Short Tutorial on Matlab

Part 2. Ordinary Differential Equations

1. Suppose we want to simulate the following set of differential equations:

$$\frac{d^2}{dt^2}y + 3\cdot\left(\frac{d}{dt}y\right) + 2\cdot y = 4\cdot\exp(-2\cdot t) - 5$$

subject to the following initial conditions,

$$y(0) = 2$$

$$\frac{d}{dt}y(0) = -1$$

2. You need to convert to state space form. Let $x_1 = y$ and $x_2 = dy/dt$, then we have

$$\frac{d}{dt}x_1 = x_2$$

$$\frac{d^2}{dt^2}x_2 = -3\cdot x_2 - 2\cdot x_1 + 4\cdot\exp(-2\cdot t) - 5$$

$$x_1(0) = 2$$

$$x_2(0) = -1$$

3. Next, create an m-file using either Matlab's editor or any text editor, e.g. "notepad":

```
function dx = tutorialEqn1(t,x)

    % x is the state vector
    % to minimize parentheses you could put them
    % in other variables

    x1=x(1);
    x2=x(2);

    % write the state equations

    dx1 = x2;
    dx2 = -3*x2 -2*x1 +4*exp(-2*t) - 5;

    % collect the derivatives into a column vector

    dx = [dx1;dx2];
```

then save as an m-file, e.g. **`tutorialEqn1.m`**

4. In matlab, you can now invoke the ode solvers. For example, you can use **ode45** command:

```
   >> [t,x]=ode45(@tutorialEqn1,[0 10],[2;-1])
```

   Remarks:
   a) Use the **'@'** symbol followed by the filename (without the file extension)
   b) **[0 10]** is the range of time values
   c) **[2;-1]** is the initial condition
   d) **[t,x]** is the solution output. **t** stores the time values while **x** stores the solution where column 1 is x(1), etc.

5. You can now plot the solutions. For instance,

```
>> plot(t,x(:,1))
```

   will plot the first column of x.

6. **Passing of parameters:** you can also pass parameters (either scalar of matrix). For instance, suppose you want to simulate the matrix equation: dx/dt = Ax. The you can use the general function:

```
function    dx = lindiff(t,x,A)

    dx = A*x;
```

   Suppose, we have defined matrix A to be

```
>> A = [-3 4 0;0 -1 2;3 3 -6];
```

   with intial condition vector

```
>> x0 =[ -1 ; 2 ;0.5 ];
```

   then use the following command:

```
>> [t,x]=ode45(@lindiff,[0 100],x0,[],A);
```

   Note: the **'[]'** between **x0** and **A** is required as a placeholder for options (see below, item 8).

7. **Evaluating solutions at specified points**

   Scenario: since **ode45** may not have fixed integration points, you may need to interpolate. Another alternative is to use the command **deval** . However, this requires that the solution output is a structure.

Example: (assuming file **`lindiff.m`** given in item 6 above already exists)

```
>> A=[0 1;-2 -2];
>> testSoln = ode45(@lindiff,[0 10],[0 -1],[],A);
>> new_t=linspace(0,10,101);
>> new_y=deval(testSoln,new_t);
```

This will result in an output that is uniformly incremented in $\Delta t=0.1$

8. Changing Options.

This requires creating a structure object conforming to ODE45. To create, use odeset. For a list of available fields,

```
>> testOptions=odeset
```

This should list all the fields with empty (defaulted) values.

So, for example if we want to change the relative tolerance to 1e-5, and absolute tolerance to [1e-4;1e-2] we could use

```
>> testOptions=odeset('RelTol',1e-5,'AbsTol',...
   [1e-4;1e-2])
```

To add more changes,

```
>> testOptions=odeset(testOptions,'Refine',10)
```

After all changes have been included, run the simulation using the created options structure,

```
>> [t,y] = ode45(@lindiff,[0 5],[0 -1],testOptions,A);
```