

MATHLAB and MATLAB 3

Rabinowitz

1. The goal of this session is to introduce you to *M*-files and allow you to try out some Euler and Runge-Kutta solvers.
2. When a function is used repeatedly, instead of retyping it each time, it is convenient to create a file, called a function *M*-file, to do this. To illustrate, suppose the function is $5(\sin t^3)e^{-t}$. Choose a name for it, e.g. `pro`. Click the mouse on **File**, scroll down to **New**, and scroll on **New** to **M-File**. This opens an *M*-file window in which you can enter the function (in array smart form) as follows

```
function y = pro(t)
y = 5.*(t.^3).*exp(-t);
```

Now click the mouse again on **File**, scroll to **Save**, and save it (e.g. in **BIN** as `pro.m`. Close `pro.m`.

3. If you want to compute `pro(7.7)`, in the command window, type

```
pro(7.7)
```

.

If you want to graph `pro(t)` over `[-1,5]`, type

```
t = -1:0.1:5;
plot(t,pro(t));
```

Also typing

```
grid;
title('y = pro(t)');
```

makes for a nicer output.

4. The name of an *M*-file must begin with a letter and consist entirely of letters, numbers, and underscores (`_`). No periods are allowed. In an *M*-file, everything

on a line after a % is ignored. This allows you to include comments such as why you created the file and any other facts you want to remember. After creating an *M*-file, if you want to recall it, type e.g.

`type pro`

to bring the file to the command window.

5. For the IVP

$$x' = x^2 \cos 2t$$

$$x(0) = 1$$

- (a) Create a function *M*-file as in (2) for the right-hand side. E.g. call it `ex 1.m` and type

```
function y = ex1(t,x)
y = x.^ 2.* cos (2.*t);
```

- (b) Since the equation is separable, the solution can be written down explicitly:

$$x(t) = \frac{1}{1-0.5 \sin 2t}$$

- (c) Now the Euler method will be used to find an approximate solution. The syntax for the Euler method is

```
[t,x] = eul('odefile', t0, tfinal, x0, stepsize);
```

(The Runge-Kutta methods use the same syntax with **eul** replaced by **rk2** and **rk4** respectively). Take 'odefile' = 'ex1', t0=0, tfinal = 6, x0 = 1 and the stepsize h=0.5. Enter Euler with this data followed by

```
plot(t,x)
```

to get a graph.

6. We want to calculate the approximate solution for several values of *h*, to compare it with the actual solution from (5b), and to calculate the largest difference between the two. To do this, it is convenient to create a new *M*-file called e.g. `comp.m`. In this file type:

```
t = 0 : 0.06 : 6;
x = ones(size(t))./(1-0.5*sin(2*t)); % the exact sol
```

```

[s,y]=eul('ex1', 0,6,1,h); % the approximate sol
plot(t,x,s,y,'o') % plot x and y
z = ones(size(s))./(1-0.5*sin(2*s)); % the exact sol at s pts
maxerror = max(abs(z-y)) % the largest error

```

The above is called a script-*M*-file. Such a file is a list of Matlab commands saved as a file. When its filename is entered in the Command Window, the commands in the file are executed in the given order. To illustrate, type

```
h = 0.5; comp
```

You get a graph of x and y as well as a statement in the Command Window

```
maxerror = 1.3475
```

Try $h = 0.1; 0.05; 0.01; 0.005$. Note that the error is linearly proportional to the step size as in the formula given in class. Print the graph for $h = 0.05$.

```
(print -z buffer)
```

7. Do the same for **rk2** and **rk4** with $h = 0.1$ and 0.05 and print out $h=0.05$.
8. Print out the maxerror computations.
9. Construct an *M*-file which graphs **eul**, **rk2** and **rk4** simultaneously and print out the $h = 0.1$ graph. (You should use different notations for the 3 approximate solutions.)
10. For the IVP

$$y' = x^3 y^3 \quad y(0) = 1$$

- (a) Find the exact solution and
- (b) for $t = 0 : 0.01 : 1$ print out a graph of eul, **rk4**, and the exact solution for $h = 0.1$ and $h = 0.05$.
- (c) Print out a max error computation for the same values of h .