MAE 107 Numerical Methods
Instructor: Prof. McEneaney

Take-Home Final

As usual, you must show all work and all codes to get full credit.

1. (12 points) Employ Euler's method, the second-order Runge-Kutta method with $\beta_2 = 1/2$ and the fourth-order Runge-Kutta method to solve the following problem.

$$\dot{y} = e^{-y}\sin(t + 2\pi y),$$
$$y(0) = 1,$$

   up to time $T = 2$. Also create a second-order Runge-Kutta of your own, selection of β_2 different from those of $\beta_2 = 1/2$ and $\beta_2 = 1$, meeting the required three equations, and employ this one as well. Use varying numbers of steps for each method, specifically $n \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\}$. Plot the solutions on one or more grpahs, with different colors for the different methods. Assuming that fourth-order Runge-Kutta with $n = 8192$ generates what is essentially the true solution, plot the log of the error in $y(T)$ versus $\log(n)$ for all four methods on the same graph, and comment on the slopes. (Note: You need to write your own codes; you cannot simply call a built-in matlab Runge-Kutta routine.)

2. (3 points) A more useful graphic might contain plots of the log of the error in $y(T)$ versus $\log(F)$, where F is the number of function evaluations, as the number of function evaluations more accurately indicates the computational cost. Do this for the problem just above. How do the resulting plots change? Explain why they change in the way that they do.

3. (15 points) You would like to solve the problem

$$\dot{y} = g(t, y), \qquad y(0) = 4,$$

   up to time $T = 6$. Here, $g(t, y)$ is the solution of

$$z = e^{-(1+\sin(z))} - \sin^2(t + y)(1 + z^2)^{1/3}.$$

   For example, if $y = \pi/2$ and $t = 1$, then $g(t, y)$ is the solution of fixed-point equation $z = e^{-(1+\sin(z))} - \sin^2(1 + \pi/2)(1 + z^2)^{1/3}$. Use your fourth-order Runge-Kutta code. The function evaluations inside the Runge-Kutta code will use some sort of fixed-point or root-finding code. This function evaluation must be (or be inside) a separate function called by the Runge-Kutta code. Plot your solution approximations for several n values, say $n \in \{8, 16, 32, 64, 128, 256, 512\}$, for example.

The goal here is more practical than that in our usual problems. You will be calling one algorithm you've written inside another algorithm you've written. The task is also more open-ended than most of our problems; there is not a unique correct answer. You do not need to prove that your code meets a certain error bound, although it should certainly yield an excellent solution for the largest value of n that you choose. Very importantly, you should explain why you selected the various step-sizes and/or numbers of steps that you used. Would it make sense to have some relation between the numbers of steps in the Runge-Kutta component and the number of steps in the fixed-point/root-finding component? (Your response to this last part can be just a sentence or two.)

Keep in mind that your codes must be commented in order to receive full credit. Further, a portion of the grade will be based on the quality of the presentation of the analysis and results in your document.