Math 128C (De Loera) **Homework 2**
Due date: May 4, 2018

INSTRUCTIONS

All homeworks will have many problems, both theoretical and practical. Programming exercises need to be submitted via CANVAS.

Theory part must be scanned or, even better, please use LateX! Be organized and use the notation appropriately. Show your work on every problem. Correct answers with no support work will not receive full credit.

1. Section 5.5 #2 (you can use your code!)
2. Section 5.6 #1a, 13, #16,
3. Section 5.7 # 2b, 7
4. Section 5.8 #2d
5. Section 5.9 #2 (use code provided in class) and Discussion exercise #1
6. Section 5.10 #2, 4, 7, 8 and Discussion exercises #1, 2
7. MATLAB has several different methods implemented: Two of the most popular are **ode45** and **ode23** which implement some form of Runge-Kutta. In MATLAB you can set up various parameters using the command *odeset*, in particular, one can set up relative and absolute tolerance of error. This exercise gives you a chance to investigate the different behavior of the MATLAB code comparing MATLAB against itself.
   (a) COMPUTER PROJECT:The famous error function (see `https://en.wikipedia.org/wiki/Error_function`) appears in many areas of applied mathematics and statistics. It can be described by a differential equation as
   $$y'(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$$
   (b) The error function in MATLAB is encoded by the command $erf(x)$. Use ode23, ode45, ode113 to solve the differential equation on the interval $[0, 2]$. Use 5 different relative tolerances **RelTol**: 10^{-1},10^{-4},10^{-7}, 10^{-10} and 10^{-13}, in the following experiments:
   Compare the results of the 3 functions with that of the command $erf(x)$ at the points chosen by the solvers. (one triple of comparisons for each tolerance).
   (c) For each tolerance make 3 plots: (a) Plot the error of the 3 solutions with respect to that of the ("true") values given by erf in the same points t_i. (b) Plot the number of steps required by each method to reach the solution of a given tolerance error and (c) Show the execution time in seconds for all 3 codes (remember you have to each triple 3 times).
8. COMPUTER PROJECT: In this project you will play a little bit with Runge-Kutta methods.

(a) Write MATLAB code that implements the Runge-Kutta Fehlberg Algorithm (automatic step-size adjustment) which we call $RKF(t, y, h, \epsilon)$. One call from your code returns t and w have been advanced by a time-step equal to or smaller than h, and the local error per unit time should be no larger than ϵ. The returned h is a reasonable step- size to use for the next step. On the first call you will need to give it a guess for an appropriate step size.

(b) Apply your subroutine to compute the solution of the IVP:

$$y' = (-1/25)(t-2)^3 y^2; \quad y(-a) = \frac{100}{((-a-2)^4+1)}$$

Choose $a = 5$ and $a = 10$.

To approximate the solution to this problem, simply initialize t and y and call RK repeatedly until you get to $t = a$. It will be necessary for you in the last step to choose h so as to hit $t = a$ exactly.

Take the initial trial value of h to be equal to 0.1. Do all of your computations twice, first with $\epsilon = 10^{-11}$ and later with $\epsilon = 10^{-7}$. Compare the error in the computed values (Plot out your estimation and the exact function, and also plot out how h varies as time change in the same graph).