Math 308 - Section 506 - Project 5

Instructor: Dr. Adam Larios

Due date: Thursday, 3 May 2012, 4:30 pm in Blocker 601

Instructions. While there are no specific neatness requirements, your work should look professional, and you should submit it in a form that would be appropriate for submitting to a boss at a job you care about. Credit will be given for full, complete, and thoughtful analysis; however, a good report is not necessarily a long report. Do not submit fluff or filler, but only quality work that you are proud of.

You are free to choose at most one partner, or work by yourself. If you submit your project as a group, submit it with all group member's names clearly labeled on the front. Please note that each person in a group is responsible for 100% of the work, so it is your responsibility to keep all group members on task, or to finish the work yourself.Submit all of your code[1], along with any relevant graphs, mathematical calculations, and explanations.

Make sure any graphs are well labeled and well referenced, and that they are easy to read. Do not just print out graphs and code and expect the grader to be able to follow it. You need to explain what you are doing at each step. Someone who is familiar with differential equations, but who does not know about this project, should be able to read your report and understand it completely.

By returning this project, you agree to follow the Aggie Honor Code. In particular, please be sure any code you represent as your own is 100% written by you and your group members.

Introduction We propose to implement a Runge-Kutta method to solve the following generic first-order ODE:

$$\begin{cases} \dfrac{d}{dt}y = f(t,y), \\ \quad y(t_0) = y_0, \end{cases}$$

for $t_0 \le t \le T$, where with think of t_0 as the initial time, and T as the final time. Let N be a positive integer, and define $h := (T - t_0)/N$. The follow numerical algorithm (written in pseudo-code) yields an approximation Y_i of y at $t_i = t_0 + (i-1)h$ for $i = 1, \ldots, N+1$.

[1]There is no need to submit code which was *not* written by your group.

$$
\begin{aligned}
t &\leftarrow t_0 \\
y &\leftarrow y_0 \\
t_i &\leftarrow t \\
Y_0 &\leftarrow y \\
&\text{FOR } i = 2, \ldots, N+1 \\
&\quad k_1 \leftarrow f(t, y) \\
&\quad k_2 \leftarrow f(t + 0.5 \cdot h, y + 0.5 \cdot h \cdot k_1) \\
&\quad k_3 \leftarrow f(t + 0.5 \cdot h, y + 0.5 \cdot h \cdot k_2) \\
&\quad k_4 \leftarrow f(t + h, y + h\, k_3) \\
&\quad t \leftarrow x + h \\
&\quad y \leftarrow y + h \cdot (k_1 + 2k_2 + 2k_3 + k_4)/6 \\
&\quad t_i \leftarrow t \\
&\quad Y_i \leftarrow y \\
&\text{END}
\end{aligned}
$$

Your first task will be to implement this in Matlab. To give you a guide, here is a Matlab implementation of the explicit Euler method. Try typing this method out yourself, and see if you can understand each line (it might take several read-throughs and a few Matlab runs before you really understand it).

```
%% FILE forward_euler.m %%

function [Y]= forward_euler(f,t0,T,y0,h)
% Solve dy/dt = f(t,y), y(t0)=y0
% for t0 <= t <= T, and with mesh size h

% Initialize variables
t = t0;
y = y0;
Y = zeros(1,N);
Y(1) = y; % initial value
i = 1;    % index

while t<T
  Y(i+1) = Y(i) + h*f(t,Y(i)); % Update approximation Y at t+h
  t = t+h; % Update t
  i = i+1; % Update i
end

%%% END FILE %%%
```

Example: With the above "forward_euler.m" file saved in the working directory [2], to solve the ODE

$$\begin{cases} \dfrac{d}{dt}y = \dfrac{2y}{t^2+1}, \\ y(t_0) = 1, \end{cases}$$

for $0 \leq t \leq 2$ using $h = 0.1$, execute the following commands in Matlab:

```
>> f = @(t,y) 2*y/(t^2+1);
>> Y = forward_euler(f,0,2,1,0.1);
>> plot(Y);
```

Part I. Implement the above Euler method, and then adapt your code in a different file to implement the Runge-Kutta method. To test your implementations, proceed as follows. We start by *choosing beforehand* an exact solution

$$y(t) = e^{-t}\sin(t^2),$$

with $t_0 = 0$ and $T = \pi \approx 3.14159$. Find y' by hand, and use it to deduce the corresponding expression for $f(t, y)$. Run your implementation on a fixed time interval with $N = 100, 200, 400, 800, 1600$ grid points, and plot the global error, defined by

$$E_N := \max_{1 \leq k \leq N-1} |y(t_0 + h \cdot k) - Y_k|,$$

versus N in a log-log plot.[3] Check that you obtain a line with the expected slope by comparing with the log-log plot of t^{-4}. Perform the same computation with the explicit Euler method given above and discuss your results.

Part II.

Extend your implementation of the above Runge-Kutta method for a system of two linear first-oder ODEs

$$\begin{aligned} \frac{d}{dt}y_1(t) &= f_1(t, y_1, y_2), \\ \frac{d}{dt}y_2(t) &= f_2(t, y_1, y_2). \end{aligned}$$

Next, consider an oscillating pendulum. Let $\theta(t)$ be the vertical angle of the pendulum at time t. Its evolution in time satisfies

$$\frac{d^2}{dt^2}\theta + \frac{g}{L}\sin(\theta) = 0,$$

where $g = 9.81 m/s^2$ and L is the pendulum length in meters.

[2] Note: The file "forward_euler.m" must be in the your current working directory for Matlab to be able to find it. Use the command `pwd` (print working directory) to check your current directory, `ls` (list) to list what's in it, and `cd directory name` to change directory.

[3] log-log plots are used to find the relationship between two quantities related by a power function. For example, if $y = x^{2.3}$, then $\log(y) = 2.3\log(x)$, so if we plot $\log(x)$ vs. $\log(y)$, we should see a line with slope 2.3. To plot a log-log plot in Matlab, just use `loglog` instead of `plot`. For example, you could use `loglog(1:N,e(1:N))` to plot your error.

- Write the above second-order ODE as a system of two first-order ODEs and use your 2×2 Runge-Kutta method to solve for $\theta(t)$, $0 \leq t \leq 10$, with $L = 1$, $\theta(0) = \pi/4$, $\theta'(0) = 0$, and $N = 200$, $N = 2000$, and $N = 20,000$. Plot θ vs. t and discuss your results. What happens if $\theta'(0) = 8$?

- It is very common for people to use the linear approximation to this problem, given by

$$\frac{d^2}{dt^2}\theta + \frac{g}{L}\theta = 0.$$

(This is sometimes called "the small-angle approximation." Set $L = 1$ and $\theta'(0) = 0$ and find the range of initial conditions $\theta(0)$ such that the linear model is accurate.