1.

a)

sin(60) = h/1AU

sin(60)AU = h

cos(60) = w /1AU

cos(60) = w

(0.5AU, 0.86602540378AU)

(with (0,0) being the location of the sun)

Velocity of the asteroid is v = sqrt(1.3E20 \* (mass of asteroid)/1.496E11 meters)

Horizontal component is:

cos(60) = x / v

v\*cos(60) = x

Vertical component is:

sin(60) = y / v

v\*sin(60)= y

V = 11029857388.5 with asteroid mass of 0.00014E15kg

b)

GlowScript 2.7 VPython

'''

ex1.py

simulate the motion of an asteroid

'''

#define constants

G = 6.7E-11

AU = 1.5E11

year = 365.25\*60\*60\*24

scene = canvas(background = color.white)

asteroid\_mass = 0.00014E15

#initial position of asteroid CHANGE THIS FOR DIFFERENT RESONANCES

a\_B = 5.2\*AU\*pow(3/2,-2/3) #initial position of an asteroid for 3:1 resonance

#define objects

sun = sphere(pos = vec(0,0,0), radius = 7E10, mass = 2E30, color = color.yellow)

jupiter = sphere(pos = vec(5.2\*AU,0,0), radius = 3E10, mass = 1.898E27, color = color.blue) #earth

jupiter.vel = vec(0,sqrt(G\*sun.mass/(5.2\*AU)),0)

v = 11029857388.5

asteroid\_list = [] #define list to hold asteroid objects

for i in range(0,101):

angle = i\*2\*pi/100

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

#set up plotting

plot = gdisplay(x=0,y=400, height = 400, width = 600, title = 'r vs t asteroid', xtitle = 't', ytitle = 'r')

data = gdots(color=color.red)

data2 = gdots(color=color.blue)

#simulation

h = 1.0E6

t = 0

while True:

for asteroid in asteroid\_list:

asteroid.vel = asteroid.vel + (-G\*sun.mass\*(asteroid.pos - sun.pos)/mag(asteroid.pos-sun.pos)\*\*3 +

-G\*jupiter.mass\*(asteroid.pos-jupiter.pos)/mag(asteroid.pos - jupiter.pos)\*\*3)\*h

asteroid.pos = asteroid.pos + asteroid.vel\*h

if asteroid == asteroid\_list[0]:

data.plot(pos = (t/year, mag(asteroid.pos/a\_B) - 1))

jupiter.vel = jupiter.vel + -G\*sun.mass\*(jupiter.pos - sun.pos)/mag(jupiter.pos - sun.pos)\*\*3\*h

jupiter.pos = jupiter.pos + jupiter.vel\*h

t = t + h

rate(500)

c)

GlowScript 2.7 VPython

'''

ex1.py

simulate the motion of an asteroid

'''

#define constants

G = 6.7E-11

AU = 1.5E11

year = 365.25\*60\*60\*24

scene = canvas(background = color.white)

asteroid\_mass = 0.00014E15

#initial position of asteroid CHANGE THIS FOR DIFFERENT RESONANCES

a\_B = 5.2\*AU\*pow(3/2,-2/3) #initial position of an asteroid for 3:1 resonance

#define objects

sun = sphere(pos = vec(0,0,0), radius = 7E10, mass = 2E30, color = color.yellow)

jupiter = sphere(pos = vec(5.2\*AU,0,0), radius = 3E10, mass = 1.898E27, color = color.blue) #earth

jupiter.vel = vec(0,sqrt(G\*sun.mass/(5.2\*AU)),0)

v = 11029857388.5

asteroid\_list = [] #define list to hold asteroid objects

for i in range(0,101):

angle = i\*2\*pi/100

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

#set up plotting

plot = gdisplay(x=0,y=400, height = 400, width = 600, title = 'r vs t asteroid', xtitle = 't', ytitle = 'r')

data = gdots(color=color.red)

data2 = gdots(color=color.blue)

#simulation

h = 1.0E6

t = 0

while True:

for asteroid in asteroid\_list:

asteroid.vel = asteroid.vel + (-G\*sun.mass\*(asteroid.pos - sun.pos)/mag(asteroid.pos-sun.pos)\*\*3 +

-G\*jupiter.mass\*(asteroid.pos-jupiter.pos)/mag(asteroid.pos - jupiter.pos)\*\*3)\*h

asteroid.pos = asteroid.pos + asteroid.vel\*h

if asteroid == asteroid\_list[0]:

data.plot(pos = (asteroid.posx/mag(asteroid.pos), asteroid.posy/mag(asteroid.pos)))

jupiter.vel = jupiter.vel + -G\*sun.mass\*(jupiter.pos - sun.pos)/mag(jupiter.pos - sun.pos)\*\*3\*h

jupiter.pos = jupiter.pos + jupiter.vel\*h

t = t + h

rate(500)

d)

GlowScript 2.7 VPython

'''

ex1.py

simulate the motion of an asteroid

'''

#define constants

G = 6.7E-11

AU = 1.5E11

year = 365.25\*60\*60\*24

scene = canvas(background = color.white)

asteroid\_mass = 0.00014E15

#initial position of asteroid CHANGE THIS FOR DIFFERENT RESONANCES

a\_B = 5.2\*AU\*pow(3/2,-2/3) #initial position of an asteroid for 3:1 resonance

#define objects

sun = sphere(pos = vec(0,0,0), radius = 7E10, mass = 2E30, color = color.yellow)

jupiter = sphere(pos = vec(5.2\*AU,0,0), radius = 3E10, mass = 1.898E27, color = color.blue) #earth

jupiter.vel = vec(0,sqrt(G\*sun.mass/(5.2\*AU)),0)

v = 11029857388.5

asteroid\_list = [] #define list to hold asteroid objects

for i in range(0,101):

angle = i\*2\*pi/100

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

asteroid = ellipsoid(pos = vec(1.0004\*0.5\*AU,1.0004\*0.86602540378\*AU,0), length = 7E9, width = 9E9, height = 5E9, color = color.white)

asteroid.vel = vec(-v\*(1 + 0.00004\*random())\*cos(pi/6.2), v\*(1 + 0.00004\*random())\*sin(pi/6.2), 0)

asteroid\_list.append(asteroid)

#set up plotting

plot = gdisplay(x=0,y=400, height = 400, width = 600, title = 'r vs t asteroid', xtitle = 't', ytitle = 'r')

data = gdots(color=color.red)

data2 = gdots(color=color.blue)

#simulation

h = 1.0E6

t = 0

while True:

for asteroid in asteroid\_list:

asteroid.vel = asteroid.vel + (-G\*sun.mass\*(asteroid.pos - sun.pos)/mag(asteroid.pos-sun.pos)\*\*3 +

-G\*jupiter.mass\*(asteroid.pos-jupiter.pos)/mag(asteroid.pos - jupiter.pos)\*\*3)\*h

asteroid.pos = asteroid.pos + asteroid.vel\*h

if asteroid == asteroid\_list[0]:

data.plot(pos = (asteroid.posx/mag(asteroid.pos), asteroid.posy/mag(asteroid.pos)))

jupiter.vel = jupiter.vel + -G\*sun.mass\*(jupiter.pos - sun.pos)/mag(jupiter.pos - sun.pos)\*\*3\*h

jupiter.pos = jupiter.pos + jupiter.vel\*h

t = t + h

rate(500)

2.

a)

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 5 12:43:15 2018

oscillator.py

solves a damped harmonic oscillator using RK4

@author: Julian

"""

from \_\_future\_\_ import division

from numpy import sin, cos, exp

from numpy import arange

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

#define our acceleration function

def acc(x,v,t):

#return -k/m\*sin(x)-beta\*v + 0.52\*cos(omega\*t)

#return -0.5\*v-sin(x)+1.5\*cos(2.0\*t/3.0)

#return 3.0\*sin(4.0\*t) - 36\*x100000

return -25\*x-0.05\*v+100\*sin(5.6\*t)

#set initial values and range

k = 1.0 #spring constant

m = 1.0 #mass

omega = 0.666 #frequency of the driving force

x = 0

beta = 0.2 #damping coefficient

a = 0 #initial time

b = 40.0 #final time

v = 0

N = 10000

h = (b-a)/N

t = 0

#grid pointss in lists

tpoints = arange(a,b,h)

xpoints = []

vpoints = []

#implement RK4

for t in tpoints:

k1v = acc(x,v,t)\*h

k1x = v\*h

k2v = acc(x+k1x/2,v+k1v/2,t+h/2)\*h

k2x = (v+k1v/2)\*h

k3v = acc(x+k2x/2,v+k2v/2,t+h/2)\*h

k3x = (v+k2v/2)\*h

k4v = acc(x+k3x,v+k3v,t+h)\*h

k4x = (v+k3v)\*h

x = x+(1/6)\*(k1x+2\*k2x+2\*k3x+k4x)

v = v + (1/6)\*(k1v+2\*k2v+2\*k3v+k4v)

xpoints.append(x)

vpoints.append(v)

#plot results

plt.figure(1)

plt.xlabel('t')

plt.ylabel('x')

plt.plot(tpoints, xpoints, 'rx')

plt.show()

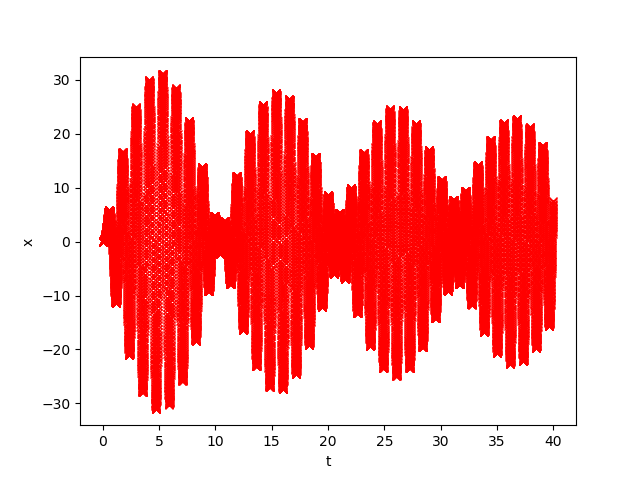
plt.figure(2)

plt.xlabel('x')

plt.ylabel('v')

plt.plot(xpoints, vpoints, 'b+')

plt.show()



The graph appears to be oscillating, but gradually decreasing in amplitude.

b) About 0.8 Seconds. Therefore frequency is about 1.25.

c)

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 5 12:43:15 2018

oscillator.py

solves a damped harmonic oscillator using RK4

@author: Julian

"""

from \_\_future\_\_ import division

from numpy import sin, cos, exp

from numpy import arange

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

#define our acceleration function

def acc(x,v,t):

#return -k/m\*sin(x)-beta\*v + 0.52\*cos(omega\*t)

#return -0.5\*v-sin(x)+1.5\*cos(2.0\*t/3.0)

#return 3.0\*sin(4.0\*t) - 36\*x100000

return -25\*x-0.05\*v+100\*sin(5.6\*t)

#set initial values and range

k = 1.0 #spring constant

m = 1.0 #mass

omega = 0.666 #frequency of the driving force

x = 0

beta = 0.2 #damping coefficient

a = 0 #initial time

b = 40.0 #final time

v = 0

N = 10000

h = (b-a)/N

t = 0

#grid pointss in lists

tpoints = arange(a,b,h)

xpoints = []

vpoints = []

#implement RK4

for t in tpoints:

k1v = acc(x,v,t)\*h

k1x = v\*h

k2v = acc(x+k1x/2,v+k1v/2,t+h/2)\*h

k2x = (v+k1v/2)\*h

k3v = acc(x+k2x/2,v+k2v/2,t+h/2)\*h

k3x = (v+k2v/2)\*h

k4v = acc(x+k3x,v+k3v,t+h)\*h

k4x = (v+k3v)\*h

x = x+(1/6)\*(k1x+2\*k2x+2\*k3x+k4x)

v = v + (1/6)\*(k1v+2\*k2v+2\*k3v+k4v)

xpoints.append(x)

vpoints.append(v)

#plot results

plt.figure(1)

plt.xlabel('t')

plt.ylabel('x')

plt.plot(tpoints, xpoints, 'rx')

plt.show()

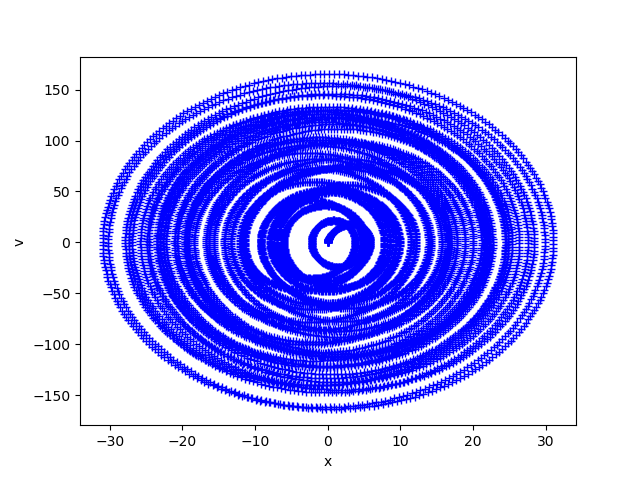
plt.figure(2)

plt.xlabel('x')

plt.ylabel('v')

plt.plot(xpoints, vpoints, 'b+')

plt.show()



The velocity and the x position are constantly changing directions, shown as the spiral-like pattern.

d)

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 5 12:43:15 2018

oscillator.py

solves a damped harmonic oscillator using RK4

@author: Julian

"""

from \_\_future\_\_ import division

from numpy import sin, cos, exp

from numpy import arange

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

#define our acceleration function

def acc(x,v,t):

#return -k/m\*sin(x)-beta\*v + 0.52\*cos(omega\*t)

#return -0.5\*v-sin(x)+1.5\*cos(2.0\*t/3.0)

#return 3.0\*sin(4.0\*t) - 36\*x100000

return -25\*x-0.05\*v+100\*sin(5.6\*t)

#set initial values and range

k = 1.0 #spring constant

m = 1.0 #mass

omega = 4.2 #frequency of the driving force

x = 0

beta = 0.2 #damping coefficient

a = 0 #initial time

b = 40.0 #final time

v = 0

N = 10000

h = (b-a)/N

t = 0

#grid pointss in lists

tpoints = arange(a,b,h)

xpoints = []

vpoints = []

#implement RK4

for t in tpoints:

k1v = acc(x,v,t)\*h

k1x = v\*h

k2v = acc(x+k1x/2,v+k1v/2,t+h/2)\*h

k2x = (v+k1v/2)\*h

k3v = acc(x+k2x/2,v+k2v/2,t+h/2)\*h

k3x = (v+k2v/2)\*h

k4v = acc(x+k3x,v+k3v,t+h)\*h

k4x = (v+k3v)\*h

x = x+(1/6)\*(k1x+2\*k2x+2\*k3x+k4x)

v = v + (1/6)\*(k1v+2\*k2v+2\*k3v+k4v)

xpoints.append(x)

vpoints.append(v)

#plot results

plt.figure(1)

plt.xlabel('t')

plt.ylabel('x')

plt.plot(tpoints, xpoints, 'rx')

plt.show()

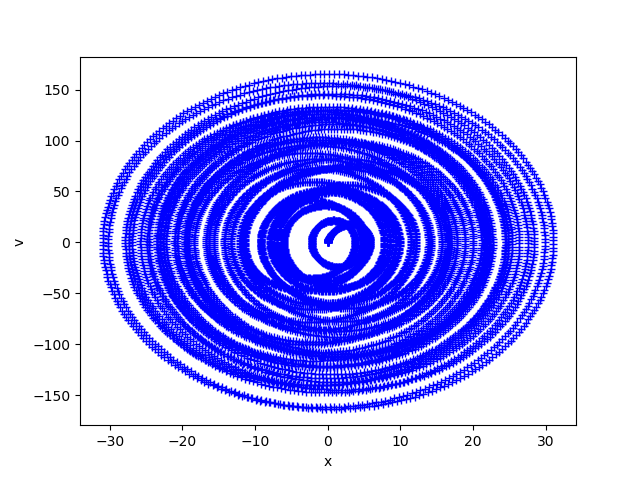
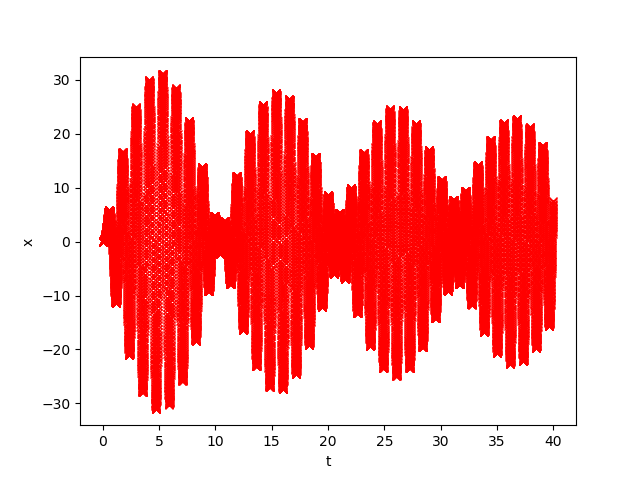
plt.figure(2)

plt.xlabel('x')

plt.ylabel('v')

plt.plot(xpoints, vpoints, 'b+')

plt.show()



Both results were very similar.