MATH 0290: Homework 2

Spring 2014

Due at the beginning of class on Wednesday, Jan. 22

You must show all your work to receive full credit. You are encouraged to discuss the homework with other students, but you must write up your own solutions. If you have any questions about the homework, please contact me in person or at `alk92@pitt.edu`.

Points will be deducted for any plots missing axis labels!

Problem 1

(A) For each of the following, classify whether the ODE is linear/not linear, separable/not separable, exact/not exact, and, if it is linear, whether it is homogeneous/inhomogeneous. To test if something is exact, you will need to put it in the form $P(x,y)dx + Q(x,y)dy = 0$. You don't need to show your work for this subproblem.

(i) $y' + 2y = 0$

(ii) $y' = -\frac{2xy+x}{x^2+y^2}$

(iii) $y' = -3y + \sin(x)$

(B) Find the general implicit solution to the following exact equation:

$$y' = -\frac{3x^2y + x}{x^3 + y}$$

(C) Suppose:

$$y' = -\frac{bx}{ay}$$

with $a > 0$, $b > 0$. Find the general solution. What does it look like? Next, assume that $y(0) = 1$. For what x is the particular solution defined?

Problem 2

(A) (PBA 2.4, #4) Find the general solution of the first-order linear equation:

$$y' + 2ty = 5t$$

(B) (PBA 2.4, #14) Find the particular solution for initial value problem:

$$y' = y + 2xe^{2x}; \;\; y(0) = 3$$

(C) (PBA 2.4, # 18) Find the solution of the initial value problem. Determine the interval of existence and sketch your solution.

$$xy' + 2y = \sin x; \;\; y(\pi/2) = 0$$

Problem 3

For this and the next problem, we will be using the Euler's method solver we implemented in class. In this problem, we will see how changing the timestep of the solver affects the accuracy of the solution. For simplicity, we'll use the same f as in class: $y' = f(t, y) = -y$ with initial condition $y(0) = 1$. We know the true solution: it is $y(t) = e^{-t}$, so we can compare our numerical solution to the true solution.

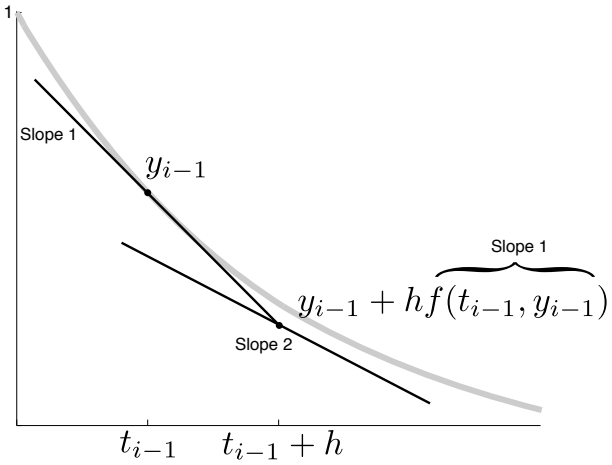
(A) Run the solver for different values of h : 0.01, 0.05, 0.1, 0.5, and 1. Use $t_1 = 0$ and $t_N = 10$. For each of these, plot $y_{\text{numerical}}$ vs t, where $y_{\text{numerical}}$ is the numerical solution. You can put all the lines on the same axis. The MATLAB command `hold on` will prevent a new plot from overwriting the previous one. Use a different color for each value of h and write down which color corresponds to which h.

(B) For each of these h, write the maximum value of $|y_{\text{numerical}} - e^{-t}|$. In MATLAB, typing `max(v)`, where `v` is a vector, will give the maximum element of that vector, and `abs(v)` will give the absolute value. You should see that smaller timesteps lead to smaller maximum errors.

Problem 4

In class, we implemented Euler's method for numerically solving first-order ODEs. In this problem, we'll improve on Euler's method, implementing a numerical solver knows as the *second order Runge-Kutta method*.

In Euler's method, we estimated y_i from y_{i-1} by drawing a line with slope $f(t_{i-1}, y_{i-1})$ starting at t_{i-1} and ending at $t_i = t_{i-1} + h$. We know this is only an approximation, because the slope of the true solution will change in between t_{i-1} and t_i. Let's make a small improvement in how we estimate the slope. Instead of estimating the slope only at (t_{i-1}, y_{i-1}), we'll estimate it at two points and average the result.



The above diagram shows what we need to do. We first estimate *slope 1*, which is given by $s_1 = f(t_{i-1}, y_{i-1})$. This is the same as Euler's method. Next, we estimate *slope 2*, the slope at $(t_{i-1}+h, y_{i-1}+hs_1)$. Slope 2 is given by $f(t_{i-1} + h, y_{i-1} + hs_1)$. Note that we are evaluating slope 2 at the point that we *would have* used as our expression for y_i if we were using Euler's method. But instead, we use the expression:

$$y_i = y_{i-1} + h\frac{s_1 + s_2}{2}$$

(A) Modify `euler.m` which we implemented in class (posted on the course website) so that it implements second order Runge-Kutta instead. Hand in the modified file with your homework.

(B) Now, let's compare the accuracy of Euler's method and second order Runge-Kutta. Again, we'll use $y' = f(t, y) = -y$ with initial condition $y(0) = 1$.

(i) Plot the error $y_{\text{numerical}} - e^{-t}$ for $t_1 = 0$, $t_N = 10$, $h = 0.01$. Do this for the Euler's method solution and the second order Runge-Kutta solution. You should see the second order Runge-Kutta method has a much smaller error. Hand in your plots.

(ii) For each of these methods, also write the maximum value of $|y_{\text{numerical}} - e^{-t}|$.

(C) MATLAB has a built-in solver called `ode45` that is a good choice for many applications. You can use it to to solve our problem by typing `[tnew ynew] = ode45(@myf,[0 10],1)`. What is the maximum magnitude of the error? You should find that it is pretty good. Now look at how many timesteps this solver used: `length(tnew)`. How many is it? You should see that it is much less than we used. This is because MATLAB varies the step size so that it takes large steps when y is changing slowly and small steps when it is changing rapidly. This allows it to achieve good accuracy with fewer steps. This is known as an *adaptive step size algorithm*.