Clearly states whose HW it is and what #

Patrick Moran
Pmm96
MAE 5735
HW #1

Clearly re-states problem. HW should function as a standalone document.

Problem #1

Consider an Euler integration of the differential equation $x = \dot{x}$ with initial condition $x_0 = 1$ over the interval $0 \leq t \leq 1$.

Part A

Matlab code is in a uniform width font, so it is easily readable

```
%Patrick Moran
%pmm96
%HW 1
%Problem 1 Part A
    %Use Euler integration with a reasonable step size to calculate x(1)
    % for the diff. eqn. x = x'.  Then compare it to the analytical value
    % 'e'.

    close all
    clear all
    clc

%define initial condition
    x0=1;
%define step size(inverse of number of steps)
    h=10^-6;

%Perform euler integration
    x=x0;
    for i = 1:(h^-1)
        x = x + h*x;
    end
    fprintf(['For a step size of %.0s seconds, the approximate value is'...
            ' x(1)=%.8f.\nThis is an error of %.4s.\n'],h,x,abs(x-exp(1)));
```

thoroughly commented

Program output clearly follows program.

```
For a step size of 1e-06 seconds, the approximate value is x(1)=2.71828047.
This is an error of 1.3591e-06.
```

Part B-C

```
%Patrick Moran
%pmm96
%HW 1
%Problem 1 Part B and C
    %Use Euler integration with a step size of 10^-n, with n=0,1,2,3... to
    %calculate x(1) for the diff. eqn. x = x'.  Then compare these to the
    % analytical value 'e' on a log-log plot.

    close all
    clear all
    clc

%define initial condition
    x0=1;
%define 'n' for minimum step size
    max_n=12;
%initialize variables to store the result and step size for each 'n'
    list_x=zeros(1,max_n);
    list_h=zeros(1,max_n);

%iterate over all n's
    for n = 1:max_n
        h = 10^-n;
        %Perform euler integration
```

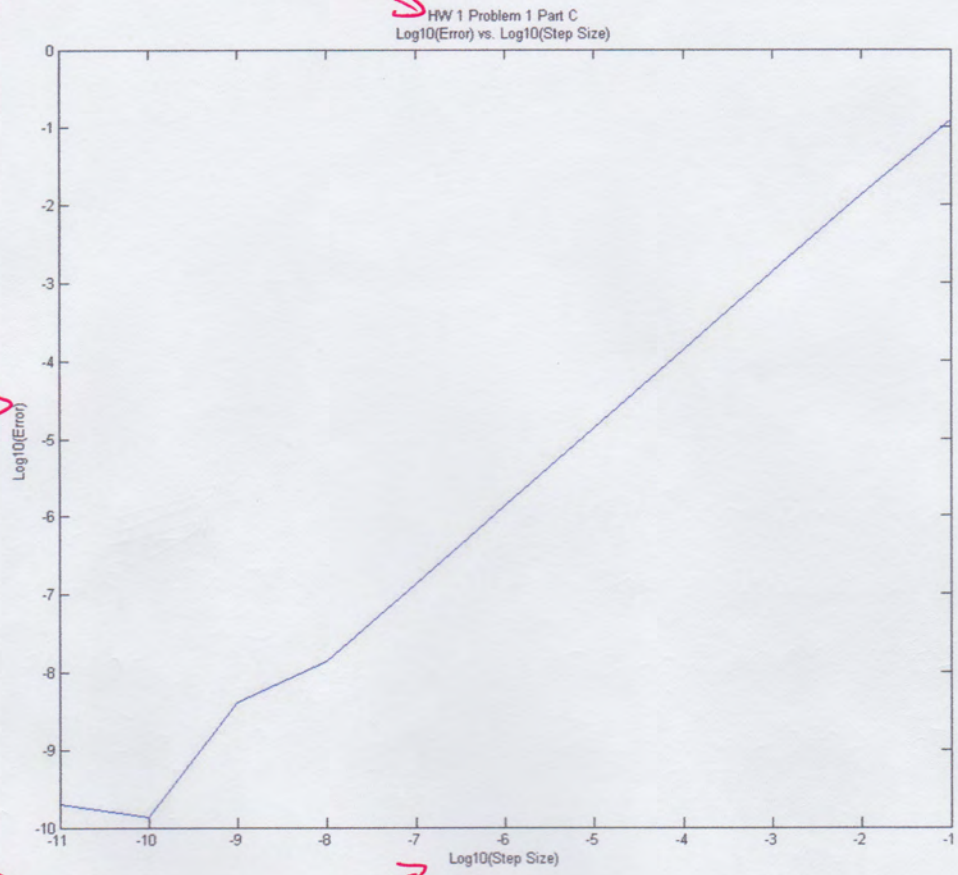
```
        x=x0;
        for i = 1:(h^-1)
            x = x + h*x;
        end
        %store results
        list_x(n)=x;
        list_h(n)=h;
    end

%plot the error vs. n on a log-log plot
plot(log10(list_h),log10(list_x-exp(1)))
axis equal
xlabel('Log10(Step Size)')
ylabel('Log10(Error)')
title(['HW 1 Problem 1 Part C',10,...
    'Log10(Error) vs. Log10(Step Size)']);
```

use axis equal

Please don't use "loglog", "semilogx" or "semilogy". Instead take the logarithims of your inputs to the "plot" command.

"log10" command computes a logarithim with base 10. "log" command uses base e, use the appropriate one.

Title and axes are clearly labeled. Use "title", "xlabel", and "ylabel" commands.



question clearly re-stated

Part D

The error for Euler integration of the differential equation, $x = \dot{x}$, with initial condition, $x_0 = 1$, on the interval $0 \leq t \leq 1$, is smallest for a step size of 10^{-10} seconds. This yields an error slightly larger than 10^{-10} at x(1).

Part E

This result for the most accurate step size can be rationalized. It is the point where the tradeoff between truncation error, due to the approximation inherent in the method, and roundoff error, due the inability of the computer to store numbers exactly, switch in dominance.

Part F

Even if an analytical result cannot be used as a comparison, it is possible to estimate an optimal step size for the problem. This can be obtained by estimating the size of the two types of error, round-off and truncation.

The truncation error is due to the fact that we are truncating the Taylor series of our differential equation as seen below.

$$x(t_0 + h) = x(t_0) + \frac{h\dot{x}}{1!} + \frac{h^2\ddot{x}}{2!} + \frac{h^3\dddot{x}}{3!} + \cdots$$

Where h is the step size. Because the Euler integration only deals with the first two terms of this series, we can estimate the error at each step as proportional to the h^2 term. However, we are interested in the error as it accumulates over the entire domain of our integration. This means that the error can be approximated as:

$$E_t \propto n * (h^2) = \frac{1}{h} * (h^2) = h$$

The round-off error has to do with the precision of the computer being used to do the calculation. If the machine epsilon is ε; we can approximate the magnitude of rounding error for each step as roughly $\varepsilon * x_n$. It is important to realize that these errors are random in direction for each step. It might round up, or it might round down. This is important because it means that the errors don't simply sum as might be expected. Instead, with errors up or down randomly, the rounding error for the entire integration can be shown to be:

$$E_r \propto \varepsilon\sqrt{n} = \varepsilon\sqrt{\frac{1}{h}} = \frac{\varepsilon}{\sqrt{h}}$$

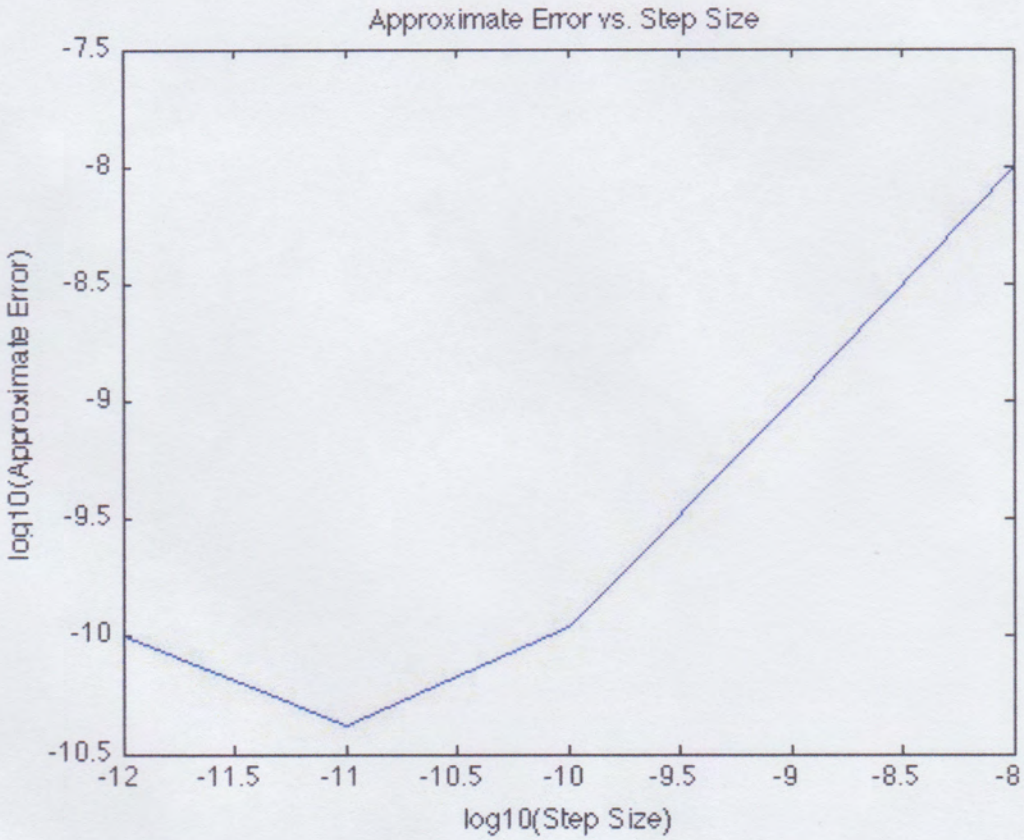
Recognizing that the total error is the sum of both sources error, and using Matlab's ε we get:

$$\varepsilon \cong 10^{-16}$$

$$E = E_r + E_t$$

$$E \propto \frac{10^{-16}}{\sqrt{h}} + h$$

The minimum of this function occurs around $h=10^{-11}$, which can be seen by quickly plotting it as seen below.



This is not exactly the optimal value which was previously found by comparing to the analytical solution. But it is very close.