Assignment 12 (worth 200 points)
Due: 6:00PM 12/3/19

Write (in C++) a program that implements the 4th order Runge-Kutta algorithm to solve the following problem: Assume the trajectory of a cannonball, including the effects of atmospheric drag, is given by the equations

$$\frac{dx}{dt} = v_x \qquad (1)$$

$$\frac{dy}{dt} = v_y \qquad (2)$$

$$\frac{dv_x}{dt} = -\alpha v v_x \qquad (3)$$

$$\frac{dv_y}{dt} = -\alpha v v_y - g \qquad (4)$$

where $g = 9.81\ m/s,\ \alpha = 5 \times 10^{-5}\ m^{-1}$, and $v = \sqrt{v_x^2 + v_y^2}$. Suppose we launch a cannonball at an angle of 30° above the horizon with a speed of $v = 800\ m/s$. How far does the cannonball travel horizontally and what timestep size (Δt) did you have to use to find your answer? *You should put your answers in comment statements in your header block along with the results of a convergence test.* ***See page 2 for an important suggestions on how to proceed in developing this code. Submit only a single source code file containing all code.***

Note: The program should use C++ functions for the problem specific parts (initial conditions, number of equations, the right-hand-sides of the ODEs) and dynamic arrays so that the main driver program is generic (no problem specific information in the main program!). ***Failure to use C++ functions to implement the problem specific parts of the code will mean an automatic score of zero for your grade on this problem.***

Make sure that you code compiles (programs receive a zero if they are not able to be compiled on the Math SINC site machines) and make sure to test it by running it and observing that you get the correct answer! Submit only the source code file, i.e. the .cpp file containing the C++ code, by uploading it into blackboard using the "attachments" button under the assignment. **Do not submit the executable file. If you submit the executable file you will receive zero credit**

If you have any problems see the TAs or the instructor for help. Do not ask other students to help you debug your code. Submissions via email will not be accepted. **DO NOT WAIT UNTIL THE LAST MINUTE TO SUBMIT THE ASSIGNMENT! LATE ASSIGNMENTS WILL NOT BEACCEPTED.**

Note:

1. All programs should have a block of comment statements at the beginning of the code containing your name, **section number** (consult SOLAR if you are in doubt as to which section you are registered for), and a description of what the code does. Consult the lecture notes for examples.

2. Your file should be named in the form of <*yourlastname*>_<*yourIDNumber*>_<*hw#*>.cpp (Do not put the # sign in the file name!)

3. All programs must compile using the g++ compiler on the Mathlab machines. **Programs that do not compile will receive an automatic grade of zero.** There are no exceptions to this policy

4. Programs must be uploaded into the blackboard assignment page. Programs may not be submitted via email or hardcopy. Programs that are not uploaded into the blackboard assignment page will not be graded.

Suggestions

This code can be challenging to develop for novice C++ programmers. **Start working on this early! It is extremely unlikely that you will be able to complete this assignment if you start working on it the day before.** I suggest that you proceed in stages as follows:

> **Stage 1:** Develop a working code that implements Euler's method to solve the orbit verification problem using functions. Make sure that you code is able to give the correct answer (two complete circular orbits in two years) for the orbit verification problem. Plot the output with gnuplot and make sure that you are getting a reasonable answer.
>
> **Stage 2:** Expand your code to implement the 2^{nd} order Runge-Kutta method and apply it to the orbit problem to make sure you are getting the correct answer. Plot you results.
>
> **Stage3:** Expand your code to implement the 4^{th} order Runge-Kutta method and apply it to the orbit problem to make sure you are getting the correct answer. Plot your results.
>
> **Stage 4:** After you have completed steps 1-3 then you should write a new set of functions (save the verification test functions) for the number of equations, the initial conditions, and the right-hand-sides for the cannon ball problem. Compile and link your main function with the new set of functions to do the cannonball problem. Make a plot of your results to assess whether you are getting a reasonable answer. Then use your code to answer the question about the horizontal travel distance.

Asking for Help

The instructor and TAs are happy to help you when you encounter difficulties on this problem. But we will insist that you show us the successful results on each development stage before we will answer your questions about the next stage of the development process. If you cannot show us that you have successfully completed stage1 we will not help you with stage 2. There are no shortcuts on this problem so get started early.