LAB #8

Numerical Methods

Goal: The purpose of this lab is to explain how computers numerically approximate solutions to differential equations.

Required tools: MATLAB routine ***dfield*** ; numerical routines **eul, rk2, rk4**; m-files.

Discussion

In this lab you will approximate solutions to differential equations using ***dfield*** with the various methods : Euler (Tangent Line) Method, The Improved Euler Method (Runge-Kutta-2 Method) and the Runge-Kutta-4 Method. Also you will examine what happens when the step size h is decreased for a particular problem using these methods.

Assignment

(1) Use ***dfield*** for $0 \le t \le 3, \quad 0 \le y \le 8$ to plot the direction field for the equation

$$y' = y + t \qquad \text{(a)}$$

with initial condition $y(0) = 0$. Under the "*Options*" pull down menu of the Display Window, set the Solutions Direction to "*Forward*." The idea of the tangent line method is to follow each of the little lines for short distances. Specifically, you could start from an initial position, follow one direction line for a short distance on the t axis, then pick up another line, follow it for the same distance, etc., eventually approximating a solution. The distance on the t axis is referred to as the "step size" and is denoted by h. If the differential equation is written

$$y' = f(t, y) \qquad \text{(b)}$$

and our starting point is (t_0, y_0), then the slope of the line through (t_0, y_0) is

$$m_0 = f(t_0, y_0) \qquad \text{(c)}$$

then our endpoint is (y_1, t_1) where

$$t_1 = t_0 + h \quad \text{and} \quad y_1 = m_0 h + y_0 \qquad \text{(d)}$$

To see this for the differential equation (a), in the "*Options*" pull down menu of the Display Window, select "*Solver*" followed by "*Euler*." A popup "*Settings*" menu should appear. Change the "*Step Size*" setting to "1" and then click on the "*Change Settings*" button TWICE. Next, ask ***dfield*** to plot the solution with the initial data $y(0) = 0$. (Use the "*Keyboard Input*" option of "*Options*" pull down menu of the Display Window.) You should get a piecewise linear graph made up of three lines. Since our initial point is $(0, 0)$, formula (c) says that the slope of the first line segment is $m_0 = 0 + 0 = 0$. Since the step size is 1, we follow this line for 1 unit, arriving at the point $(t, y) = (1, 0)$. From formula (c), the slope of the next segment is $m_1 = 0 + 1$. Why is the slope of the third line segment 3 ?

(2) Next, we study the "Improved Euler" Method, also known as the "Runga-Kutta-2" Method. This method uses the *average* of the first two Euler slopes to determine which direction to follow. Thus, if our starting point is (t_0, y_0) and our differential equation is as in formula (b), then our slope is

$$m_0^* = \frac{1}{2}(f(t_0, y_0) + f(t_1, y_1)) \qquad \text{(e)}$$

where t_1 and y_1 are as in formula (d). Our endpoint is then determined by equations similar to (d), using m_0^* in place of m_0.

To see this on your graph, in the "*Options*" pull down menu of the Display Window, change the solver to "Runge-Kutta-2." Keep the solutions direction set to "*Forward*." Again, change the step size to 1 and use the "*Keyboard Input*" to enter the same initial data as before.

You should get a piecewise linear graph made up of three lines. The first line segment has slope 0.5, which is just the average of the first two "Euler" slopes. The right endpoint of this line is $(1, 0.5)$. The second improved Euler slope is the average of the slopes of the first two segments of the Euler approximation to the solution *beginning* at $(1, 0.5)$. To check this, do the following:

(i) Erase the Euler approximation from the display window. (Use the "*Delete a Graphics Object*" option of the Edit pull down menu of the Display Window. You will need to left click onto the solution to be deleted after selecting this option.)

(ii) Change the Solver back to "*Euler*," select "1" for the Step Size, and input the initial value $(1, 0.5)$ into ***dfield*** . This should produce the Euler approximation starting at $(1, 0.5)$.

Using formula (c), compute the slopes of the first two Euler lines in your picture. Using your picture, check that the slope m^* of the Improved Euler line starting at $(1, 0.5)$ is the average of the two Euler slopes. Then use formula (d) to compute the next point on the Improved Euler approximation and check that it agrees with your picture.

(3) In the Edit pull down menu of the Display window, select "*Erase All Solutions*" and then plot the Euler approximation to the solution to the differential equation (a) using step size $h = 0.5$ and initial data $(0, 0)$. Use formula (c) to compute the slope of the first three line segments. Show your calculations and get the graph printed.

(4) Plot the Improved Euler approximation to the solution to the differential equation (a) using step size $h = 0.5$ and initial data $(0, 0)$. Use formula (e) to compute the slopes of the first three line segments. Note that this will require first computing (t_1, y_1) at each point in question. Show your calculations and print the resulting graphs.

(5) Change the solver back to the ***dfield*** default solver (Dormand-Prince) and plot the solution with initial data $(0, 0)$. From your graph, what is the value of the maximum error in the Euler and Improved Euler methods (as compared with the

plot generated by ***dfield***) for this problem when $0 \le t \le 3$? (The error at any given t is the absolute value of the difference in the y-coordinates.)

(6) Close and restart ***dfield*** . Consider the differential equation $y' = (3-y)(y+1)$ with the initial condition $y(0) = 0$. Using the default ranges for ***dfield*** , plot both the solution and the Euler approximation (with step size $h = 0.5$). Try to explain why the Euler approximation has the "zig-zags". Print out the graph. For your explanation, note the direction of the slope lines above and below the line $y = 3$. What is the significance of the line $y = 3$?

(7) Now consider the differential equation

$$y' = (3-y)^2(y+1) \qquad \text{(f)}$$

with the initial condition $y(0) = 0$. Plot both the solution and then the Euler approximation (with step size $h = 0.5$) using the default range for t and the range $-1 \le y \le 10$ for y. Why does the Euler method produce such a bad approximation? Explain in similar terms to the explanation above. Try decreasing the step size to $h = 0.1$. Does this help ? Print this graph.

(8) Repeat the previous exercise using the Improved Euler Method. Decrease the step size until you get what seems to be a reasonable approximation.

(9) Compute the maximum error of the approximation to the solution of equation (f) with the initial data $\left(0, 1 + \frac{\mathbf{seed}}{10}\right)$ obtained using step size $h = 1$ and the

(i) Euler Method

(ii) Improved Euler (Runge-Kutta-2) Method

(iii) Runge-Kutta-4 Method.

Use the interval $0 \le t \le 5$ and use the plot from default solver for ***dfield*** as the "actual" solution.

(10) Write a discussion of your conclusions. Address the relative merits of the Euler, Improved Euler and Runga-Kutta methods. What you can conclude about how much faith you can put in numerical solutions ?

Question: Can you be certain that the graphs produced by ***dfield*** are correct ? (***dfield*** uses the Runga-Kutta method with a small step size.)

(11) The reader should not get the impression that the inaccuracies observed with overly large step sizes can be totally eliminated by making the step size sufficiently small. Even the best numerical algorithm can fail when applied to the wrong equation. Deciding whether the numerical approximations to the problem can be trusted is a significant and difficult problem.

Let us consider the initial value problem:

$$y' = t(y-1), \quad y(-10) = 0\,.$$

(i) Verify that $y(t) = 1 - e^{\frac{(t^2-100)}{2}}$ is the solution to the above initial value problem.

(ii) Here you will approximate the actual solution at $t = 10$ using the Euler (**eul**), Runge-Kutta-2 (**rk2**), and Runge-Kutta-4 (**rk4**) Methods. See the tutorial below. Find the largest h of the form $h = \dfrac{1}{2^m}$ such that the error is smaller than 10^{-2} and fill in the following table:

Method	h	Estimate at $t = 10$	Actual solution at $t = 10$
eul			0.000
rk2			0.000
rk4			0.000

(iii) Using **rk4**, find the largest h of the form $h = \dfrac{1}{2^m}$ such that the error is smaller than 0.0001. Any comments ?

Remark: There are numerical routines for the Euler, Runge-Kutta-2 and Runge-Kutta-4 methods (**eul, rk2, rk4**). To use these routines, you must first create the appropriate m-file and then use the proper syntax. Here is a very brief tutorial:

Numerical Methods & .m Files

In order to use MATLAB routines for the Euler, Improved Euler (Runge-Kutta-2) and Runge-Kutta-4 Methods, you will need the files **eul.m, rk2.m** and **rk4.m**, respectively. These files are already present on all ITaP machines as standard software. If you are using your own copy of MATLAB you may need to download these files at:

http://math.rice.edu/~dfield/

- You must first create a function file in the same directory (or folder) as your version of MATLAB. Pull down the **File** menu and select "**New M-File**". A window will pop up for you to create your function file. To create a function file for the function $f(t, y) = 6t^3 - e^{2y} + \frac{\sqrt{t}}{y}$, type :

```
function W=yp(t,y)
W=6*t^3-exp(2*y)+sqrt(t)/y;
```

(Don't forget the "**;**" at the end.)

- Save this file as a **.m file** with the SAME name as your function. The above example would be saved as "**yp.m**". You can check if your function has been saved by typing something like the following at a MATLAB prompt:

```
>> yp(1,3)
```

(you should get the value of $f(1, 3)$)

- Write the initial value problem as : $\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$. Assuming $f(t, y)$ was saved as the file **yp.m**, the syntax for **eul**, **rk2** and **rk4** will be the same.

  To approximate the actual solution to the IVP at t_L using the Euler Method **eul**, with given h, just type the following at a MATLAB prompt:

```
>> [t,y]=eul('yp',[t0,tL],y0,h);
```

(Check to make sure your version of MATLAB requires brackets, some versions do not. Type **help eul** to find out.) The approximations $y_0, y_1, y_2, \ldots, y_n$ are stored in the matrix **y**.

To print out *all* values of t and y, type : >> `[t,y]`

To print out the *last* value of y, type : >> `y(length(y))`

To plot the approximations, type : >> `plot(t,y)`