

# Numerical Methods

David Arnold  
College of the Redwoods

February 24, 1998

## Abstract

This activity introduces students to solution technique available for the numerical solution of first order ordinary initial value problems. In particular, euler's method, rk2, and rk4 are compared and contrasted. Because the approach is completely visual, students will need little, if any, prerequisite material before attempting this activity. **Prerequisites.** Some familiarity with Matlab's `plot` command. Also, you will need to download `eul.m`, `rk2.m`, and `rk4.m`. The activity also uses Version 2 of the Symbolic Toolbox, a Matlab interface to Maple. However, if you do not have the Symbolic Toolbox, you can find exact solutions of the IVPs by hand, then use Matlab to draw their graphs.

## 1 First Order Ordinary Differential Equations

A first order ordinary differential equation (ODE) has the following form:

$$\frac{dy}{dt} = f(t, y) \tag{1}$$

For example, each of the following is a first order ODE:

$$\frac{dy}{dt} = y \tag{2}$$

$$\frac{dy}{dt} = t + y \tag{3}$$

Equation (2) is *autonomous* because the right side of the ODE in (2) does not involve the independent variable ( $t$  in this case). Equation (3) is not autonomous because the right hand side does involve the independent variable  $t$ . Equations (2) and (3) are *first order* ODEs because the derivatives in each equation is a first derivative. There are no second, third, or higher derivatives in any of these equations.

## 2 Initial Value Problems

If equations (2) and (3) are each given an initial condition, then each equation becomes an *initial value problem* (IVP).

$$\frac{dy}{dt} = y, \quad y(0) = 3 \quad (4)$$

$$\frac{dy}{dt} = t + y, \quad y(0) = 1 \quad (5)$$

In this activity you will work with several famous numerical routines that provide approximate solutions for IVPs.

## 3 Exact Solutions of IVPs

**Example 1** Consider the following IVP.

$$\frac{dy}{dt} = 2y + 1, \quad y(0) = 3 \quad (6)$$

Sketch the exact solution on the interval  $[0, 2]$ .

You can use the `dsolve` command<sup>1</sup> from the Symbolic Toolbox (Version 2) to find an exact solution of IVP (6) as follows:

```
>> y=dsolve('Dy=2*y+1','y(0)=3')
```

```
y =
```

```
-1/2+7/2*exp(2*t)
```

If you type `whos` at the Matlab prompt, Matlab responds with a description of the variables in your workspace. Note that `y` is a *symbolic object*. Use the Symbolic Toolbox's `ezplot` command to sketch the graph of this solution over the interval

```
>> ezplot(y,[0,2])
```

Enter the command

```
>> hold on
```

From now on, any time we invoke the `plot` command, the resulting plot will be superimposed on the existing figure window.

---

<sup>1</sup>Type `help dsolve` to read the helpfile for the command `dsolve`.

## 4 Euler's Method

Euler's method is a numerical routine for computing solutions of IVPs. Dr. Polking of Rice University has written a routine, `eul.m`, which will produce a numerical solution of an IVP using Euler's method. You will need to do two things to solve IVP (6):

- Create a function M-file defining the right side of the IVP.
- Use Dr. Polking's `eul.m` routine to find an approximate solution.

**Example 2** Use Dr. Polking's `eul` routine to plot the solution of the IVP

$$\frac{dy}{dt} = 2y + 1, \quad y(0) = 3 \quad (7)$$

on the interval  $[0, 2]$ . Use step sizes  $h = 0.5, 0.025,$  and  $0.0125$ .

The equation  $dy/dt = 2y + 1$  has the form  $dy/dt = f(t, y)$ , where  $f(t, y) = 2y + 1$ . We need to build a function M-file for the right side of IVP (7). That is, a file that evaluates  $f(t, y) = 2y + 1$ .

Open the M-File Editor/Debugger provided with Matlab 5<sup>2</sup>. Enter the following lines in the editor and save the file as `f.m`.

```
function yprime=f(t,y)
yprime=2*y+1;
```

The first step in creating a function M-file is to type the word `function`. You must then choose names for the dependent and independent variables. In this case, I have chosen `yprime` for the dependent variable (`yprime` seems a descriptive name since the function  $f(t, y) = 2y + 1$  is evaluating the derivative). The independent variables<sup>3</sup> are `t` and `y`. The last line of any function M-file assigns values to the dependent variable(s). In this case, `yprime` is assigned the value `2*y+1`. The semicolon suppresses the output.

You must always save the file with the same name as the function. In this case, save the file as `f.m`. If the first line of the function M-file had been `yprime=david(t,y)`, you would save the file as `david.m`.

You will now use `eul.m` to find a numerical solution of (7) on the interval  $[0, 2]$ . First, type `help eul` and read the resulting helpfile. Note that we need to pass this routine several parameters.

<code>'F'</code>	A string containing the name of the function M-file
<code>tspan</code>	The time interval in the form <code>[t0,tfinal]</code>
<code>y0</code>	The initial <code>y</code> -value
<code>ssize</code>	The step size

---

<sup>2</sup>You can actually use any editor you wish, but the one provided by Matlab is quite nice.

<sup>3</sup>Actually, `y`, representing the solution of the IVP, depends on the variable `t`. That is, `y` is a function of `t`. However, in the function M-file you are writing, both `y` and `t` serve as independent variables for the function `f`.

The function string name is 'f', the name of the function M-file we created to evaluate the right side of IVP (7) `Tspan` is the interval `[0,2]` and the initial  $y$ -value is 3. Use the first step size (`ssize`) requested in the directions:  $h = 0.5$ .

```
>> [teul,yeul]=eul('f',[0,2],3,0.5)
```

```
teul =
```

```
    0
  0.5000
  1.0000
  1.5000
  2.0000
```

```
yeul =
```

```
  3.0000
  6.5000
 13.5000
 27.5000
 55.5000
```

Add this numerical solution to your plot of the exact solution as follows.

```
>> plot(teul,yeul,'ro-')
```

You'll note that this numerical solution is not particularly accurate. You can improve the accuracy of Euler's method by decreasing the step size to  $h = 0.25$ .

```
>> [teul2,yeul2]=eul('f',[0,2],3,0.25)
```

```
teul2 =
```

```
    0
  0.2500
  0.5000
  0.7500
  1.0000
  1.2500
  1.5000
  1.7500
  2.0000
```

```
yeul2 =
```

```
  3.0000
```

```
4.7500
7.3750
11.3125
17.2188
26.0781
39.3672
59.3008
89.2012
```

Add this last solution to your plot.

```
>> plot(teul2,yeul2,'g*-'), shg
```

Note that this last solution, because of the smaller step size, is a better approximation of the exact solution. Decrease the step size to  $h = 0.125$  and plot the result. If you wish, remove the suppressing semicolon in the following command to view the results.

```
>> [teul3,yeul3]=eul('f',[0,2],3,0.125);
>> plot(teul3,yeul3,'ys-'), shg
```

Note that this last solution is the best approximation of the exact solution thus far.

**Experiment.** Take some time to experiment with smaller step sizes. Decrease the step size further and add the resulting numerical solution to your plot. Note that decreasing the step size can make for a better solution, but the cost in computer time goes up each time you decrease the step size. If you are working on a really fast computer, try a step size of 0.0001. Time your result by wrapping the command between `tic` and `toc`.

```
>> tic,[t,y]=eul('f',[0,2],3,0.0001);toc
```

## 5 Better Numerical Solvers

As you saw in the last example, You can improve the accuracy of Euler's method by decreasing the step size. However, each time you decrease the step size, the computer must make more calculations. This can be very inefficient. Let's look at some other solvers that can improve the accuracy of the solution without decreasing the step size.

### 5.1 RK2

The M-file `rk2.m`, a second order Runge-Kutta method, was also written by Dr. Polking. Type `help rk2` and read the resulting helpfile. Note that the calling sequence and the parameters are identical to those of `eul.m`. We proceed exactly as before, only we type `rk2` instead of `eul`. Let's use a step size of 0.5.

```

>> close all
>> ezplot(y,[0,2])
>> hold on
>> [teul,yeul]=eul('f',[0,2],3,0.5);
>> [trk2,yrk2]=rk2('f',[0,2],3,0.5);
>> plot(teul,yeul,'ro-',trk2,yrk2,'go-'),shg

```

Note that the `rk2` solution is a better approximation than the `eul` solution. Repeat this same set of commands, but use a stepsize  $h = 0.25$ . Repeat again with a stepsize  $h = 0.125$ .

## 5.2 RK4

The M-file `rk4.m` is a fourth order Runge-Kutta method whose use is identical to the routines `eul.m` and `rk2.m`.

```

>> close all
>> ezplot(y,[0,2])
>> hold on
>> [teul,yeul]=eul('f',[0,2],3,0.5);
>> [trk2,yrk2]=rk2('f',[0,2],3,0.5);
>> [trk4,yrk4]=rk4('f',[0,2],3,0.5);
>> plot(teul,yeul,'ro-',trk2,yrk2,'go-',trk4,yrk4,'yo-'),shg

```

Note that the `rk4` solution provides the best approximation of all. Repeat this same set of commands, but use a stepsize  $h = 0.25$ . Repeat again with a stepsize  $h = 0.125$ .

## 6 Homework

1. Consider the following IVP:

$$\frac{dy}{dt} = 2 - y, \quad y(0) = 1 \quad (8)$$

- (a) Use `eul.m`, `rk2.m`, and `rk4.m` to find numerical solutions of IVP (8) on the interval  $[0, 2]$ . Use a step size of  $h = 0.25$ .
  - (b) Use `dsolve` of the Symbolic Toolbox to find the exact solution.
  - (c) Plot the exact solution, the `eul.m` solution, the `rk2.m` solution, and the `rk4.m` solution, all on the same axis.
  - (d) Use `title`, `xlabel`, and `ylabel` to provide a title and labels for the axes.
  - (e) Use `gtext` to label each solution.
  - (f) Obtain a printout and hand in.
2. Repeat the above exercise with a stepsize of  $h = 0.125$ . Obtain a printout and hand in.

