ME EN 273 – Intro to Scientific Computing and CAE

Lab 09: ODE Solvers

To be submitted before 4pm on the day of your assigned lab period within the week of:
March 19, 2018 - March 23, 2018

Learning Goals:	"I can use and understand the solution mechanism of the ODE solution methods involved in this lab." "I understand, can explain, and can make use of the relationships between step size, method, and error."

Evaluation: Code that:

- Runs
- Satisfies all the requirements listed below
- Has a complete header
- Has comments for every substantial line of code

Self-evaluation of learning
Peer Observation Sheet

Materials: MATLAB Quick Reference Sheet
Peer Observation Sheet

Activities: Part I: Observe a classmate's coding process at any stage and complete a Peer Observation Sheet for that student (two 10-minute observation sessions).

Part I: Write five ODE solving routines:

```
[x, y] = Euler_XXXX(fh, Xrange, y0, N)
[x, y] = Heun_XXXX(fh, Xrange, y0, N, n)
[x, y] = Midpoint_XXXX(fh, Xrange, y0, N)
[x, y] = Ralston_XXXX(fh, Xrange, y0, N)
[x, y] = RK4_XXXX(fh, Xrange, y0, N)
```

Each of these solvers should accept a function handle, **`fh(x,y)`**, the desired x starting and ending points (e.g., **`Xrange = [xstart, xend]`**, an initial condition, **`y0`**, and the number of points, **`N`** to use between and including **`xstart`** and **`xend`**. Thus, the outputs should both be vectors of length **`N`**.

Note that the Heun_XXXX code requires one additional input, **`n`**, the number of iterations to perform for each step in the x -direction.

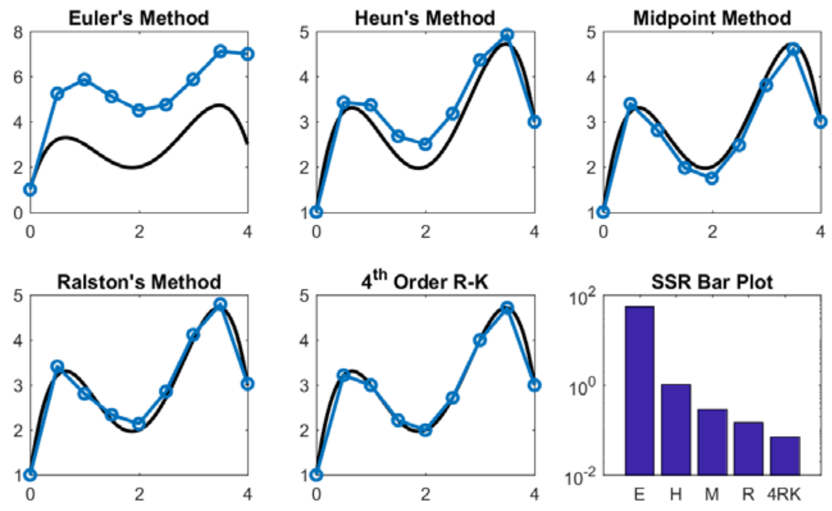
*NOTE: Your functions must be coded to accept a function handle that is a function of both x and y, **`fh(x,y)`** even though certain ODEs are only functions of x.*

Part II: Write a function that uses the above functions, and produces a figure showing the solution for each of the solutions listed above and a bar chart of their respective errors. This function should have the following syntax:

```
[SSR] = ODEcharts_XXXX(fh, Xrange, y0, N, fhs)
```

The first four inputs are the same as for the functions above, while the last input is the function handle for the *solution* to the ODE. For Heun's method, use 3 iterations (no "n" input required).

Finally, the function will plot a bar chart of the vector SSR, which is the *sum of the squares of the errors between each numeric solution and the analytic solution*.



Part III: Write a function that uses the functions from Part I to generate two (2) plots in a single figure. The first is a chart of percent error vs. step size, and the second is a plot similar to the one shown in Example 25.8 of the textbook. The syntax of this function should be as follows:

```
[ ] = ErrPlots_XXXX(fh, Xrange, y0, fhs, N)
```

The inputs for this function are the same as the previous function, but in this function, N is a vector of steps instead of a scalar. For example, N = [5, 10, 15, 20] would indicate that the Xrange interval should be solved with 5 steps, 10 steps, and so forth. The function does not return an output.

NOTE: Your second plot (% error vs. effort) will likely require additional calls to the lower-order methods. As shown in equation E25.8.1, computational effort can be quantified as the number of ODE function evaluations multiplied by the number of steps taken to the solution. So even when the same number of steps are used by each method, very different effort values will be obtained. In order to make your chart look like Figure 25.16, you'll need to add additional evaluations of the lower-order methods.

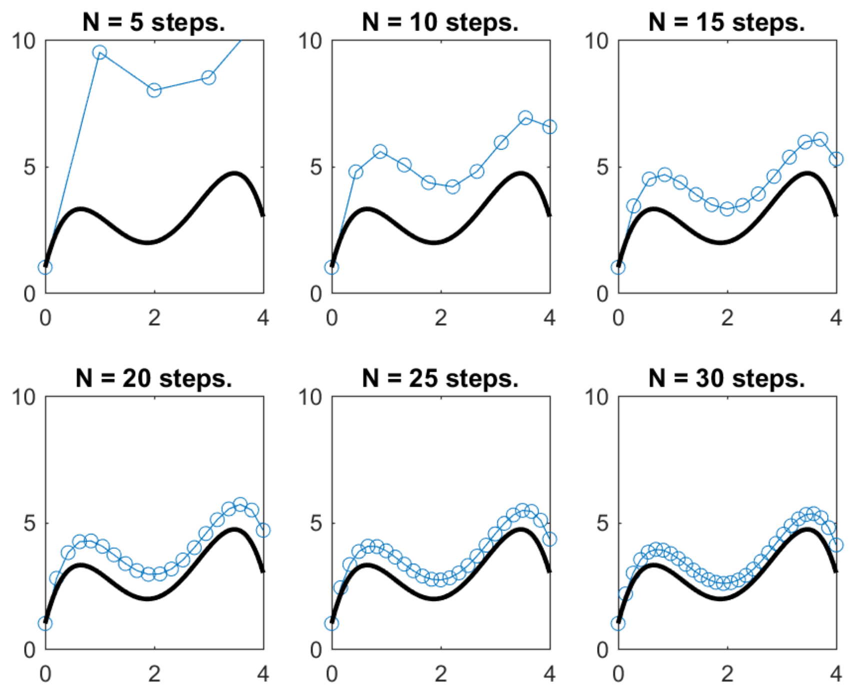
Deliverables (submit through the link below):	1. Self-evaluation of your learning relative to the learning goals of this assignment. 2. The observation form CPO_XXXX.pdf with your 4 digits in place of the Xs. 3. The seven (7) MATLAB function files listed above.

Submission Link:
https://goo.gl/forms/Q9v1OMLK3aMkkHdq2

Suggestions

IMPORTANT: Thoroughly and carefully test your solvers. Some of these methods are relatively forgiving – i.e., they may give seemingly correct answers even when your code is incorrect.

You may find it useful to animate your solvers. This can help you see how quickly each method converges to the solution as more steps are added. For example, here are images of the Euler's method solution for various step sizes:



Many different aspects of these methods can easily be animated using the same approach as used in Lab 2.