Name \_\_Jack Baumann\_\_\_ Grade \_100/100\_\_

Computational Physics Exam 1

# Instructions

You are to work all of the following problems by writing and running Python programs. Your solutions to the exam **must** include the following for each problem:

1. Answers to questions typed on this document.
2. Any associated graphs pasted into the document with captions.
3. Python source code for problems that require code, labeled with a descriptive name. Paste the code at the end of each problem

**Place your completed document along in the eCollege Dropbox by 9 p.m. Saturday September 24, 2016.**

Notes, textbook, and other *non-human* resources are permitted. You *may not* discuss the exam or your solutions with anyone but your instructor.If you have questions, please contact me via e-mail or phone (972-570-4262).

Best wishes!

1. (5 pts.)

(a) Write a line of code that reads a line of text from the user.

p = raw\_input("Please enter a line of text: ")

(b) Write a line of Python code that reads a floating-point number from the user.

f = float(input("Please enter a float: "))

1. (5 pts.) What is the output of the following code?

num1 = 5

if num1 >= 91:

num2 = 3

else:

if num1 < 6:

num2 = 4

else:

num2 = 2

x = num2 \* num1 + 1

print (x,x%7)

(21, 0)

1. (5 pts.) Rewrite following for loop into a while loop (which does the same thing as the for loop):

for i in range(1,10):

i = 1

while i < 10:

print "i = ", i

i += 1

print “i = “, i

1. (5 pts.) Translate the following while loop into a for loop (which does the same thing as the while loop):

for i in range(20, 0, -1):

print "i = ", i

i = 20

while (i > 0):

print “i = “, i

i = i - 1

1. (15 pts.) The Rydberg formula for the wavelengths λ of the hydrogen atom is given by

}where *R* is the Rydberg constant and *n* and *m* {\displaystyle n\_{2}\!}are integers such that *n* < *m* that {\displaystyle n\_{1}<n\_{2}\!}correspond to the principal quantum numbers of the orbitals occupied before and after. Write a program that computes and prints the first four wavelengths (with 4 decimal places) for the Lyman, Balmer, and Pachen series (*n*1 = 1,2,3, respectively) and the values of *n* and *m*. Take the Rydberg constant to have the value *R* = 1.097 x 10-2.

R = 1.097\*10\*\*-2

for n in range(1, 4):

for m in range(n+1, n+5):

l = 1.0/(R\*(1/float(n)\*\*2 - 1/float(m)\*\*2))

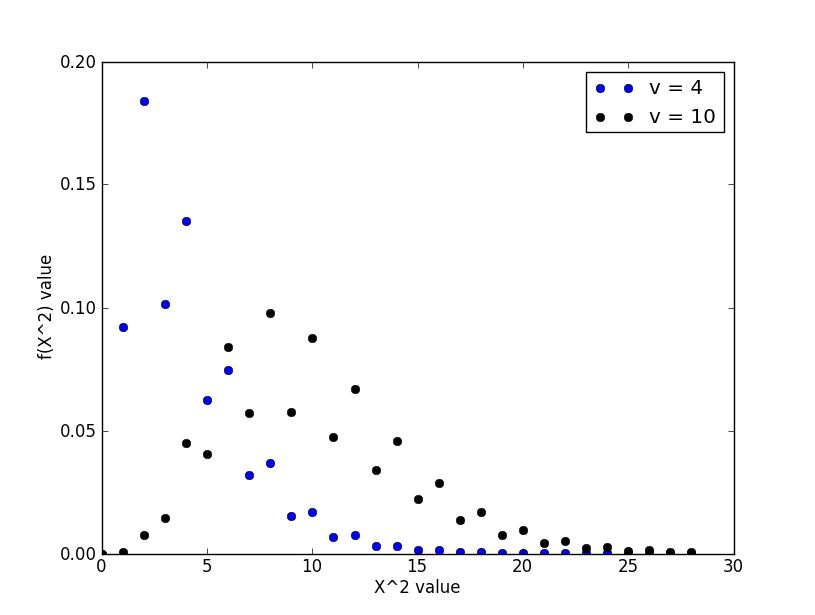
print "For n = {0} and m = {1}, the wavelength is {2:.4f}".format(n, m, l)

1. (15 pts.) The chi-square probability function is given by

where Γ(ν/2) is the gamma function. Write a Python program that plots the chi-square probability function for the range of χ2 equal to [0,28] and for ν = 4 and ν = 10 on the same graph.

What is the peak value for each value of ν?

The peak values for v=4 and v=10 are 0.18394 and 0.09768, respectively.



import matplotlib.pyplot as plt

import numpy as np

from scipy import special

def chisquare(x, v):

return (1/((2\*\*(v/2)) \* special.gamma(v/2))) \* (np.e)\*\*(-x/2) \* x\*\*((v/2)-1)

xs = []

f4 = []

f10 = []

for n in range(0, 29):

xs.append(n)

for x in xs:

f4.append(chisquare(x, 4))

plt.plot(xs, f4, "bo", label="v = 4")

for x in xs:

f10.append(chisquare(x, 10))

plt.plot(xs, f10, "ko", label="v = 10")

plt.xlabel("X^2 value")

plt.ylabel("f(X^2) value")

plt.legend(loc="best")

print "The peak values for v=4 and v=10 are {0:5.5f} and {1:5.5f}, respectively.".format(max(f4), max(f10))

plt.show()

1. (25 pts.) Write a Python program, analysis.py, that
2. reads a multicolumn spreadsheet of data (first column is the independent variable),
3. computes the average value of the dependent variable,
4. computes the standard deviation,
5. finds the least squares fit for a straight line to the data,
6. plots a graph of the data as circles with appropriate error bars,
7. plots the least squares fit to the data,
8. prints out the equation of the straight line, and
9. plots a graph of the residuals (dependent variable minus calculated value) for each point.

Run your program using the file Problem7Data.txt provided on eCollege (DocSharing) which is data on the difference in blue and visible filter magnitudes of a star as a function of the amount air the light travels through (airmass).

import matplotlib.pyplot as plt

import numpy as np

import math

#Part 1

data = np.loadtxt("Problem7Data.txt", skiprows=1)

independent = data[:, 0]

dependent = data[:, 1]

error = data[:, 2]

N = len(dependent)

#Part 2

avg\_dependent = 0.0

for d in dependent:

avg\_dependent += d

avg\_dependent /= N

print "The average value of the dependent variable (Air Mass) is: ", avg\_dependent

#Part 3

std\_dependent = 0.0

for d in dependent:

std\_dependent += (d - avg\_dependent)\*\*2

std\_dependent = math.sqrt(std\_dependent/N)

print "The standard deviation of the dependent variable (Air Mass) is: ", std\_dependent

#Part 4

sum\_x = 0 #Independent/B-V

sum\_y = 0 #Dependent/Air Mass

sum\_xx = 0

sum\_xy = 0

x\_max = 0 #Maximum independent variable value

for i in range(0, N):

if independent[i] > x\_max:

x\_max = idependent[i]

plt.plot(independent, dependent, "bo")

sum\_x += independent[i]

sum\_y += dependent[i]

sum\_xx += independent[i] \* independent[i]

sum\_xy += independent[i] \* dependent[i]

A = (sum\_xx\*sum\_y - sum\_x\*sum\_xy)/(N\*sum\_xx - sum\_x\*sum\_x)

B = (N\*sum\_xy - sum\_x\*sum\_y)/(N\*sum\_xx - sum\_x\*sum\_x)

#Part 5

plt.errorbar(independent, dependent, xerr=np.std(dependent)/N, fmt="b.", label="Data and error bars")

x\_calc = np.linspace(-0.85, -0.64)

y\_calc = A + B\*x\_calc

#Part 6

plt.plot(x\_calc, y\_calc, "r-", label="Best fit line")

#Part 7

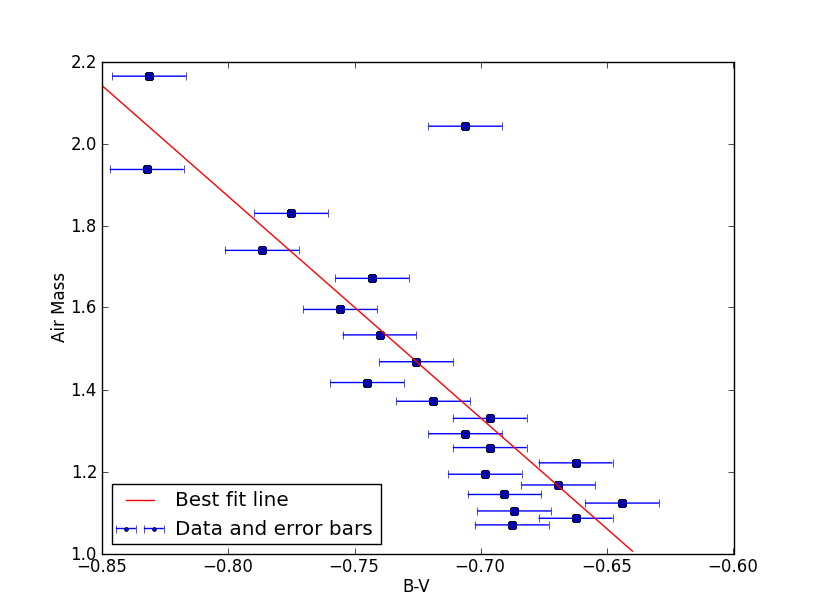
print "The least squares fit for a straight line to the data is: y = {0:5.2f} + {1:5.2f}x".format(A, B)

plt.xlabel("B-V")

plt.ylabel("Air Mass")

plt.legend(loc="best")

plt.show()



#Part 8

y\_residual = dependent - (A + B\*independent)

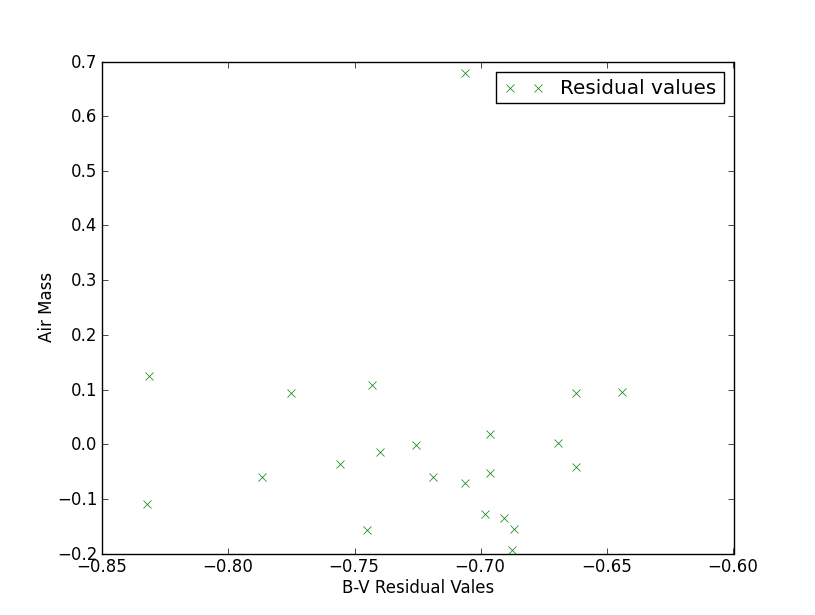
plt.plot(independent, y\_residual, "gx", label="Residual values")

plt.xlabel("B-V Residual Vales")

plt.ylabel("Air Mass")

plt.legend(loc="best")

plt.show()



1. (25 pts.) The Human Motion lab digitized the motion of a person performing a broad jump. The data can be found on eCollege (DocSharing) as Problem8Data.csv. Write a Python program that reads the data for the person’s left and right knees (three dimensions) and plots graphs of position, computed velocity and computed acceleration for each direction (*x,y,z*) of motion. Describe what the curves show about the person’s motion.

import matplotlib.pyplot as plt

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

data = np.loadtxt("Problem8Data.csv", skiprows=6, delimiter=',')

time = data[:, 0]

rx = data[:, 49]

ry = data[:, 50]

rz = data[:, 51]

lx = data[:, 67]

ly = data[:, 68]

lz = data[:, 69]

rvx = [0]

rvy = [0]

rvz = [0]

lvx = [0]

lvy = [0]

lvz = [0]

rax = [0, 0]

ray = [0, 0]

raz = [0, 0]

lax = [0, 0]

lay = [0, 0]

laz = [0, 0]

for t in range(1, len(time)):

rvx.append((rx[t] - rx[t-1])/(time[t]-time[t-1]))

rvy.append((ry[t] - ry[t-1])/(time[t]-time[t-1]))

rvz.append((rz[t] - rz[t-1])/(time[t]-time[t-1]))

lvx.append((lx[t] - lx[t-1])/(time[t]-time[t-1]))

lvy.append((ly[t] - ly[t-1])/(time[t]-time[t-1]))

lvz.append((lz[t] - lz[t-1])/(time[t]-time[t-1]))

for t in range(1, len(time)-1):

rax.append((rvx[t] - rvx[t-1])/(time[t]-time[t-1]))

ray.append((rvy[t] - rvy[t-1])/(time[t]-time[t-1]))

raz.append((rvz[t] - rvz[t-1])/(time[t]-time[t-1]))

lax.append((lvx[t] - lvx[t-1])/(time[t]-time[t-1]))

lay.append((lvy[t] - lvy[t-1])/(time[t]-time[t-1]))

laz.append((lvz[t] - lvz[t-1])/(time[t]-time[t-1]))

#3D Figure

fig = plt.figure()

graph = fig.gca(projection='3d')

graph.plot(rx, ry, rz, label="Right Knee Position")

graph.plot(lx, ly, lz, label="Left Knee Position")

graph.plot(rvx, rvy, rvz, label="Right Knee Velocity")

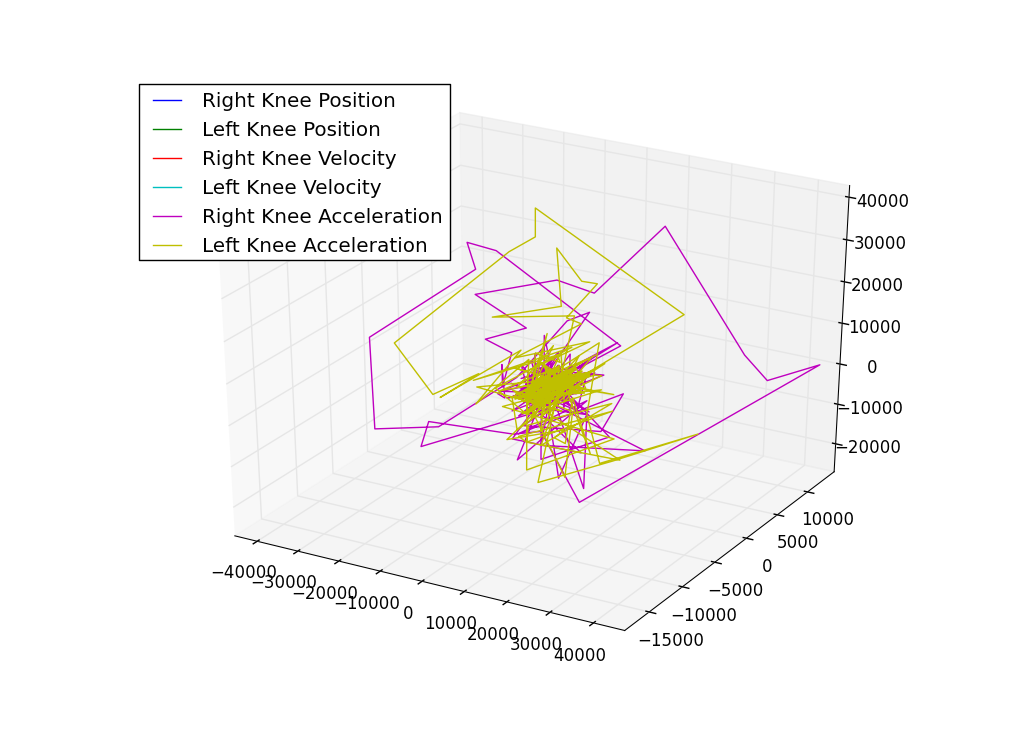
graph.plot(lvx, lvy, lvz, label="Left Knee Velocity")

graph.plot(rax, ray, raz, label="Right Knee Acceleration")

graph.plot(lax, lay, laz, label="Left Knee Acceleration")

plt.legend(loc='best')

plt.show()



#2D Figures

plt.figure()

plt.subplot(321)

plt.xlabel("Time (s)")

plt.ylabel("Right Foot Position (m)")

plt.plot(time, rx, 'ro', label='x')

plt.plot(time, ry, 'wo', label='y')

plt.plot(time, rz, 'bo', label='z')

plt.legend(loc="best")

plt.subplot(323)

plt.xlabel("Time (s)")

plt.ylabel("Right Foot Velocity (m/s)")

plt.plot(time, rvx, 'ro', label='x')

plt.plot(time, rvy, 'wo', label='y')

plt.plot(time, rvz, 'bo', label='z')

plt.legend(loc="best")

plt.subplot(325)

plt.xlabel("Time (s)")

plt.ylabel("Right Foot Acceleration (m/s^2)")

plt.plot(time, rax, 'ro', label='x')

plt.plot(time, ray, 'wo', label='y')

plt.plot(time, raz, 'bo', label='z')

plt.legend(loc="best")

plt.subplot(322)

plt.xlabel("Time (s)")

plt.ylabel("Left Foot Position (m)")

plt.plot(time, lx, 'ro', label='x')

plt.plot(time, ly, 'wo', label='y')

plt.plot(time, lz, 'bo', label='z')

plt.legend(loc="best")

plt.subplot(324)

plt.xlabel("Time (s)")

plt.ylabel("Left Foot Velocity (m/s)")

plt.plot(time, lvx, 'ro', label='x')

plt.plot(time, lvy, 'wo', label='y')

plt.plot(time, lvz, 'bo', label='z')

plt.legend(loc="best")

plt.subplot(326)

plt.xlabel("Time (s)")

plt.ylabel("Left Foot Acceleration (m/s^2)")

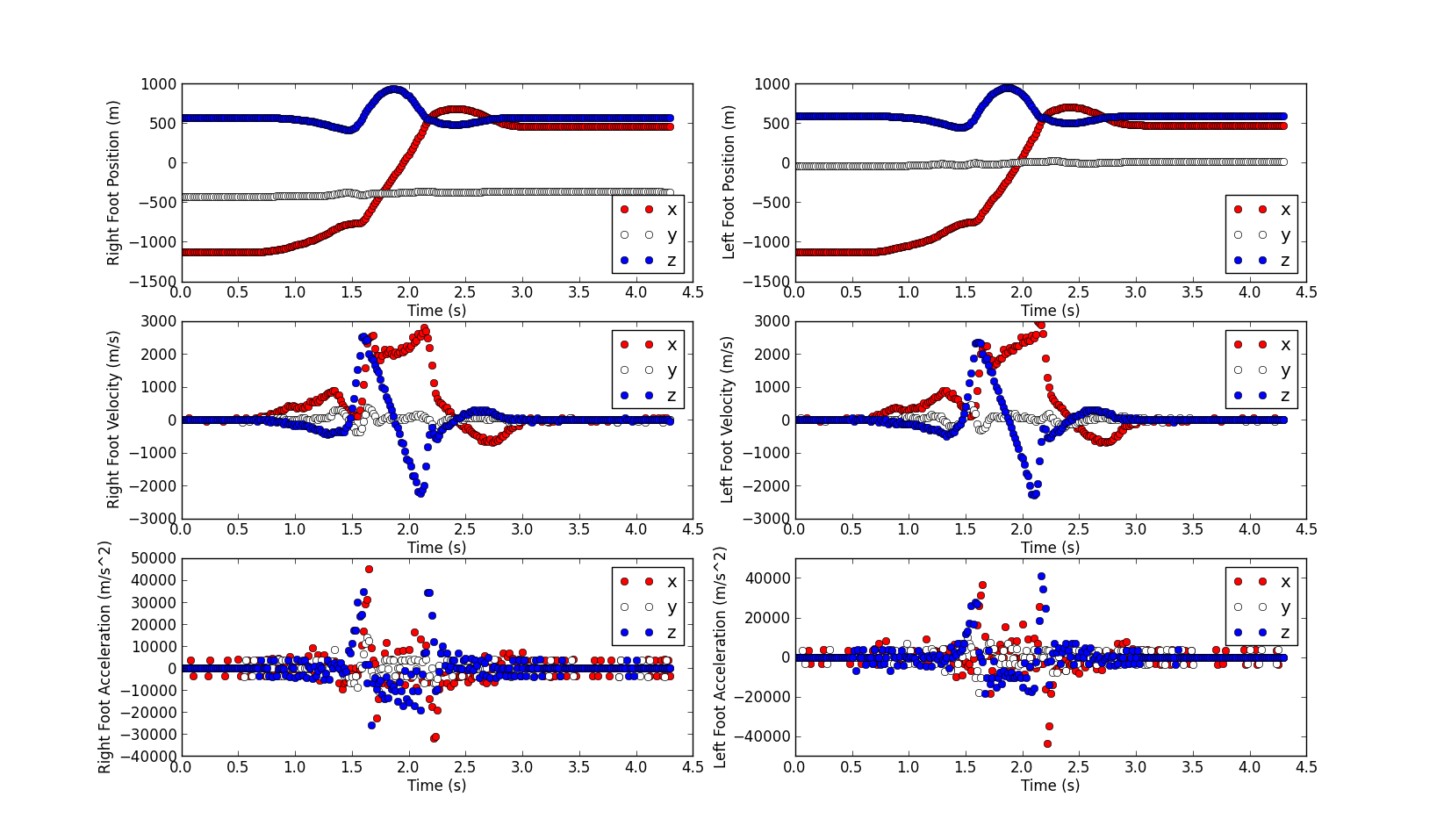
plt.plot(time, lax, 'ro', label='x')

plt.plot(time, lay, 'wo', label='y')

plt.plot(time, laz, 'bo', label='z')

plt.legend(loc="best")

plt.show() extra credit



Both feet are traveling almost exclusively in the positive x direction from time t = 1s to t = 2.5 seconds, when they move backwards a little. The feet also move in the z direction for a quarter second from time t = 1.5 to t =1.75 and then adjust themselves back to their previous constant z positions in the following quarter second. The feet do not move in the y direction. The y velocity and acceleration for both feet is approximately zero. The z velocity and acceleration only change when accounting for the change in z position from time t = 1.5s to t = 2s. The x velocity increases as the feet begin moving at t = 0.5s, then decreases a little before increasing again. When the feet reach maximum velocity at time t = 1.75s, the velocity plateaus and falls back down to zero and goes into the negatives before re-stabilizing at zero at time t = 3s. The acceleration in the x direction is constantly changing to account for the movement and change in the x position and velocity.