Assignment 1

Question 1

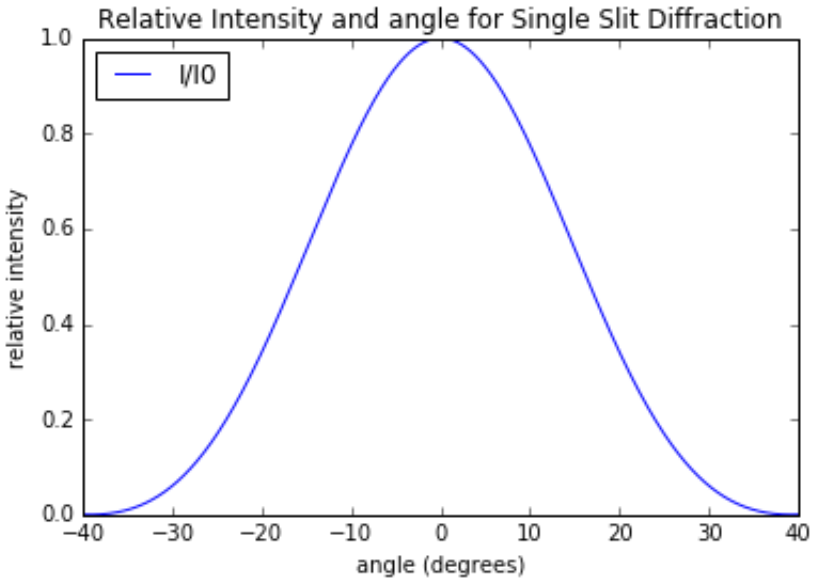
In [1]:
```
#This imports the necessary modules and defines the key variables.
from numpy import *
import matplotlib.pyplot as plt
%matplotlib inline
#theta is in degrees, wavelength and slitWidth are in nm
theta = linspace(-40,40,1000)
wavelength = 632.8
slitWidth = 1000
```

In [2]:
```
#Defines a function for B(d)
def Beta(xList,x):
    B = (pi*x/wavelength)*sin(xList*pi/180)
    return B
```

In [3]:
```
#Produces and array of y-values for the single slit for each input angle
def singleSlit(xList):
    B = Beta(xList, slitWidth)
    intensity = (sin(B)/B)**2
    return intensity
```

In [4]:
```
#Plots a function from the given x and y arrays
def plotFunction(xList, yList, figTitle):
    plt.figure()
    plt.plot(xList, yList, label = 'I/I0')
    plt.xlabel('angle (degrees)')
    plt.ylabel('relative intensity')
    plt.title(figTitle)
    plt.legend(loc='upper left')
    plt.show()
```

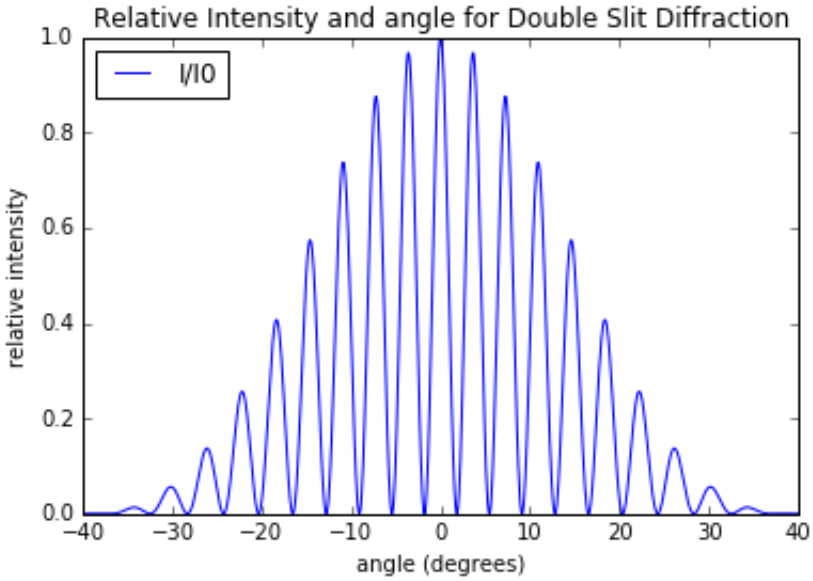
In [5]:
```
#Actually plotting the single slit diffraction
relativeIntensity = singleSlit(theta)
plotFunction(theta,relativeIntensity,"Relative Intensity and angle for Single Sli
t Diffraction")
```



Question 2

In [6]:
```
a = 10000 #The slit spacing, also in nm
#a function for the intensity as a function of angle
def doubleSlit(xList):
    B = Beta(xList, slitWidth)
    intensity = ((sin(B)/B)**2)*(cos(Beta(xList, a)))**2
    return intensity
```

In [7]:
```
doubleSlitIntensity = doubleSlit(theta)
plotFunction(theta, doubleSlitIntensity, "Relative Intensity and angle for Double
Slit Diffraction")
```



Question 3

In [8]:
```
#A function for calculating the mean of an array of numbers
def myMean(xArray):
    mean = sum(xArray)/size(xArray)
    return mean
#Note that to avoid the use of sum() and size(), I could have made a loop with a
counter. In fact, I did this originally, but
#decided that it would be nicer to use the existing BIF's.

#Testing the mean function
data = arange(5,15)
mean = myMean(data)
print(mean)
```

```
9.5
```

In [9]:
```
#Calculates the standard deviation for an array of numbers. Uses sample std formu
la
def myStd(xArray):
    mean = myMean(xArray)
    tempSum = 0
    for x in xArray:
        tempSum += (x - mean)**2
    sdt = sqrt(tempSum/(size(xArray)-1))
    return sdt
#Testing it with the same array as for mean
sdt = myStd(data)
print(sdt)
```

```
3.0276503541
```

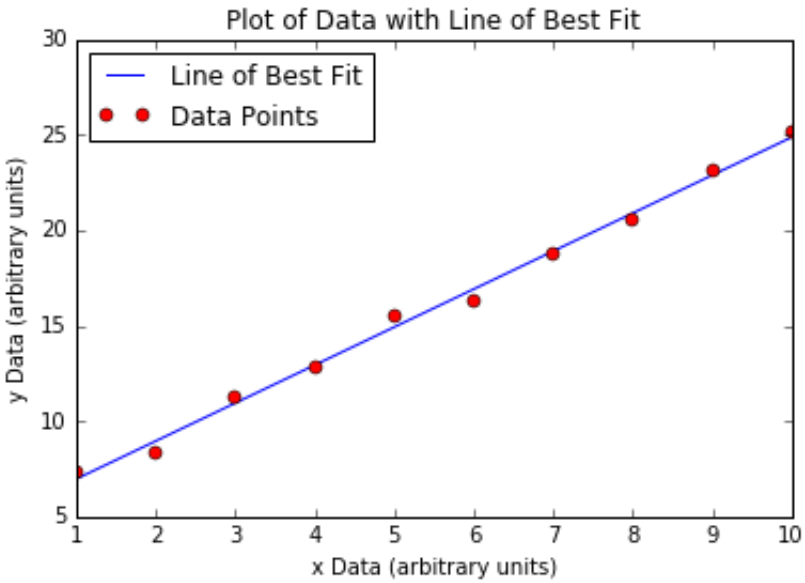
Question 4

In [10]:
```
#Calculates the standard error for an array of numbers
def myStdErr(xArray):
    std = myStd(xArray)
    stderr = std/sqrt(size(xArray))
    return stderr
#Testing the mean, standard deviation, and standard error functions
moreData = [20.28, 21.26, 20.96, 20.70, 20.31, 21.16, 20.60, 20.36, 20.55, 19.95]
anotherMean= myMean(moreData)
anotherStd = myStd(moreData)
stdErr = myStdErr(moreData)
print(anotherMean, anotherStd, stdErr)
```

```
20.613 0.415827154263 0.131496092042
```

Question 5

In [11]:
```
#Plots a linear regression on an array of data. Calculates the error of the slope
and y-intercept.
def myLinReg(xArray, yArray):
    if size(xArray) != size(yArray):
        print("There are an unequal number of x and y inputs for this function.")
        return
    N = size(xArray)
    D = N*sum(xArray**2)-(sum(xArray))**2
    yint = (sum(xArray**2)*sum(yArray)-sum(xArray)*sum(xArray*yArray))/D
    slope = (N*sum(xArray*yArray)-sum(xArray)*sum(yArray))/D
    yFit = yint + slope * xArray
    h = yArray - yFit
    sigmay = sqrt((sum(h**2))/(N-2))
    yintError = sigmay*sqrt(sum(xArray**2)/D)
    slopeError = sigmay*sqrt(N/D)
    plt.figure()
    plt.plot(xArray, yFit, label = 'Line of Best Fit')
    plt.plot(xArray, yArray, 'ro', label = 'Data Points')
    plt.xlabel('x Data (arbitrary units)')
    plt.ylabel('y Data (arbitrary units)')
    plt.title('Plot of Data with Line of Best Fit')
    plt.legend(loc='upper left')
    plt.show()
    return[yint, slope, yintError, slopeError]
xData = array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
yData = array([7.371, 8.292, 11.296, 12.805, 15.550, 16.340, 18.789, 20.493, 23.1
27, 25.104])
#Displays in the order, y-int, slope, y-int error, slope error
myLinReg(xData, yData)
```



Out[11]:
```
[4.9777333333333305,
 1.9889030303030317,
 0.31965202134113707,
 0.05151658785876384]
```

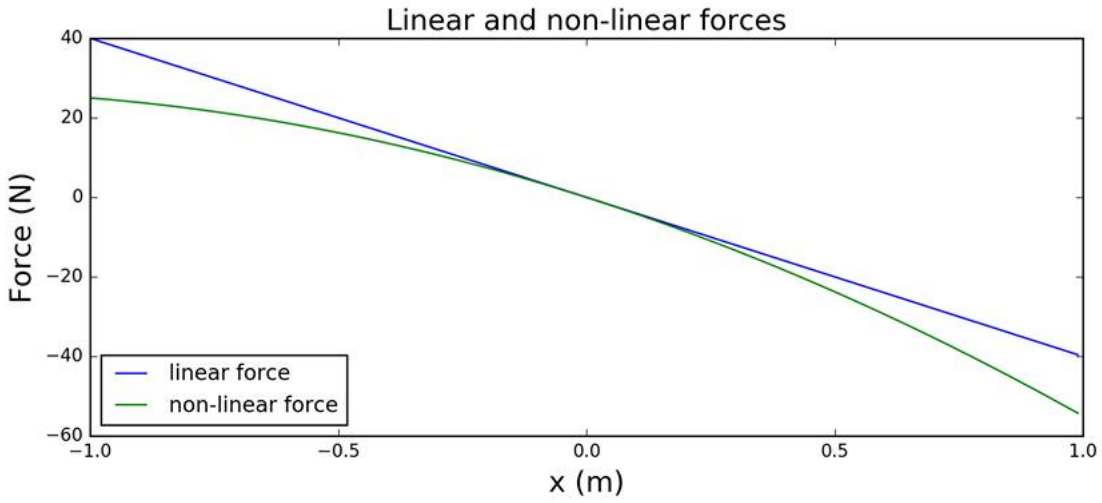
Question 6

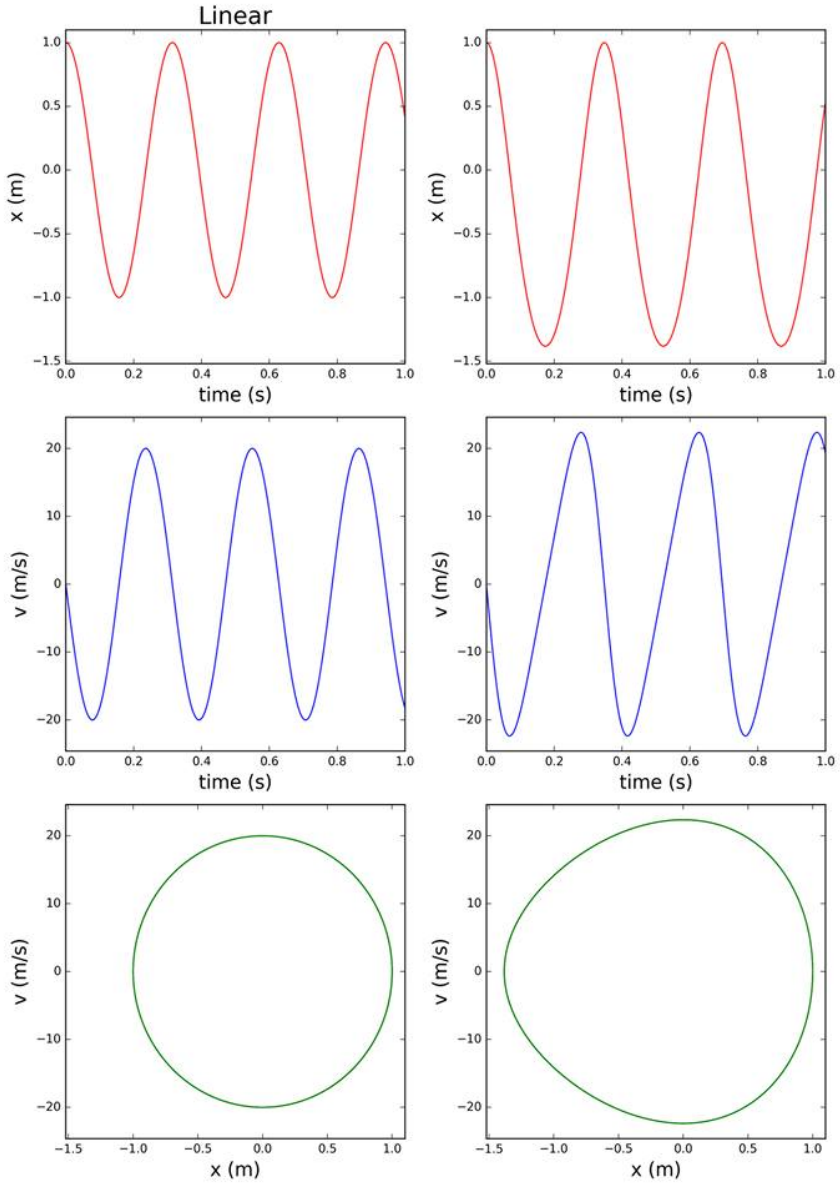
For the case in which k2 = 15, the force had a concave down plot, and thus the magnitude of the force was lower for x > 0 and higher for x < 0. The plots of position and velocity as a function of time showed that the amplitude was higher for the non-linear case, and the frequency was slightly lower. The phase plot of position and velocity was no longer perfectly symmetrical. Instead, it was skewed slightly towards negative values of position, meaning the mass extended further to one side than the other (specifically the negative side).

For the case of k2 = -15, the situation was mostly reversed. The force was concave up, so the force was assymetric about zero (greater for positive displacements). The amplitudes of the position and velocity plots were smaller, and the frequency was lower as well. The phase plot was shifted from negative positions, so it reached less negative values than it did positive (although it did not seem to go any further in the positive direction than the linear oscillator)

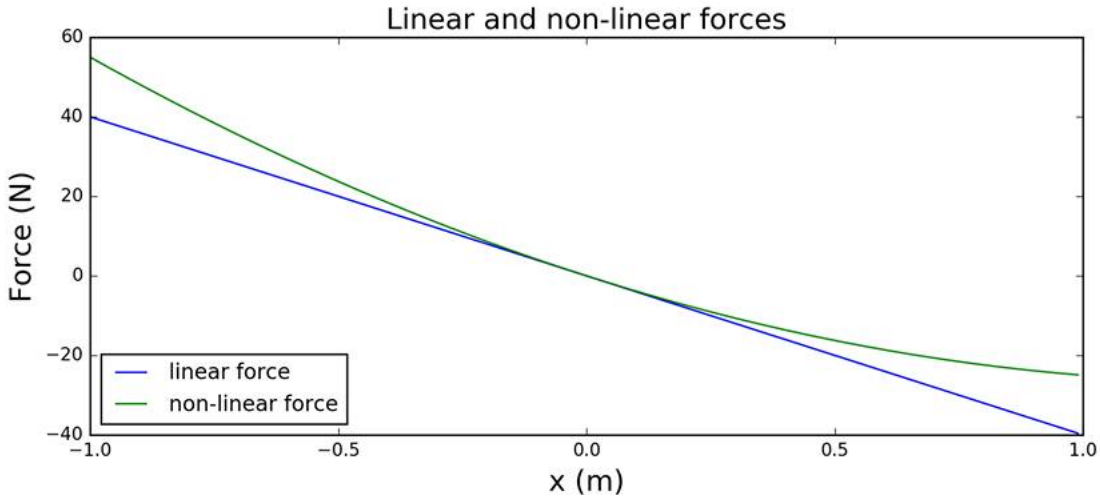
See attached images of plots.

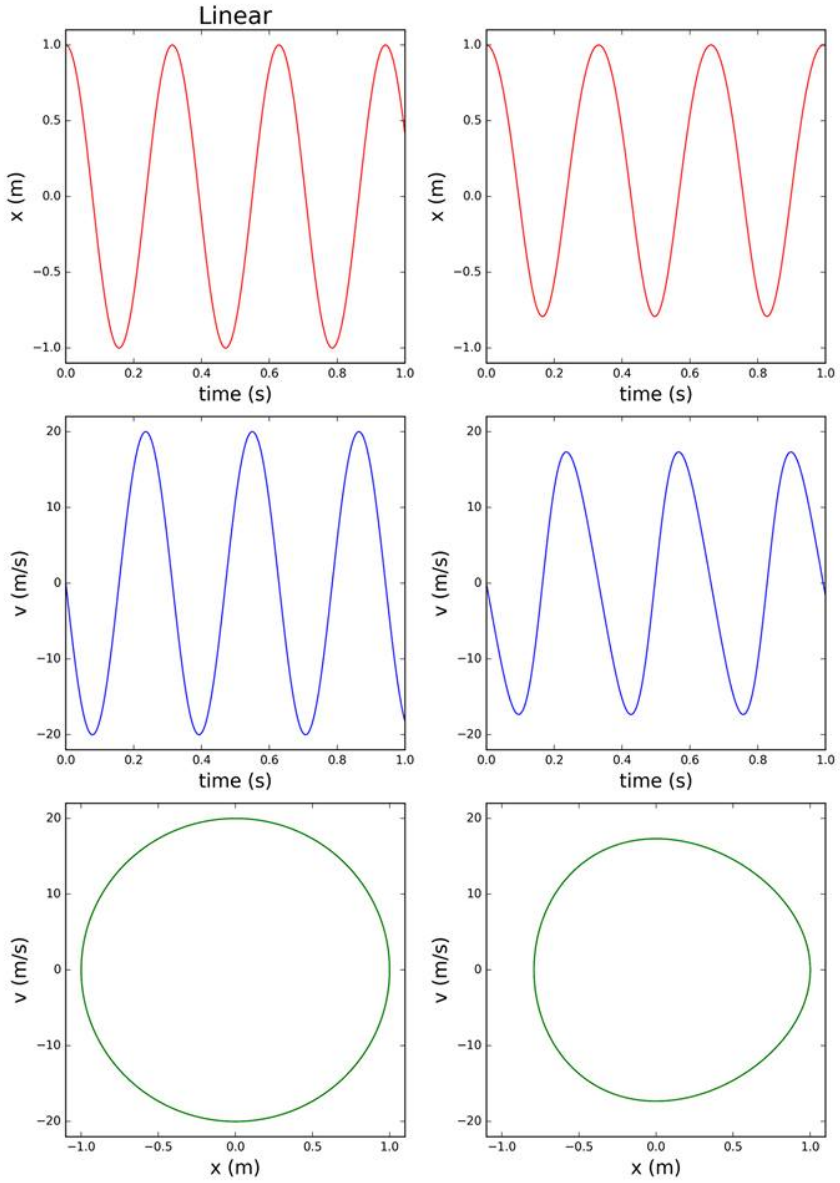
Plots for positive k




Linear
x (m)
time (s)
x (m)
time (s)
v (m/s)
time (s)
v (m/s)
time (s)
v (m/s)
x (m)
v (m/s)
x (m)

Plots for negative k





In [ ]: