Computational Physics: Assignment

We continue our study of motion with friction.   In this assignment however, we will solve for both position and velocity as a function of time using the RK algorithm. As discussed in assignment 2, an object moving through a viscous medium experiences a force due to friction. In our first approximation, we often say that the frictional force is

$$\vec{F}_f = -bv^n \hat{v}$$

Here the frictional force is in a direction that opposes the velocity.   The simple case that we will consider is

$$\vec{F}_f = -b\vec{v}$$

The presence of this force makes a dramatic change in the manner that projectiles travel. In fact, being able to calculate where a projectile (like a cannon shell or nuclear warhead) has driven a wide range of studies. In the case of projectile motion, our vector equation is

$$m\vec{a} = -m\vec{g} - b\vec{v}$$

This problem separates into x and z motions:

$$ma_x = -bv_x$$

$$ma_z = -mg - bv_z$$

Unlik‚e in the previous assignment, we will solve the second order differential equation for the position as a function of time. In this case $a_x = \dfrac{d^2 x}{dt^2}$ and $a_z = \dfrac{d^2 z}{dt^2}$, so both of our equations become second order differential equations

$$\frac{d^2 x}{dt^2} = -\frac{b}{m}\frac{dx}{dt} \qquad\qquad \frac{d^2 z}{dt^2} = -g - \frac{b}{m}\frac{dz}{dt}$$

$$\ddot{x} = -\frac{b}{m}\dot{x} \qquad\qquad\qquad \ddot{z} = -g - \frac{b}{m}\dot{z}$$

In this assignment, we wish to solve for and plot the position and velocity as a function of time. We wish to find the terminal velocity in each direction and how long it takes to get within 10% of that velocity.

Using Runge Kutta

In this case, rather than developing our own algorithm, we will use the tried and true RK algorithm in fourth order. To use this algorithm, we see eq. 25.25.20 in Abramowetz and Stegun.

We need to get our differential equation in the right form. RK requires

$$y'' = f(x,y,y')$$

$$\frac{d^2 y}{dx^2} = f(x,y,\frac{dy}{dx})$$

For our purpose, since our differential equations are in t,

$$\frac{d^2 x}{dt^2} = -\frac{b}{m}\frac{dx}{dt} \qquad\qquad \frac{d^2 z}{dt^2} = -g - \frac{b}{m}\frac{dz}{dt}$$

$$= f_x(t,x,\frac{dx}{dt}) \qquad\qquad = f_z(t,z,\frac{dz}{dt})$$

$$\ddot{x} = f_x(t,x,\dot{x}) \qquad\qquad \ddot{z} = f_z(t,z,\dot{z})$$

$$f_x(t,x,\dot{x}) = -\frac{b}{m}\dot{x} \qquad\qquad f_z(t,z,\dot{z}) = -g - \frac{b}{m}\dot{z}$$

Now that we have our equations in the form $\ddot{x} = f_x(t,x,\dot{x})$ and $\ddot{z} = f_z(t,z,\dot{z})$, we can easily apply the RK algorithm

$$x_{n+1} = x_n + h\left[\dot{x}_n + \frac{1}{6}(k_{1x} + k_{2x} + k_{3x})\right] \qquad z_{n+1} = z_n + h\left[\dot{x}_n + \frac{1}{6}(k_{1z} + k_{2z} + k_{3z})\right]$$

$$\dot{x}_{n+1} = \dot{x}_n + \frac{1}{6}(k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x}) \qquad \dot{z}_{n+1} = \dot{z}_n + \frac{1}{6}(k_{1z} + 2k_{2z} + 2k_{3z} + k_{4z})$$

$$k_{1x} = h\,f_x(t_n,x_n,\dot{x}_n) \qquad\qquad k_{1z} = h\,f_z(t_n,z_n,\dot{z}_n)$$

$$k_{2x} = h\,f_x(t_n + \frac{1}{2}h, x_n + \frac{h}{2}\dot{x}_n + \frac{h}{8}k_{1x}, \dot{x}_n + \frac{k_{1x}}{2}) \qquad k_{2z} = h\,f_z(t_n + \frac{1}{2}h, z_n + \frac{h}{2}\dot{z}_n + \frac{h}{8}k_{1z}, \dot{z}_n + \frac{k_{1z}}{2})$$

$$k_{3x} = h\,f_x(t_n + \frac{1}{2}h, x_n + \frac{h}{2}\dot{x}_n + \frac{h}{8}k_{1x}, \dot{x}_n + \frac{k_{2x}}{2}) \qquad k_{3z} = h\,f_z(t_n + \frac{1}{2}h, z_n + \frac{h}{2}\dot{z}_n + \frac{h}{8}k_{1z}, \dot{z}_n + \frac{k_{2z}}{2})$$

$$k_{4x} = h\,f_x(t_n + h, x_n + h\dot{x}_n + \frac{h}{2}k_{3x}, \dot{x}_n + k_{3x}) \qquad k_{4z} = h\,f_z(t_n + h, z_n + h\dot{z}_n + \frac{h}{2}k_{3z}, \dot{z}_n + k_{3z})$$

How to do it:

Read in the initial conditions and parameters from a file or from the terminal. Initial conditions include the magnitude of the velocity, the angle, the mass, the time step, the number of time steps, and the friction constant b.

Construct two functions that are the f functions used by RK. In the future, you will be able to compute any second order differential equation with minimal effort or change..

Use a simple do loop to step through and compute velocity at each time.

Write the results to files with the form:

Programming Suggestions:

Echo print inputs with units so that the user knows what he/she is calculating. You should also print to the terminal while you are printing to a file.

Check:

Design and carry out tests to show the program is correct. Does the trajectory show the correct behavior in the cases tested? Do you get linearly increasing z velocity with no friction? Does the x velocity remain constant? What happens if b=0?

How many time points are necessary for the solution to converge? Try computing the same trajectory using different time step/Number of step cases.

Finally...

Try using different masses. How does the terminal velocity depend on the mass? How does the time depend on the mass.

Try using frictional forces that go as the second and third power. Plot in each case, assuming the same initial velocity. Compare how the terminal velocity changes

Turn in your code and graphs that show your solutions for a set of inputs. Also write a brief paragraph (supported with results) that demonstrates that your code is correct...

25.5.14

P: $y_{n+1}=y_{n-5}+\dfrac{3h}{10}(11y_n'-14y_{n-1}'$

$\qquad +26y_{n-2}'-14y_{n-3}'+11y_{n-4}')+O(h^7)$

C: $y_{n+1}=y_{n-3}+\dfrac{2h}{45}(7y_{n+1}'+32y_n'$

$\qquad +12y_{n-1}'+32y_{n-2}'+7y_{n-3}')+O(h^7)$

Formulas Using Higher Derivatives

25.5.15

P: $y_{n+1}=y_{n-2}+3(y_n-y_{n-1})+h^2(y_n''-y_{n-1}'')+O(h^5)$

C: $y_{n+1}=y_n+\dfrac{h}{2}(y_{n+1}'+y_n')-\dfrac{h^2}{12}(y_{n+1}''-y_n'')+O(h^5)$

25.5.16

P: $y_{n+1}=y_{n-2}+3(y_n-y_{n-1})+\dfrac{h^3}{2}(y_n'''+y_{n-1}''')+O(h^7)$

C: $y_{n+1}=y_n+\dfrac{h}{2}(y_{n+1}'+y_n')-\dfrac{h^2}{10}(y_{n+1}''-y_n'')$

$\qquad +\dfrac{h^3}{120}(y_{n+1}'''+y_n''')+O(h^7)$

Systems of Differential Equations

First Order: $y'=f(x,y,z)$, $z'=g(x,y,z)$.

Second Order Runge-Kutta
25.5.17

$y_{n+1}=y_n+\dfrac{1}{2}(k_1+k_2)+O(h^3),$

$\qquad z_{n+1}=z_n+\dfrac{1}{2}(l_1+l_2)+O(h^3)$

$k_1=hf(x_n,y_n,z_n),\qquad l_1=hg(x_n,y_n,z_n)$

$k_2=hf(x_n+h,y_n+k_1,z_n+l_1),$

$\qquad l_2=hg(x_n+h,y_n+k_1,z_n+l_1)$

Fourth Order Runge-Kutta
25.5.18

$y_{n+1}=y_n+\dfrac{1}{6}(k_1+2k_2+2k_3+k_4)+O(h^5),$

$\qquad z_{n+1}=z_n+\dfrac{1}{6}(l_1+2l_2+2l_3+l_4)+O(h^5)$

$k_1=hf(x_n,y_n,z_n)\qquad l_1=hg(x_n,y_n,z_n)$

$k_2=hf\left(x_n+\dfrac{1}{2}h,y_n+\dfrac{1}{2}k_1,z_n+\dfrac{1}{2}l_1\right)$

$l_2=hg\left(x_n+\dfrac{h}{2},y_n+\dfrac{k_1}{2},z_n+\dfrac{l_1}{2}\right)$

$k_3=hf\left(x_n+\dfrac{1}{2}h,y_n+\dfrac{1}{2}k_2,z_n+\dfrac{1}{2}l_2\right)$

$l_3=hg\left(x_n+\dfrac{h}{2},y_n+\dfrac{k_2}{2},z_n+\dfrac{l_2}{2}\right)$

$k_4=hf(x_n+h,y_n+k_3,z_n+l_3)$

$l_4=hg(x_n+h,y_n+k_3,z_n+l_3)$

Second Order: $y''=f(x,y,y')$

Milne's Method
25.5.19

P: $y_{n+1}'=y_{n-3}'+\dfrac{4h}{3}(2y_{n-2}''-y_{n-1}''+2y_n'')+O(h^5)$

C: $y_{n+1}'=y_{n-1}'+\dfrac{h}{3}(y_{n-1}''+4y_n''+y_{n+1}'')+O(h^5)$

Runge-Kutta Method
25.5.20

$y_{n+1}=y_n+h\left[y_n'+\dfrac{1}{6}(k_1+k_2+k_3)\right]+O(h^5)$

$y_{n+1}'=y_n'+\dfrac{1}{6}(k_1+2k_2+2k_3+k_4)$

$k_1=hf(x_n,y_n,y_n')$

$k_2=hf\left(x_n+\dfrac{1}{2}h,y_n+\dfrac{h}{2}y_n'+\dfrac{h}{8}k_1,y_n'+\dfrac{k_1}{2}\right)$

$k_3=hf\left(x_n+\dfrac{1}{2}h,y_n+\dfrac{h}{2}y_n'+\dfrac{h}{8}k_1,y_n'+\dfrac{k_2}{2}\right)$

$k_4=hf\left(x_n+h,y_n+hy_n'+\dfrac{h}{2}k_3,y_n'+k_3\right)$

Second Order: $y''=f(x,y)$

Milne's Method
25.5.21

P: $y_{n+1}=y_n+y_{n-2}-y_{n-3}$

$\qquad +\dfrac{h^2}{4}(5y_n''+2y_{n-1}''+5y_{n-2}'')+O(h^6)$

C: $y_n=2y_{n-1}-y_{n-2}+\dfrac{h^2}{12}(y_n''+10y_{n-1}''+y_{n-2}'')+O(h^6)$

Runge-Kutta Method

25.5.22 $\quad y_{n+1}=y_n+h\left(y_n'+\dfrac{1}{6}(k_1+2k_2)\right)+O(h^4)$

$y_{n+1}'=y_n'+\dfrac{1}{6}k_1+\dfrac{2}{3}k_2+\dfrac{1}{6}k_3$

$k_1=hf(x_n,y_n)$

$k_2=hf\left(x_n+\dfrac{h}{2},y_n+\dfrac{h}{2}y_n'+\dfrac{h}{8}k_1\right)$

$k_3=hf\left(x_n+h,y_n+hy_n'+\dfrac{h}{2}k_2\right).$

Type to enter a caption.