

MAE 305
Engineering Mathematics I
Princeton University

Assignment # 2

September 19, 1997

Due on Friday, 2PM, September 26, 1997

1. **Matlab Tutorials.** I am distributing—via the HANDOUT BOX outside my office—a “Matlab Primer” (hard copy) written by Dr. Sigmon of the University of Florida. Every unix system in the world does something differently, so not everything will work exactly the same (By copyright restriction, I am not allowed to modify the Sigmon Primer but I am allowed to distribute it). Run 'intro' and 'demo' of Matlab, and get the feel of the package. From the show of hands in class, most of you have heard of (and used?) matlab before. If you need help, go to the TA's, or send email to them or me.
2. **Learning Matlab.** It is not my intention to teach all the wonderful capabilities of Matlab. My goal is to show you just the capability to do ODE problems, and to display the results in graphical form—so that you can do ODE problems which are not on your short list of solvable problems using a computer. I want you to be able to write a 'm-file' function, and to use either 'ode23' or 'ode45' to solve quasi-linear ODE initial-value problems numerically. You can see for yourself the strengths and weaknesses of analytical methods versus numerical methods. You are encouraged to explore Matlab's considerable “other” capabilities, especially on matters related to matrices.
3. **Readings.** Boyce and DiPrima, Chapter 8.
§8.1, The Euler or Tangent Line Method. pp.415-423. Just scan it. No one should use this method anywhere in the civilized world. The section is to pay tribute to Euler, and to show you how terrible the accuracy is.

§**Errors in Numerical Procedures.** pp.423-427. Just scan it. Learn that error can come from two sources: truncation error (because a truncated Taylor Series was used in the computation), and round-off error (because the computer uses a finite number of binary digits inside the machine).

Skip §8.3. Very few people use this.

§**8.4, The Runge-Kutta Method.** pp.436-439. This is probably the most popular ODE solver. The whole method is given by equations (2) and (3) on page 436. Look at Table 8.4.1 on page 438! You will agree that any one caught using Euler or Modified Euler in solving an engineering problem should be locked up! (Note that Improved Euler is not that bad if you use a very small step). Ask me in class how does one pick the step size. (How come matlab needs no advice from you or me to pick the step size?)

The Backward Euler Formula. page 420. Don't need to read in detail. Just make a mental note that the backward Euler is in fact a pretty neat idea (which is followed up by the "Backward Differentiation Formulas" discussed on page 443) for "stiff" problems. What is a stiff problem? We will talk about that.

§**8.7, Systems of First Order Equations.** pp.454-456. Only 2 and 1/2 pages! And you can now do a "system" of ODEs! Not just one ODE for one dependent variables, but 34500 ODEs for 34500 dependent variables!

4. **Home Work Problems.** You know how to do the following problems analytically. Now lets do them using Matlab. All you need to do is to work out either a demo problem (see below) or the first problem. Once the first problem works, doing the rest requires only editing a new M-file.

Page 23. Linear 1st Order and Integrating Factor. Problems (13).
Do it for $0 \leq x \leq 5$.

Page 33. Bernoulli Equations. Problem 38. Do it for $1 \leq t \leq 5$ with initial condition $y(1)=1$.

Page 38. Separable Equations. Problem 23. Do it for $0 \leq x \leq 2$.

Page 45. Theorem 2.4.1. Problem 18, which is actually the Clairnaut Equation I talked about in class. This is a really tricky problem. Solve this problem for $2 \leq t \leq 5$. Comments?

Page 93. Homogeneous Equations. Problem 3. Do it for $1 \leq x \leq 3$ with initial condition $y(1)=1$. Then with $y(1)=0.5$. Plot both on the same graph.

Supplementary Notes

Leibnitz Rule. I stated it in class. Here it is again, for your reference. Given $g(x)$ defined by:

$$g(x) \equiv \int_{a(x)}^{b(x)} R(x, \xi) d\xi. \quad (1)$$

Its derivative with respect to x is given by the Leibnitz Rule:

$$\frac{dg}{dx} = \frac{db(x)}{dx} R(b(x), x) - \frac{da(x)}{dx} R(a(x), x) + \int_{a(x)}^{b(x)} \frac{\partial R(x, \xi)}{\partial x} d\xi. \quad (2)$$

Example: $a(x) = 0$, $b(x) = 2 + x^2$, and $R(x, \xi) = \xi^3 \cos(\xi - x)$. The answer is:

$$\frac{dg}{dx} = 2x \left((2 + x^2)^3 \cos(2 + x^2 - x) \right) + \int_0^{2+x^2} \xi^3 \sin(\xi - x) d\xi. \quad (3)$$

Quasi-Linear. An ODE is *quasi-linear* if its highest derivative occurs in a linear fashion—regardless of how the lower derivatives or the dependent variable itself appears. Hence, $f = a(x, y)y' - h(x, y) = 0$ is quasi-linear, regardless of how y appears in $a(x, y)$ and $h(x, y)$ —because the highest derivative term appeared in a linear fashion (*i.e.* the term containing the highest derivative is proportional to the highest derivative, the proportionality 'constant' itself may depend on y).

The Chain-Rule. Just a reminder of what the Chain-Rule is. Given a function $g(x, y, t)$ where $x = x(t)$ and $y = y(t)$, then the time derivative of g is given by:

$$\frac{dg}{dt} = \frac{\partial g}{\partial t} + \frac{\partial g}{\partial x} \frac{dx}{dt} + \frac{\partial g}{\partial y} \frac{dy}{dt} \quad (4)$$

Example: For $g(x, y, t) = ax + bt \sin(cy)$, we have

$$\frac{dg}{dt} = b \sin(cy) + a \frac{dx}{dt} + ct \cos(cy) \frac{dy}{dt}.$$

Some Notes on Matlab

The Matlab on the unix computers at Princeton is Professional version 4.2c. I believe the latest Students' Edition of Matlab is version 4a. They are nearly the same.

If you have the Students Edition, you can learn about ODEs from the demo provided. Unfortunately, 4.2c does not have an ODE demo.

If you have version 4a, type 'demo' at the >> Matlab prompt, and click first 'continue' then select 'Visit Matlab.' Then under Numerics select 'Select a demo' to click on 'Cmd Line Demos.' Then click on 'ODE' to see the demo.

Here comes a brief tutorial on a Mac or PC using Students' Edition of Matlab 4a.

- Start Matlab. Issue at the Matlab prompt:

what

and look at the list. There is nothing there named "diffy."

- Go to the 'File' menu, and select 'New M-File'. A window opens to receive your typed inputs.
- Type in the following:

```
function dydt=diffy(t,y);  
% everything after a % on a line is ignored as a comment  
dydt = -(1 + t * t) * y;
```

- Save the file with the name 'diffy.m' to your disk. You now have a M-file that allows you to integrate the ODE

$$\frac{dy}{dt} = -(1 + t^2)y$$

Type

what

at the Matlab prompt. You will see 'diffy' listed there.

- Suppose you want the solution starting at $x = 0$ and ends at $x = 5$, and the initial condition is $x(0) = 0.97$. Issue at the Matlab prompt:

```
[t, y] = ode45('diffy', 0, 5, 0.97);
```

You can try this with or without the semi-colon and see what the difference is. Now issue at the Matlab prompt:

```
plot(t, y, 'r')
```

And Voila! you have done it.

- Issue

```
[t, y] = ode45('diffy', 0, 5, 0.53); hold on; plot(t, y, 'g'); hold off
```

and you have two plots on the same graph! If you want to do some other equation, give it a name, and edit a M-file. And Voila! you can do it again.

- For the adventurous, look at page 140-141 of The Students' Edition of Matlab Users' Guide (SEMUG). You can find it in the reserved room of the Engineering Library. It is an example of TWO equations for TWO unknowns! The 'y' there is a vector in two-dimension, with components $y(1)$ and $y(2)$. The 'yprime' there is also a vector in two-dimension. Don't you agree this is neat stuff? Try

```
dsolve('Dy = (1 + a * x * x) * y')
```

and see what comes out. (This won't work in D-229 of E.Q. because the Macs there do not have *Symbolic Toolbox*).

- Read §5.19, 2-D graphics in SEMUG. You already have plotted a graph. But if you want to dress it up, read this section.