ECE 3620
Programming Assignment # 1

Numerical Solution of Differential Equations: The Zero-Input Solution

Due: Oct 10, 2014

Objective and Background

This assignment has two basic objectives:

1. To remind you that you know how to program (or that you *should* know!)
2. To give you a little practice solving differential equations numerically

In this assignment you will write a program in C or C++ to solve first-order differential equations of the form

$$(D-a)y_0(t) = 0, \tag{1}$$

subject to some initial condition on $y_0(t)$. One way (not the best, but probably the easiest) is to make the following approximation:

$$Dy_0(t) \approx \frac{y_0(t+\Delta t) - y_0(t)}{\Delta t}. \tag{2}$$

Observe that in the limit as $\Delta t \to 0$, this approximation becomes exact. Substituting (2) into (1) we obtain

$$\frac{y_0(t+\Delta t) - y_0(t)}{\Delta t} - ay_0(t) = 0$$

Solving for $y_0(t+\Delta t)$,

$$y_0(t+\Delta t) = (1+a\Delta t)y_0(t) \tag{3}$$

If we know $y_0(t)$ at time $t=0$, say $y_0(0) = y_0$ and we choose some Δt, then we can find the (approximate) value at later times by iterating (3):

$$\begin{aligned}
y_0(\Delta t) &= (1+a\Delta t)y_0(0) \\
y_0(2\Delta t) &= (1+a\Delta t)y_0(\Delta t) \\
y_0(3\Delta t) &= (1+a\Delta t)y_0(2\Delta t) \\
y_0(4\Delta t) &= (1+a\Delta t)y_0(3\Delta t)
\end{aligned}$$

and so forth. This immediately suggests a `for` loop in a programming language:

```
y = y0;
for(i = 0; i < N; i++) {
  y = (1+a*deltat)*y;
}
```

(You must, of course, set all the variables correctly).

What works for one first-order differential equation works for systems of first-order equations (i.e., differential equations in state-variable form). Suppose we have a 3rd-order system given by

$$(D^3 + a_2D^2 + a_1D + a_0)y_0(t) = 0.$$

We will assign the derivitives of $y(t)$ in the following way:

$$\begin{aligned}
x_1(t) &= y_0(t) \\
x_2(t) &= \dot{y}_0(t) \\
x_3(t) &= \ddot{y}_0(t).
\end{aligned}$$

With this definition, it should be obvious that

$$\begin{aligned} x_2(t) &= \dot{x}_1(t) \quad (4) \\ x_3(t) &= \dot{x}_2(t) \quad (5) \end{aligned}$$

and the entire differential equation can be written as:

$$\dot{x}_3(t) + a_2 x_3(t) + a_1 x_2(t) + a_0 x_1(t) = 0. \quad (6)$$

Let

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$$

and let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

Then a system of first-order differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \quad (7)$$

may be written as

$$(D - \mathbf{A})\mathbf{x}(t) = \mathbf{0}$$

where the vector derivative is defined as

$$D\mathbf{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix}.$$

Then following the same steps as before, it is possible to approximate the next step of the solution:

$$\mathbf{x}(t + \Delta t) = (\mathbf{I} + \mathbf{A}\Delta t)\mathbf{x}(t). \quad (8)$$

where $\mathbf{I}$ is the identity matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

How do we form the matrix $\mathbf{A}$? Substituting (4), (5), and (6) into (7), we find that $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix}.$$

This structure can be extended to any order differential equation.

Assignment

1. For the differential equation

$$(D + 2.5)y_0(t) = 0$$

with initial condition $y_0(0) = 3$:

   (a) Find and plot the analytical (exact) solution to the differential equation for $0 \le t \le 10$.

   (b) Write a program in C(++) to plot a numerical solution using (3). You may have to try several values of Δt to get a good enough approximation.

(c) Compare the exact solution with the approximate solution.

2. For the third-order differential equation

$$(D^3 + 0.6D^2 + 25.1125D + 2.5063)y_0(t) = 0$$

with initial conditions $y_0(0) = 1.5$, $\dot{y}_0(0) = 2$, $\ddot{y}(0) = -1$:

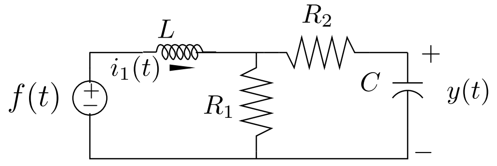
   (a) Find and plot the analytical solution to the differential equation for $0 \leq t \leq 10$. Identify the roots of the characteristic equation and plot them in the complex plane.

   (b) Put the third-order differential equation into state-space form.

   (c) Write a program in C(++) to plot an approximate solution using (8). You may have to try several values of Δt to get a reasonable approximation.

   (d) Compare the exact solution with the approximate solution.

3. For the circuit shown here:



where $R_1 = 1\ k\Omega$, $R_2 = 22\ k\Omega$, $C = 10\ \mu F$, and $L = 5\ H$.

   (a) Determine a differential equation relating the input $f(t)$ to the output $y(t)$.

   (b) Determine the initial conditions on $y(t)$ if $i_1(0) = 0.2$ A and $y(0) = 5$ V.

   (c) Determine the analytical solution for the zero-input response of the system with these initial conditions.

   (d) Represent the differential equation for the circuit in state variable form.

   (e) Using your program, determine a numerical solution to the differential equation for the zero-input response.

   (f) Plot and compare the analytical and the numerical solution. Comment on your results.

   (g) Suppose that the circuit had nonlinear element in it, such as dependent sources. Describe how the analytical solution and numerical solution would change.

Turn in your programs, your plots, and your observations.

Hints and helps

Making plots To make plots, I recommend you write out an ascii file with the t and y values that you want. Then use Matlab to make the plots.

To write the files using C++, at the top of your program you need to open a file to write into:

```
#include <fstream>   /* appears near the beginning of your program */
   .
   .
   .

main()
{
   .
   .
   .
   ofstream outfile("junk"); /* open a filed called 'junk' for writing */
   .
   .
   .
```

Then any time you want to write, you simply use the ofstream operators:

```
   outfile << t << " " << y << endl;
```

At the end of the program you should close the file:

```
   outfile.close();
```

The steps above will save the computed data into a file named "`junk`". Then you can use the plot facilities of Matlab to make the plot. This is done by reading the data into Matlab, then plotting the data. The following commands are to be done inside Matlab.

```
load junk     % load the file 'junk' into a variable called 'junk'
plot(junk(:,1),junk(:,2))   % plot the first column vs. the second
                            % column
```

Matrix/vector operations The following are some simple, fixed size matrix and vector routines that might be helpful.

```
void mattimes(double t, double a[][3],double m[][3])
{
   int i,j;
   for(i = 0; i < 3; i++)
      for(j = 0; j < 3; j++)
         m[i][j] = t*a[i][j];
}

void matsub(double a[][3],double b[][3],double c[][3])
{
   int i,j;
   for(i = 0; i < 3; i++)
      for(j = 0; j < 3; j++)
         c[i][j] = a[i][j]-b[i][j];
}

void matvecmult(double m[][3], double *v, double *prod)
{
   double sum;
   int i,j;
   for(i = 0; i < 3; i++) {
      sum = 0;
      for(j = 0; j < 3; j++) {
         sum += m[i][j]*v[j];
      }
      prod[i] = sum;
   }
}
```

And here is a suggested way to use some of these (but check the signs carefully!):

```
double a[3][3] = {{0,-1,0},{0,0,-1},{20.002,100.05,0.4}};
double m[3][3];
double I[3][3] = {{1,0,0},{0,1,0},{0,0,1}};

...

mattimes(deltat,a,m);
matsub(I,a,m);
```

Miscellaneous hints Remember that the starting index for arrays in C/C++ is 0.

An example loop for the time variable:

```
for(t = 0; t <= 4; t += deltat) {
```