a_1^3 - 3a_1a_2 + 3a_3)(r - r_0)^3 + \cdots \end{aligned} \tag{22}$$

Behaviour for $r \to +\infty$:

$$V(r) =A_\mathrm{e} - a_N(A_\mathrm{e} - T_\mathrm{e})(r - r_e)^N e^{-a_1(r-r_e)} + \cdots \tag{23}$$

where N is the maximum exponent included in the expansion. For long r the potential goes to the constant value A_e, and the aymptotic behavior is determined by the coefficients of the term with the highest exponent.

12. `Chebyshev` selects an expansion in Chebyshev polynomials in the variable $y = [r - (b + a)/2]/[(b - a)/2]$. The scaled variable y ranges from -1 to 1 for r in $[a, b]$. The expansion is

$$V(r) = a_0 + a_1T_1(y) + a_2T_2(y) + \cdots \tag{24}$$

Example:

```
spin-orbit  2 2
name "<+1,S=1 (a3Pi)|LSZ|+1  (a3Pi),S=1>"
spin   1.0 1.0
```

```
type  chebyshev
factor   1
values
a              0.80000000000000E+00
b              0.26500000000000E+01
A0            -0.25881057805341E+02
A1             0.82258425882627E+01
A2             0.52391700137878E+00
A3             0.28483394288286E+01
A4            -0.15136422837793E+00
A5             0.97553692867070E-01
A6            -0.25825811071417E+00
A7            -0.69105144347567E-01
A8            -0.44700771508442E-01
A9             0.11793957297111E-01
A10            0.16403055376257E-01
A11            0.92509900186428E-02
A12            0.50789943150707E-02
A13           -0.39439903216016E-03
end
```

13. `COSH-POLY` This function can be used as a coupling for a diabatic representation of potentials characterised by an avoiding crossing and is given [Var09]:

$$F(r) = F_0 + \frac{\sum_{i=0}^{N} a_i\,(r - r_{\rm ref})^i.}{\cosh\beta(r - r_{\rm ref})}. \tag{25}$$

Example

```
diabatic  1 8
name "<X1Sigmag+|D|Bp 1Sigmag+>"
spin   0.0 0.0
lambda  0  0
type  COSH-poly
units  cm-1
factor    i   (0, 1 or i)
values
v0           0.0000
beta         5.62133
RE           1.610505
B0          -0.307997
B1           0.0000000000E+00
B2           0.0000000000E+00
BINF         0.0000000000E+00
end
```

14. `POLYNOM_DECAY_24` This function is similar to `Surkus` expansion

$$F(r) = \sum_{k=0}^{N} B_k\, z^k (1 - \xi_p) + \xi_p\, B_\infty, \tag{26}$$

where z is either taken as the damped-coordinate given by:

$$z = (r - r_{\rm ref})\, e^{-\beta_2 (r - r_{\rm ref})^2 - \beta_4 (r - r_{\rm ref})^4}, \tag{27}$$

see also **?**] and **?**]. Here $r_{\rm ref}$ is a reference position equal to $r_{\rm e}$ by default and β_2 and β_4 are damping factors. When used for morphing, the parameter B_∞ is usually fixed to 1.

6 Eigenfunctions

The computed eigenfunctions can be printed out into a sperate file (checkpoint). This option can be enabled via the section `Checkpoint`:

```
Checkpoint
 eigenvectors save
 Filename xxxxx
End
```

The keywords `eigenvectors save` are to stitch the corresponding checkpointing on.

The eigenfunction-checkpoints consist of two files, `eigen_vectors.chk` and `eigen_vib.chk`. The file `eigen_vib.chk` contains the vibrational part of the basis set in the grid representation in the following format (example):

```
    1           0.000000       1    0    A1Sigma+
  0.124132175316E-13
 -0.952336606315E-14
  0.982508543282E-14
......
```

where the each basis function is given in a block. The first line specifies the sate (number, energy, electronic state and vibrational quantum number) followed by the grid values.

The file `eigen_vectors.chk` contains the expansion coefficients of the eigenfunction in terms of the Duo ro-vibronic basis set functions using the following format (example):

```
 Molecule = Ca-40           O-16
 masses   =      39.962590600000     15.994914630000
 Nroots   =       10
 Nbasis   =       50
 Nestates =        5
 Npoints   =      501
 range   =      1.0000000     4.0000000
X1Sigma+, Ap1Pi, a3Pi, b3Sigma+, A1Sigma+,    <- States
  |  # |    J  | p |           Coeff.   | St vib Lambda  Sigma  Omega ivib|
      1      0.0  0   0.999999782551E+00   1   1   0      0.0      0.0    0
      .....
```

Here the first eight lines represent a 'signature' of the spectroscopic model (atoms, masses, specification of the basis), the line 9 is a header followed by the records with the eigen-coefficients and corresponding quantum numbers and labels using the following format: running number within the J,parity(p)-block i, J, p, the coefficient $C_i^{J,p}$, State, v, Λ, Σ and vibrational basis set number (a combined number representing the contracted vibrational basis set function from for all electronic states combined).

The optional keyword `Filename` (alias `Vector-Filename`) is to change the checkpoint-prefix `eigen` to `filename`. The default name is `eigen_vectors.chk`.

7 Running and compiling Duo

Duo is provided both as source code and as compiled executables for Linux and Microsoft Windows. Using the executables is the easiest way to run Duo. Duo works from the command line (also known as the 'terminal' or 'command prompt'). A Duo input is a plain text input file which is fed into the program with a command of the kind

```
./duo.exe < ./inputs/my_input.inp > my_output.txt
```

If you accidentally start Duo without specifying any input nothing will happen and you will be temporarely 'stuck'; to terminate Duo press 'C' while holding down the Ctrl key.

Please note that Duo is still in active development and new versions with bug fixes and new functionalities are expected to appear regularly. If you have found a bug or you would like to make a comment please do not hesitate to contact the authors (contact details are reported in the first page of this manual).

7.1 Compilation

You may need to re-compile Duo if the provided executables do not work on your system or, for example, if you want to make modifications to the program. Duo makes use of some Fortran 2003 features and therefore requires a compiler with (at least partial) support for Fortran 2003. At the time of writing (July 2015) there are two freely-available Fortran compilers (for Windows, Linux and OS X) which can be used for compiling Duo, namely gfortran and g95. If you are a university student[4] or if you use the compiler to develop open-source products[5] you may qualify for a free version of Intel Parallel Studio XE, which includes the Intel Fortran compiler (Windows, Linux and OS X), or for Oracle Solaris Studio[6] for Linux. Other compilers may be available to you through your institution. Lists of Fortran compilers can be found on the Internet[7,8]. The Fortran 95 Windows compiler Silverfrost FTN95 v.7.20 is also available for free for personal use[9] but its support for Fortran 2003 is very incomplete and Duo will *not* work with this compiler at this time.

Duo has been tested with the Intel Fortran Compiler v. 12.1 under Linux and Windows 8, with the Portland Group Fortran compiler v. 13.1, with the NAG Fortran compiler v 5.2 under Linux and Windows 8, with g95 v. 0.94 under Windows 8 and with gfortran v.4.9.2 under Windows 8, Linux and OS X.

[4]https://software.intel.com/en-us/qualify-for-free-software/student
[5]https://software.intel.com/en-us/qualify-for-free-software/opensourcecontributor
[6]www.oracle.com/technetwork/server-storage/solarisstudio/downloads/index.html
[7]See http://fortranwiki.org/fortran/show/compilers
[8]https://en.wikipedia.org/wiki/List_of_compilers#Fortran_compilers
[9]http://www.silverfrost.com/32/ftn95/ftn95_personal_edition.aspx

7.2 Compilation under Windows with gfortran

For compiling Duo under Windows using gfortran four steps are typically needed:

1. Installation of a Linux-like enviroment such as MinGW or Cygwin.
2. Installation of the compiler.
3. Installation of the BLAS/LAPACK libraries.
4. Compilation of Duo.

Step 1 will make available under Windows a terminal window in which one can use standard Linux command such as `ls` or `grep`; in particular, it will make available the command `make` which will be used to compile Duo. In the following we describe how to use MinGW and gfortran. If you need a Windows text editor for editing Duo input files or the source code we recommend Notepad^{++}, which is free[10].

7.2.1 Installing MinGW and gfortran

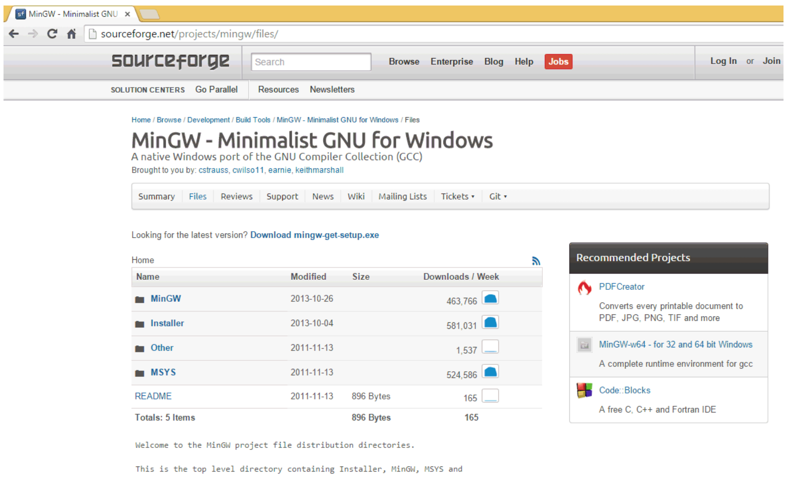
Go to the MinGW section on SourceForge[11], click on 'Download mingw-get-setup.exe' (see fig. 2) and download, install and run the MinGW Installation manager. In the 'Basic Setup' section select 'mingw32-gcc-fortran', as shown in fig. 3, if you want to use gfortran for compiling Duo and go on with the installation. We will assume in the following that MinGW is installed in the default directory `C:\MinGW\`.

Once the installation has finished MinGW and the gfortran compiler should be installed. To check if gfortran is correctly installed open a Command Prompt window[12] and type
`gfortran -dumpversion`
You should see an output similar to the one shown in fig.4. If instead of the output of fig. 4 you get an error message then try the following. Type
`cd c:\MinGW\bin` and try again to type `gfortran -dumpversion`. If gfortran now works then you have to manually add the directory c:\MinGW\bin in your `PATH` environment variable. How exactly this is done depends on the Windows version in use; we describe here how this is done with Windows 8.1, for other

[10]https://notepad-plus-plus.org/

[11]http://sourceforge.net/projects/mingw/files/

[12]Depending on the Window language in use the Command Prompt will have different names such as 'Símbolo del sistema' (Spanish), 'Invite de commandes' (French), 'Prompt dei comandi' (Italian), 'Eingabeaufforderung' (German), 'Wiersz polecenia' (Polish) and so on. Independently from the language in use it can be executed by writing in the Windows 'Run' box `cmd.exe`. In version of Windows prior to Windows XP the Command Prompt was called 'MS-DOS prompt'.



MinGW - Minimalist GNU for Windows

A native Windows port of the GNU Compiler Collection (GCC)

Brought to you by: cstrauss, cwilso11, earnie, keithmarshall

Looking for the latest version? Download mingw-get-setup.exe

Home

Name	Modified	Size	Downloads / Week
MinGW	2013-10-26		463,766
Installer	2013-10-04		581,031
Other	2011-11-13		1,537
MSYS	2011-11-13		524,586
README	2011-11-13	896 Bytes	165
Totals: 5 Items		896 Bytes	165

Recommended Projects

PDFCreator
Converts every printable document to PDF, JPG, PNG, TIF and more

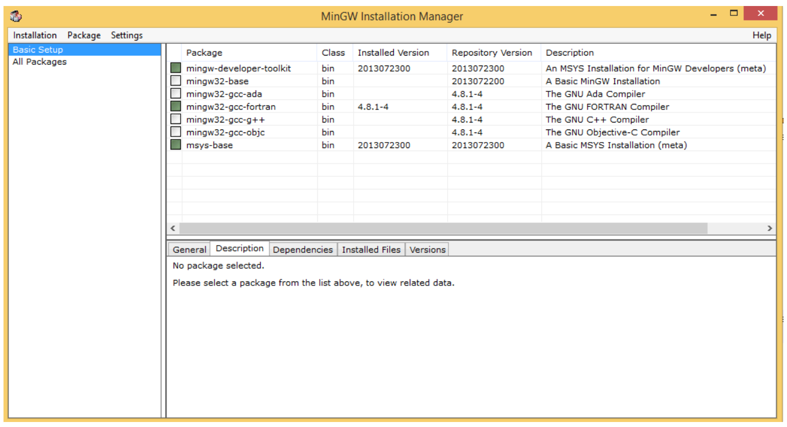
MinGW-w64 - for 32 and 64 bit Windows
A complete runtime environment for gcc

Code::Blocks
A free C, C++ and Fortran IDE

```
Welcome to the MinGW project file distribution directories.

This is the top level directory containing Installer, MinGW, MSYS and
```

Figure 2: Installation of MinGW.

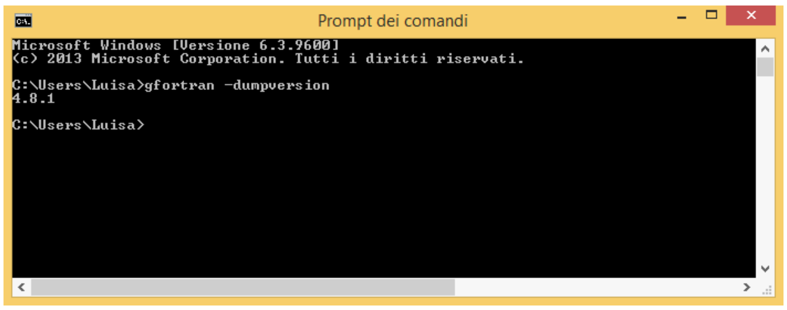


	Package	Class	Installed Version	Repository Version	Description
🟩	mingw-developer-toolkit	bin	2013072300	2013072300	An MSYS Installation for MinGW Developers (meta)
☐	mingw32-base	bin		2013072200	A Basic MinGW Installation
☐	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
🟩	mingw32-gcc-fortran	bin	4.8.1-4	4.8.1-4	The GNU FORTRAN Compiler
☐	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
☐	mingw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
🟩	msys-base	bin	2013072300	2013072300	A Basic MSYS Installation (meta)

No package selected.

Please select a package from the list above, to view related data.

Figure 3: Installation of MinGW.

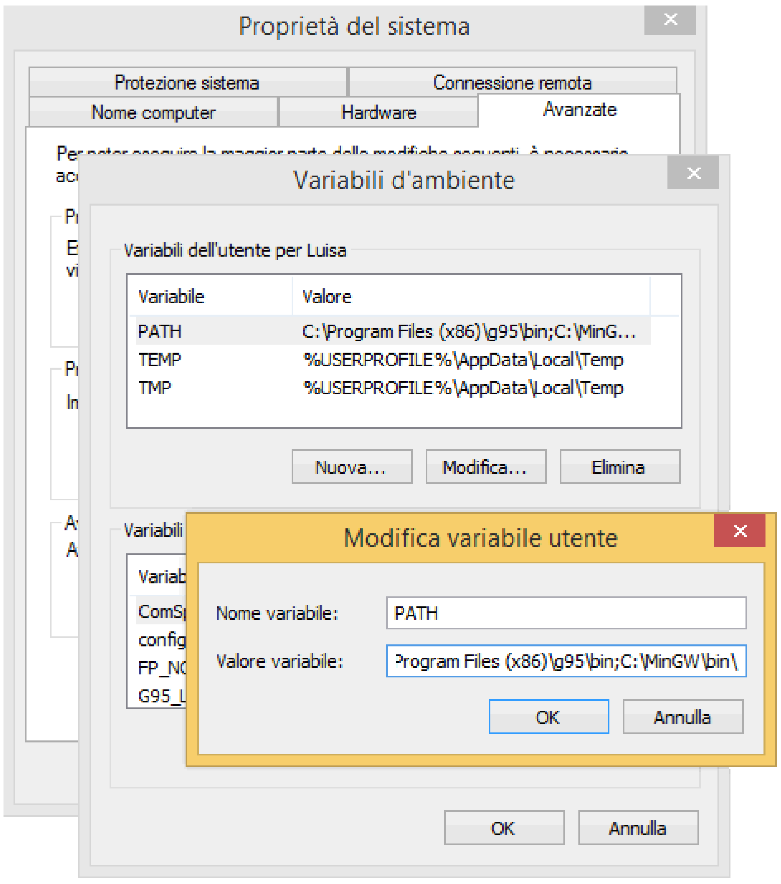


```
Microsoft Windows [Versione 6.3.9600]
(c) 2013 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\Luisa>gfortran -dumpversion
4.8.1

C:\Users\Luisa>
```

Figure 4: Checking gfortran version.



Variabili dell'utente per Luisa

Variabile	Valore
PATH	C:\Program Files (x86)\g95\bin;C:\MinG...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

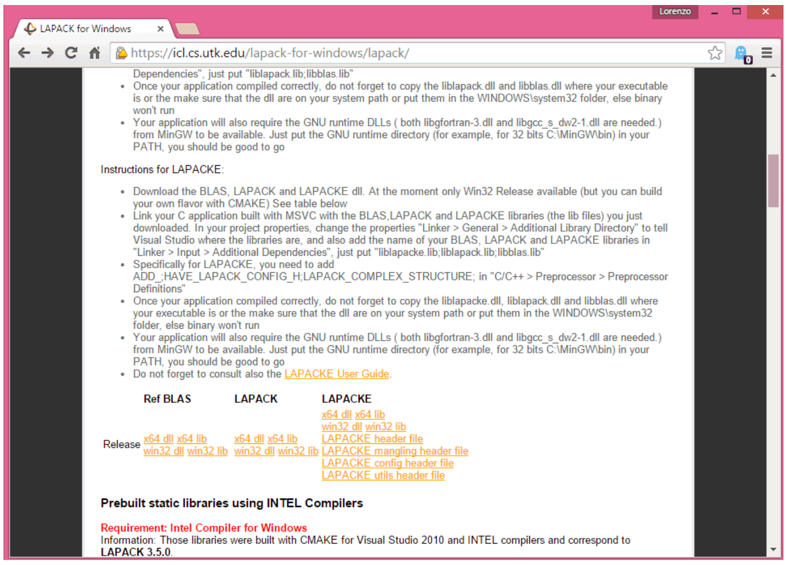
Nome variabile: PATH

Valore variabile: ?rogram Files (x86)\g95\bin;C:\MinGW\bin\

Figure 5: Setting the PATH enviroment variable.

versions it works similarly and guides can found on the Internet[13]. Press at the same time the Windows key and the 'Pause/Break' key to bring up the 'System' window (alternatively open the Control Panel and then click on 'System and Security' and the 'System'); in the left column click on 'Advanced System Settings' and then in the System Properties window click the Environment Variables button near the bottom of the window. Select 'PATH' in the upper part of the window and then click on 'Modify' (see fig. 5). In the second line add at the end of whatever is present a semicolon (NB: not a colon as one would in Linux!) and the directory where gfortran resides, `c:\MinGW\bin` in our case. Open a new Command Prompt window and try again to type `gfortran -dumpversion`, it should now work.

[13]For example, see http://www.computerhope.com/issues/ch000549.htm



Dependencies", just put "liblapack.lib;libblas.lib"

- Once your application compiled correctly, do not forget to copy the liblapack.dll and libblas.dll where your executable is or the make sure that the dll are on your system path or put them in the WINDOWS\system32 folder, else binary won't run
- Your application will also require the GNU runtime DLLs (both libgfortran-3.dll and libgcc_s_dw2-1.dll are needed.) from MinGW to be available. Just put the GNU runtime directory (for example, for 32 bits C:\MinGW\bin) in your PATH, you should be good to go

Instructions for LAPACKE:

- Download the BLAS, LAPACK and LAPACKE dll. At the moment only Win32 Release available (but you can build your own flavor with CMAKE) See table below
- Link your C application built with MSVC with the BLAS,LAPACK and LAPACKE libraries (the lib files) you just downloaded. In your project properties, change the properties "Linker > General > Additional Library Directory" to tell Visual Studio where the libraries are, and also add the name of your BLAS, LAPACK and LAPACKE libraries in "Linker > Input > Additional Dependencies", just put "liblapacke.lib;liblapack.lib;libblas.lib"
- Specifically for LAPACKE, you need to add ADD_;HAVE_LAPACK_CONFIG_H;LAPACK_COMPLEX_STRUCTURE; in "C/C++ > Preprocessor > Preprocessor Definitions"
- Once your application compiled correctly, do not forget to copy the liblapacke.dll, liblapack.dll and libblas.dll where your executable is or the make sure that the dll are on your system path or put them in the WINDOWS\system32 folder, else binary won't run
- Your application will also require the GNU runtime DLLs (both libgfortran-3.dll and libgcc_s_dw2-1.dll are needed.) from MinGW to be available. Just put the GNU runtime directory (for example, for 32 bits C:\MinGW\bin) in your PATH, you should be good to go
- Do not forget to consult also the LAPACKE User Guide.

	Ref BLAS	LAPACK	LAPACKE
Release	x64 dll x64 lib win32 dll win32 lib	x64 dll x64 lib win32 dll win32 lib	x64 dll x64 lib win32 dll win32 lib LAPACKE header file LAPACKE mangling header file LAPACKE config header file LAPACKE utils header file

Prebuilt static libraries using INTEL Compilers

Requirement: Intel Compiler for Windows
Information: Those libraries were built with CMAKE for Visual Studio 2010 and INTEL compilers and correspond to **LAPACK 3.5.0**.

Figure 6: Downloading pre-compiled BLAS/LAPACK libraries for Windows.

7.2.2 Installing the BLAS and LAPACK libraries

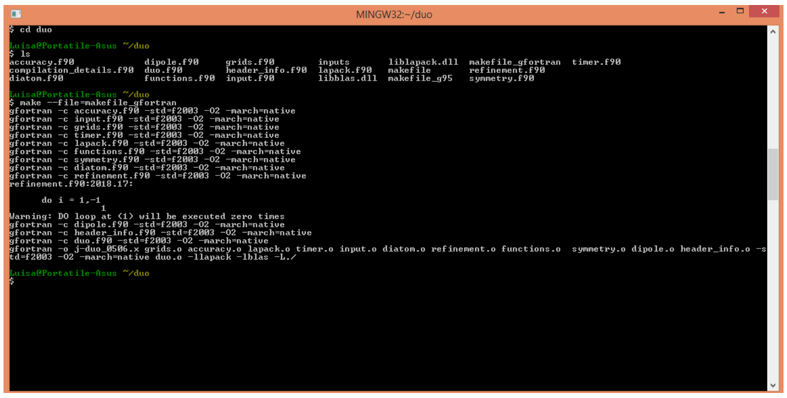
The easiest way to use BLAS/LAPACK under Windows is to use pre-build dlls, which at the time of writing can be downloaded from the Internet[14]. In the section 'Prebuilt dynamic libraries using Mingw' download two files: the reference BLAS win32 dll file `libblas.dll` and the LAPACK win32 dll file `liblapack.dll`, see fig. 6. Be sure to download the 32 bit versions (not the ones under the link 'x64 dll'). The easiest way to use these two files is to copy them in the same directory where the Duo source files are; they will also be needed to run Duo, and should be copied along with the Duo executable.

7.2.3 Compiling Duo

After having installed the Fortran compiler and the libraries we are now almost ready to compile Duo; however, we need the `make` command and for this reason we need to open a MinGW terminal window instead of the standard Command Prompt.

Open File Explorer (in versions of Windows prior to Windows 8 it was known as Windows Explorer; it can be opened by pressing simultaneously the windows key and the 'E' key), navigate to c:\MinGW\msys\1.0\ and double click on the file `msys.bat`. This will open a MinGW terminal window. Unpack the Duo package in c:\MinGW\msys\1.0\home\ ; this will create several directories, including one called `duo`. Go to the `duo` directory by typing `cd duo` and copy

[14]https://icl.cs.utk.edu/lapack-for-windows/lapack/



```
$ cd duo

Luisa@Portatile-Asus ~/duo
$ ls
accuracy.f90              dipole.f90     grids.f90        inputs       liblapack.dll  makefile_gfortran  timer.f90
compilation_details.f90  duo.f90        header_info.f90  lapack.f90   makefile       refinement.f90
diatom.f90               functions.f90  input.f90        libblas.dll  makefile_g95   symmetry.f90

Luisa@Portatile-Asus ~/duo
$ make --file=makefile_gfortran
gfortran -c accuracy.f90 -std=f2003 -O2 -march=native
gfortran -c input.f90 -std=f2003 -O2 -march=native
gfortran -c grids.f90 -std=f2003 -O2 -march=native
gfortran -c timer.f90 -std=f2003 -O2 -march=native
gfortran -c lapack.f90 -std=f2003 -O2 -march=native
gfortran -c functions.f90 -std=f2003 -O2 -march=native
gfortran -c symmetry.f90 -std=f2003 -O2 -march=native
gfortran -c diatom.f90 -std=f2003 -O2 -march=native
gfortran -c refinement.f90 -std=f2003 -O2 -march=native
refinement.f90:2018.17:

      do i = 1,-1
                1
Warning: DO loop at (1) will be executed zero times
gfortran -c dipole.f90 -std=f2003 -O2 -march=native
gfortran -c header_info.f90 -std=f2003 -O2 -march=native
gfortran -c duo.f90 -std=f2003 -O2 -march=native
gfortran -o j-duo_0506.x grids.o accuracy.o lapack.o timer.o input.o diatom.o refinement.o functions.o  symmetry.o dipole.o header_info.o -s
td=f2003 -O2 -march=native duo.o -llapack -lblas -L./

Luisa@Portatile-Asus ~/duo
$
```

Figure 7: Compilation output with gfortran.

in this directory (e.g., using File Explorer) the dll files download previously. We are now finally ready for the actual compilation. Duo is provided with separate makefiles for the compilers it has been tested with. To compile Duo using gfortran type
`make --file=makefile_gfortran` You should get an output similar to the one shown in fig.7. Compilation should only take a couple of minute or so. At the end you should have a Duo executable file you can run (either from the MinGW terminal window or from the standard Command Prompt; just remember to copy the BLAS / LAPACK dll along with the executable.)

7.2.4 Installing the g95 compiler

Instead of gfortran you can also use the g95 compiler. You may want to do this if, for example, you could not get gfortran to work correctly. To install g95 under Windows go to the page[15] and download the file `g95-MinGW-41.exe`. At the time of writing the ftp server pointed to by the links in the page above does not work; however, there are many places on the Internet were the file above can be found[16]. The latest version is version v. 0.94 dated 18 May 2013.

Run it and install g95 in a directory of your choice. Accept to ”Install MinGW Utilities and libs?” and also accept the suggestions of setting up the PATH for all users.

Once this is done g95 should be automatically available at the command line. Open a new command prompt window and type g95 -dumpversion. You should get a short message whose first line is
`G95 (GCC 4.1.2 (g95 0.94)` May 18 2013)!

[15]http://www.g95.org/downloads.shtml#Win
[16]for example, we downloaded it from http://tcc.customer.sentex.ca/g95/4.1/g95-MinGW-41.exe

Once the g95 compiler is installed, install MinGW and download the BLAS/LAPACK dlls exactly as described in the sections above. At the point of compilation type

```
make --file=makefile_g95
```

8 Duo-script

The cshell script `do-duo-input.csh` can be used to replace the varied parameters (marked with `Fit`) in the Duo-input by the corresponding refined values from the Duo-output

```
do-duo-input.csh duo_1.out > duo_2.inp
```

9 Programmer's guide to Duo

In this section we describe the structure of the source code. This can be useful to people who plan to modify Duo, e.g. to implement new features.

At present Duo's source code consists in 13 files for a total of 18 951 lines of code in total (including comments). The three files containing the bulk of the program are:

1. `diatom.f90` (7 506 lines) contains the code relative to the calculation of energy levels and wave functions and is the core portion of Duo.
2. `dipole.f90` (1 790 lines) contains the code relative to the calculation of line intensities and related quantities.
3. `refinement.f90` (2 095 lines) contains the part of Duo dealing with fitting.

These other files contains various other parts of the codes:

4. `duo.f90` (99 lines) is a small front-end which calls the subroutines for energy, line intensities or fitting calculations.
5. `accuracy.f90` (88 lines) contains values of physical constants, variable kind parameters and other data to be shared.
6. `atomic_and_nuclear_data.f90` (2 788 lines) contains the internal database of atomic and nuclear properties, and various functions and subroutines to access the data.
7. `functions.f90` (809 lines) contains the definitions of the analytical function forms used in Duo.

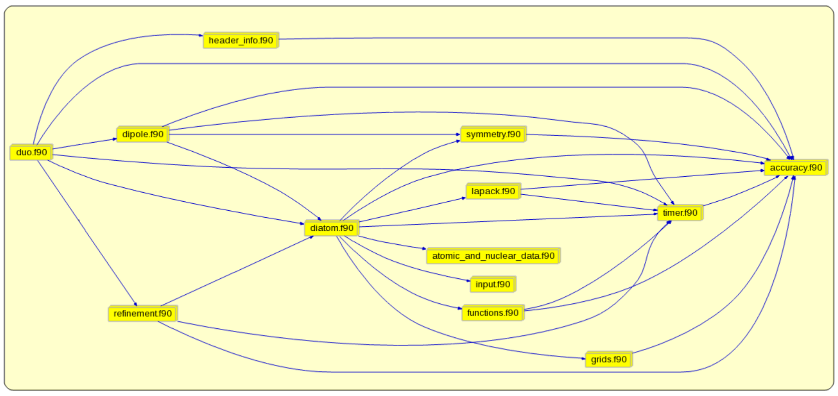


Figure 8: This diagrams show how the various Duo source files refer to one another.

8. `grids.f90` (513 lines) contains the code used for the calculation of the grid points (for uniform and non-uniform grids) as well as the routines used for spline interpolation.

9. `symmetry.f90` (330 lines) contains subroutines dealing with permutation symmetry of the molecule, calculation of degeneracy factors etc.

10. `header_info.f90` (30 lines) contains a simple subroutine which prints the initial Duo logo and information such as version number.

11. `timer.f90` (800 lines) contains utility subroutines which monitor the time spent on various parts of the code as well as memory usage.

12. `input.f90` (1 020 lines) is an input parsing module written by Anthony J. Stone, see [Sto05]. Is is used by Duo to read the input file.

13. `lapack.f90` (1 083 lines) is set of explicit interfaces to lapack routines. It is not the lapack library itself!

Here below we give some comments on the structure of the code.

The source file `diatom.f90` contains a module called `diatom_module` which provides the bulk of subroutines necessary for computing wave functions and energy levels. In lines 13 to 404 we define thresholds, variables and derived data types (classes) used throughout the module. In particular, the derived data type `fieldT` is used for all for all curves used in Duo (potential energy and dipole curves as well as coupling curves). For example, if `my_curve` is an object of type `fieldT` then `my_curve%class` returns a text string which is `poten` for potential energy curves, `spinorbit` for a spin-orbit coupling curves etc. A Fortran 2003 feature in the data type `fieldT` is the use of abstract interfaces and of procedure pointers. This is done in line nr. 187

```
procedure (analytical_fieldT),pointer, nopass ::  analytical_field => null()
```

This line defines the procedure point `analytical_field` (initially nullified and with the 'nopass' attribute) which has the same interface as the procedure `analytical_fieldT`; in turn, `analytical_fieldT` is an abstract interface (i.e., a template) for all curves and is defined in the file `functions.f90`.

From line 407 to 3361 there is the subroutine `ReadInput`, which deals with reading the inputs file, perfoming initial consistency checks etc. From lines 3367 to 3612 there are the subroutines `define_quanta_bookkeeping` and `check_and_set_atomic_data`; the first defined the quantum numbers used for the calculations and allocates some relevant arrays while the second reads from the internal database values for the atomic masses.

From lines 3614 to 5074 the subroutine `map_fields_onto_grid` evaluates all the curves at the grid points, performing interpolation and extrapolation if necessary and prints the sampled curves. For potential energy curves it will also find the minimum and compute and print quantities such as the value of the derivative at the minimum, the harmonic vibrational frequency etc. The subroutines also perfoms a variety of checks.

From lines 5078 to 7321 the subroutine `duo_j0` builds and diagonalizes the $J = 0$ vibrational matrices for each electronic state.

Finally, from lines 7324 to 7503 there are a few service subroutine whose purpuse should be clear from the source code.

References

[AKW+12] G. Audi, F. G. Kondev, M. Wang, B. Pfeieffer, X. Sun, J. Blachot, and M. MacCormick. The nubase2012 evaluation of nuclear properties. *Chin. Phys. C*, 36:1157–1286, 2012.

[Le 07] R. J. Le Roy. *LEVEL 8.0 A Computer Program for Solving the Radial Schrödinger Equation for Bound and Quasibound Levels.* University of Waterloo Chemical Physics Research Report CP-663, `http://leroy.uwaterloo.ca/programs/`, 2007.

[LH07] Robert J. Le Roy and R. D. E. Henderson. A new potential function form incorporating extended long-range behaviour: application to ground-state ca$_2$. *Mol. Phys.*, 105:663–677, 2007.

[LHJ06] Robert J. Le Roy, Yiye Huang, and Calvin Jary. An accurate analytic potential function for ground-state n$_2$ from a direct-potential-fit analysis of spectroscopic data. *J. Chem. Phys.*, 125:164310, 2006.

[LHTL11] R. J. Le Roy, C. C. Haugen, J. Tao, and H. Li. Long-range damping functions improve the short-range behaviour of mlr potential energy functions mlrd-pec. *Mol. Phys.*, 109:435–446, 2011.

[MS74] J. N. Murrell and K. S. Sorbie. New analytic form for potential-energy curves of stable diatomic states. *J. Chem. Soc. Faraday Trans. II*, 70(9):1552–1556, 1974.

[MSLR08] Vladimir V. Meshkov, Andrey V. Stolyarov, and Robert J. Le Roy. Adaptive analytical mapping procedure for efficiently solving the radial schrodinger equation. *Phys. Rev. A*, 78:052510, 2008.

[Sto05] A. J. Stone. input.f90 — a fortran90 module for parsing text input, 2005. See http://www-stone.ch.cam.ac.uk/programs/.

[THY13] J. Tennyson, C. Hill, and S. N. Yurchenko. Data structures for ExoMol: Molecular line lists for exoplanet and other atmospheres. In *6th international conference on atomic and molecular data and their applications ICAMDATA-2012*, volume 1545 of *AIP Conference Proceedings*, pages 186–195. AIP, New York, 2013.

[Var09] A.J.C. Varandas. A simple, yet reliable, direct diabatization scheme. the states of. *Chemical Physics Letters*, 471(4–6):315 – 321, 2009.

[VD97] L. Visscher and K.G. Dyall. Dirac-fock atomic electronic structure calculations using different nuclear charge distributions. *Atom. Data Nucl. Data Tables*, 67:207–224, 1997.

[vRB84] A. A. Šurkus, R. J. Rakauskas, and A. B. Bolotin. The generalized potential-energy function for diatomic-molecules. *Chem. Phys. Lett.*, 105:291–294, 1984.

[Wat03] J. K. G. Watson. Robust weighting in least-square fits. *J. Mol. Spectrosc.*, 219:326–328, 2003.

[WAW$^+$12] M. Wang, G. Audi, A.H. Wapstra, F.G. Kondev, M. MacCormick, X. Xu, and B. Pfeiffer. The ame2012 atomic mass evaluation (ii). tables, graphs and references. *Chin. Phys. C*, 36:1603–2014, 2012.

[YLTS15] S. N. Yurchenko, L. Lodi, J. Tennyson, and A. V. Stolyarov. Duo: a general program for the calculation of spectra of diatomic molecules. *Comput. Phys. Commun.*, (to be submitted), 2015.