**Generating Odefiles for Matlab's ODE Solvers**

**Jonathan duSaint**

**College of the Redwoods**

In my Ordinary Differential Equations class, I was presented with a wide variety of methods for solving DEs. One frequently employed method was numerical analysis with the ODE Suite in MATLAB. The ODE Suite is set up such that a differential equation must be converted to an ode file before running a solver on it. This process requires a bit of programming skill, but is relatively straightforward.

**First Order**

Consider the following first order initial value problem:

*dy/dt=y2-2y+2, y(0)=0*

Before using one of Matlab's solvers, you first need to create what is known as an ODE m-file, as follows:

function Yprime=f1(t,Y)

Yprime=Y.^2-2\*Y+2;

This is an ODE m-file at its most simple. To use, it must first be stored as f1.m (the function name and the file name should always be the same) in a directory on the MATLAB path. With the file created, a solver from the ODE Suite may be selected. In the following line, ode45 (a combination of fourth and fifth degree variable step Runge-Kutta algorithms) is used.

[t,Y]=ode45('f1',[0, 2.25],0)

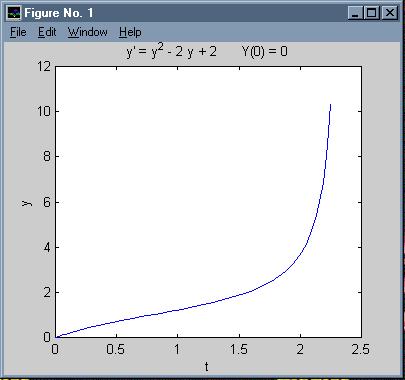
The syntax above is used to solve the IVP *dy/dt=y2-2y+2, y(0)=0* over the time interval [0, 2.25]. The output is stored as vectors in the variables t and Y. The following command should produce an image similar to that in Figure 1.

plot(t,Y)

xlabel('t')

ylabel('y')

title('y'' = y^2 - 2 y + 2 y(0) = 0')



**Figure 1**

**Second Order**

Due to the fact that MATLAB is matrix oriented, solving higher order equations is just as simple as the above first order equation. Consider the following second order IVP:

*y''+2y=sin(1.2t), y(0)=1, y'(0)=0*

This equation must first be broken into a system of first order DEs using a technique taught in standard differential equations classes. First introduce the following variables : *y1=y, y2=y'.* Then, *y1'=y'=y2,* and *y2'=y''=-2y+sin(1.2t)=-2y1+sin(1.2t).* This produces the following system of first order differential equations.

*y1' = y2*

*y2' = -2y1+sin(1.2t)*

Or, in vector notation, *(y1',y2') = (y2, -2y1+ sin(1.2t))*. Note that this vector equation takes the form *Y' = f(t,Y)*, where *Y = (y1,y2)* and *f(t,Y) = (y2, -2y1+ sin(1.2t))*. In addition, note that the initial condition, *y(0) = 1, y'(0) = 0*, when written as a vector, takes the form *Y(0) = (y1(0), y2(0)) = (y(0), y'(0)) = (1, 0)*. Vector form makes it easier to visualize the next step, which is a conversion to an ODE m-file.

function Yprime=f2(t,Y)

Yprime=[Y(2); -2\*Y(1) + sin(1.2\*t)];

The command

[t,Y]=ode45('f2',[0, 20],[1; 0])

produces vectors, t and Y. The vector Y has two columns. The first column of Y contains solutions of *y*1 (or *y*), the second column contains solutions of *y2* (or *y'*), evaluated at the times in corresponding positions in vector t. The data in t and Y can be manipulated in a number of ways. For example, the following code should produce an image similar to Figure 2.

subplot(221)

plot(t,Y(:,1))

title('y1 vs. t')

subplot(222)

plot(t,Y(:,2))

title('y2 vs. t')

subplot(223)

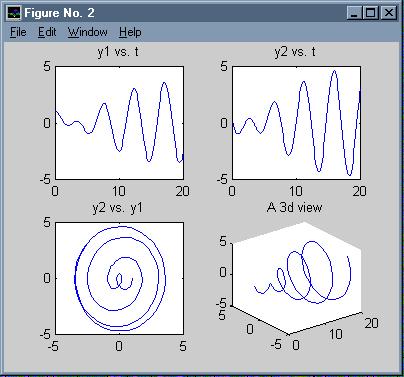
plot(Y(:,1),Y(:,2))

title('y2 vs. y1')

subplot(224)

plot3(t,Y(:,1),Y(:,2))

title('A 3d view')



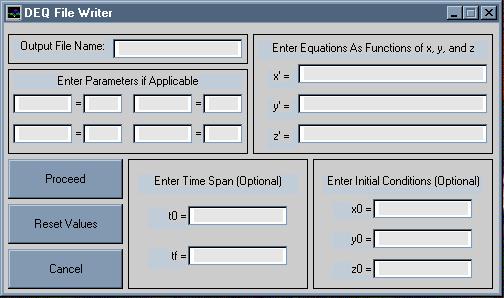
**Figure 2**

**Defwrite**

While the ODE m-files shown in the previous examples are fairly simple, they can get complex rather quickly, requiring considerably more programming skill. I have written a graphic user interface (GUI) to simplify and expedite the solving of initial value problems. Typing

defwrite

at the MATLAB prompt will bring up the GUI shown in Figure 3.



**Figure 3**

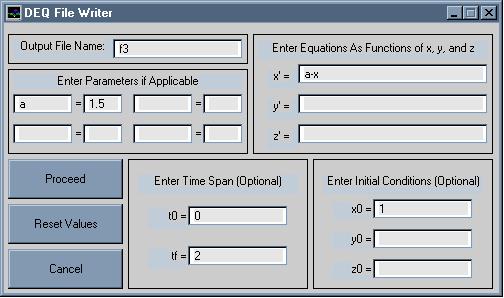
While the boxes are somewhat self explanatory, a few examples are in order.

**A First Order Example**

Consider the following IVP.

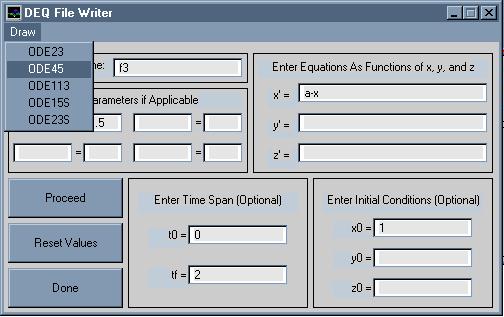
*x' = a-x, x(0) = 1*

Note the introduction of a parameter, *a*. Letting the parameter *a* = 1.5, let's find the solution of the IVP over the time interval [0, 2]. After typing defwrite at the MATLAB prompt, fill in the boxes as shown in Figure 4.



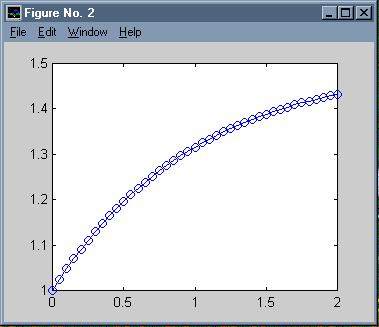
**Figure 4**

Note that I have chosen to name the output file f3, but any name would suffice (call it jon, for example). Clicking on the the proceed button creates the draw menu, as shown in Figure 5.



**Figure 5**

Selecting ode45 from the menu automatically produces the graph shown in Figure 6.



**Figure 6**

**A Second Order Example**

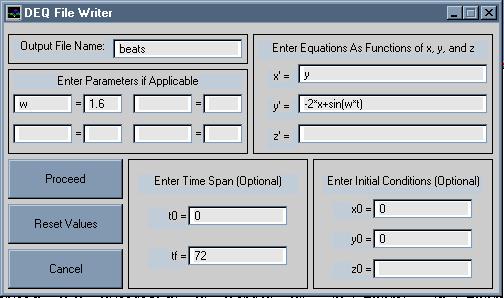
Consider the second order equation

*y'' + 2y = sin(wt), y(0) = 0, y'(0) = 0*

Introducing new variables *x = y, y = y'* leads to *x' = y' = y* and *y' = y'' = -2y + sin(wt) = -2x + sin(wt)* and initial conditions *x(0) = y(0) = 0,* and *y(0) = y'(0) = 0.* Thus, we have created the system

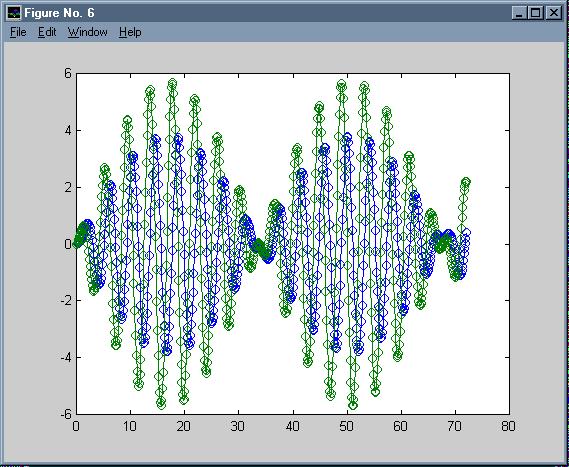
*x' = y  
y' = -2x + sin(wt)*

with initial conditions *x(0) = 0, y(0) = 0.* The entries in Figure 7 should now be clear.



**Figure 7**

Clicking on proceed and selecting ode45 from the Draw menu produces an image similar to that in Figure 8.



**Figure 8**

Note that the solutions for both x and y are drawn in Figure 8.

**Examining the Output of Defwrite**

The actual output of defwrite is an ODE m-file. For example, entering

type beats

at the MATLAB prompt will allow you to examine the ODE m-file created in the last example.

function [out1, out2, out3]=beats(t,Y,flag,w)

%This ODE file was created by defwrite.m

if nargin < 4 | isempty(w)

w = 1.600000e+000;

end

if nargin < 3 | isempty(flag)

out1=[Y(2);-2.\*Y(1)+sin(w.\*t)];

elseif strcmp(flag, 'init')

out1=[ 0.00000000 ; 72.00000000 ];

out2=[ 0.00000000 0.00000000]';

out3=[];

end

This code is significantly more complicated than what was shown in the first few examples. The value of the parameter is set in the line w = 1.600000e+000. The time interval, [0, 72], is set in the line out1=[ 0.00000000 ; 72.00000000 ], and the initial conditions are set in the line out2=[ 0.00000000 0.00000000]. For further information on writing ODE m-files, refer to the MATLAB documentation, or type "help odefile" at the MATLAB prompt.

Just because the values are set in the file does not mean that different values may not be specified. The following line of code calls ode45 with a time interval of [0 144], default initial conditions of *Y(0) = (1, -1)*, and sets the parameter to 1.414.

[t,Y] = ode45('beats',[0 144],[1; -1],flag,1.414)

In the call to ode45, "flag" is just a placeholder for flags that actually get passed by ode45 to the ODE m-file.

The ODE m-file beats.m can now be used in calls to the solvers in MATLAB's ODE Suite. However, because the parameter, time span, and initial conditions have all been set in the file beats.m, the calling syntax is greatly simplified. One need only enter:

[t,Y]=ode45('beats');

The following commands use the data in vectors t and Y to produce an image similar to that in Figure 9.

subplot(221)

plot(t,Y(:,1))

title('y1 vs. t')

subplot(222)

plot(t,Y(:,2))

title('y2 vs. t')

subplot(223)

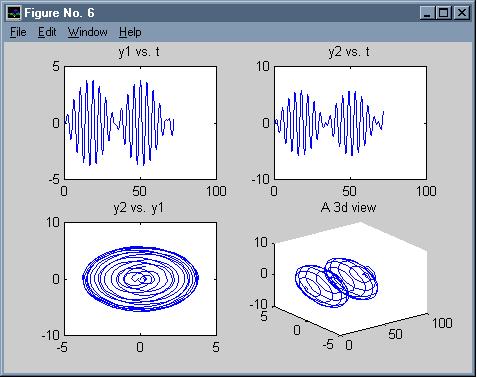
plot(Y(:,1),Y(:,2))

title('y2 vs. y1')

subplot(224)

plot3(t,Y(:,1),Y(:,2))

title('A 3d view')



**Figure 9**

**Conclusion**

defwrite will run on any platform, provided MATLAB version 5.0 or higher is available. To download a copy of the program defwrite.m, click [here](http://online.redwoods.cc.ca.us/instruct/darnold/mfiles/defwrite.m)