Problem 1

In this problem, I modified the chi_square_fit for Quadratic $y = Ax^2$, for use for CO2 Data.

This was done by transforming $y = Ax^2$ into $log(y) = log(A) + Nlog(x)$ in the begining of the chi_square_fit function, and then runing the rest of the function as usual.

Also includes an edited function for loading in the CO2 data from a txt file.

```
import numpy as np
import math
from P1 import chi_square_fit
from matplotlib import pyplot as plt
from read_co2 import read_co2
```

Below is a function to assist with creating a line out of just a slope and intercept.

```
def abline(slope, intercept):
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    return(x_vals, y_vals)
```

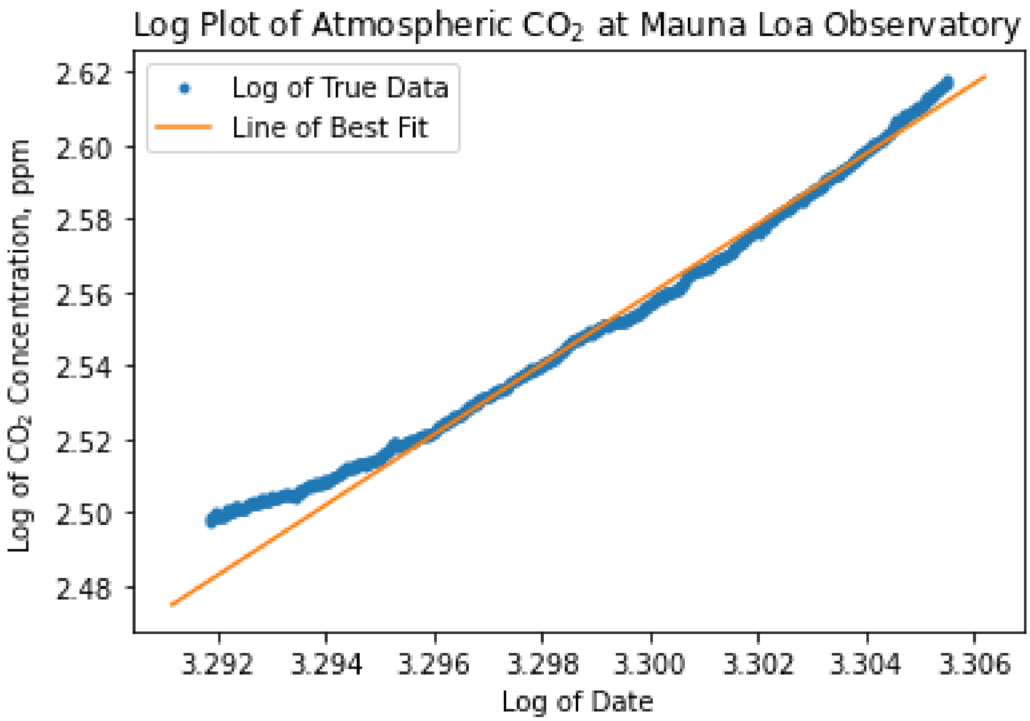
Below I read in the data from the CO2.txt file, and run the modified chi_square_fit function from P1.py.

```
dates,data,err = read_co2("CO2.txt")
```

```
z = chi_square_fit(dates,data,err)
```

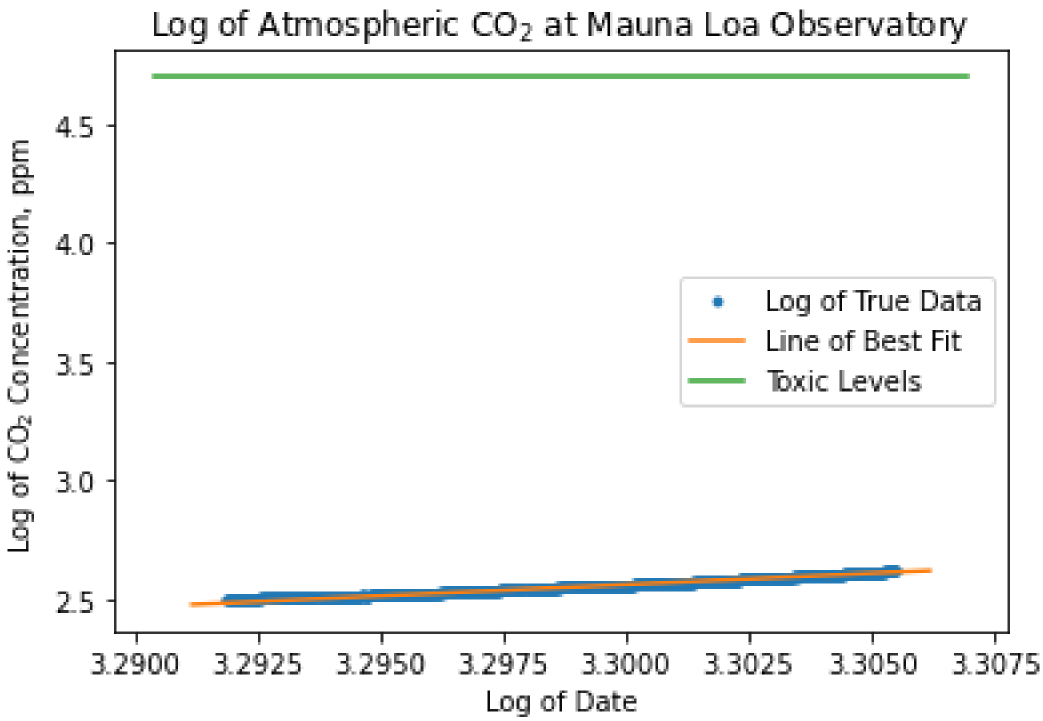
Below, I convert the original data into an equivilant form using the equation $log(y) = log(A) + Nlog(x)$ to linearize it. Next, the output from chi_square_fit is plotted with this linearized direct data.

```
def f(g):
    return math.log(g,10)
def h(yi,j):
    return math.sqrt(1/(yi*math.log(10)))*j

f1 = np.vectorize(f)
f2 = np.vectorize(f)
h1 = np.vectorize(h)

lineDates = f1(dates)
lineData = f2(data)

plt.plot(lineDates,lineData, '.',  label='Log of True Data')
xLine, yLine = abline(z[1],z[0])
plt.plot(xLine,yLine, label = 'Line of Best Fit')

plt.title(r"Log Plot of Atmospheric CO$_{2}$ at Mauna Loa Observatory")
plt.xlabel("Log of Date")
plt.ylabel(r"Log of CO$_{2}$ Concentration, ppm")
plt.legend()


plt.show()
```



Below is the above plots, but also shown is a line demostrating when the atmosphere will become unbreathable due to CO2 levels being above 50,000 Parts Per Million.

```
plt.plot(lineDates,lineData, '.',  label='Log of True Data')
xLine, yLine = abline(z[1],z[0])
plt.plot(xLine,yLine, label = 'Line of Best Fit')

xToxic, yToxic = abline(0,math.log(50000,10))
plt.plot(xToxic,yToxic, label = 'Toxic Levels')

plt.title(r"Log of Atmospheric CO$_{2}$ at Mauna Loa Observatory")
plt.xlabel("Log of Date")
plt.ylabel(r"Log of CO$_{2}$ Concentration, ppm")
plt.legend()


plt.show()
```



Below is the intersection point of the Toxic line with the Best Fit line, and the date this will occur:

```
xi = (math.log(50000,10)-z[0]) / (z[1]-0)
yi = z[1] * xi + z[0]

Point = 10**xi

print(Point)
```

3335.8198967814374

In the year 3335, the air will be unbreathable due to CO2.

Problem 2

For this problem, I ploted the Hubble data, created 9 groups of it and ploted that as well. For both data sets I generated a line of best fit using the least squares algorithm.

```
import numpy as np
import math
from HubbleFit import least_squares
from matplotlib import pyplot as plt
from read_hubble import read_hubble
from read_hubbleGroups import read_hubbleGroups
```

```
The history saving thread hit an unexpected error (DatabaseError('database disk image is malformed')).Hi
```

Below is a function to create a line of the slope and intercept.

```
def abline(slope, intercept):
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    return(x_vals, y_vals)
```

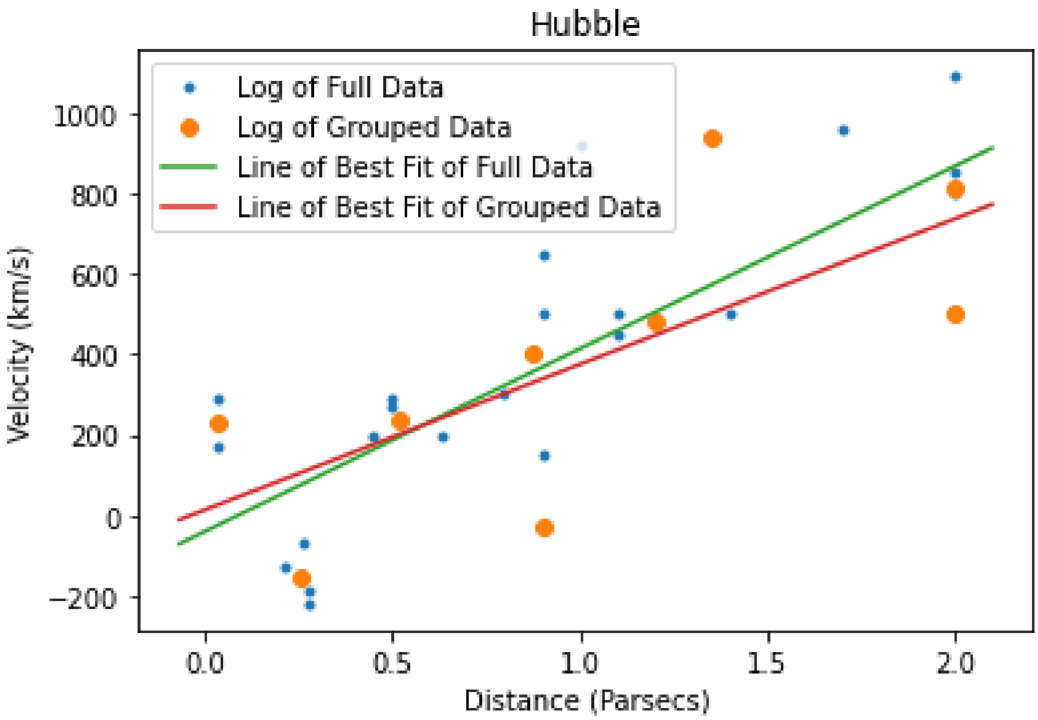
Below, I read in both the full data and the 9 Groups data set, and run a least squares fit on each.

```
distance, velocity = read_hubble("Hubble.txt")
distanceGroups, velocityGroups = read_hubbleGroups("HubbleGroups.txt")
```

```
z = least_squares(distance,velocity)
zGroups = least_squares(distanceGroups,velocityGroups)
```

Below, I plot the full data and its least squares line as well as the 9 grouped data and its best fit line.

```
plt.plot(distance,velocity, '.',  label='Log of Full Data')
plt.plot(distanceGroups,velocityGroups, 'o',  label='Log of Grouped Data')

xLine, yLine = abline(z[1],z[0])
xLineGroups, yLineGroups = abline(zGroups[1],zGroups[0])

plt.plot(xLine,yLine, label = 'Line of Best Fit of Full Data')
plt.plot(xLineGroups,yLineGroups, label = 'Line of Best Fit of Grouped Data')

plt.title("Hubble")
plt.xlabel("Distance (Parsecs)")
plt.ylabel("Velocity (km/s)")
plt.legend()


plt.show()
```



Comparing the slope of the fitted straight line to Hubble's value of K.

```
print("The full data best fit line slope is: ", z[1])
print("The groups best fit line slope is: ", zGroups[1])
print("Hubble constant k was found to be around 500.")
```

```
The full data best fit line slope is:  454.1584409226284
The groups best fit line slope is:  361.73946381986593
Hubble constant k was found to be around 500.
```

Problem 3

In this problem, I modified the chi_square_fit for the Expodential function $y = Ae^x$, for use for CH4 Data.

This was done by transforming $y = Ae^x$ into $ln(y) = ln(A) + x$ in the begining of the chi_square_fit function, and then runing the rest of the function as usual.

Also includes an edited function for loading in the CH4 data from a txt file.

```
import numpy as np
import math
from P3 import chi_square_fit
from matplotlib import pyplot as plt
from read_ch4 import read_ch4
```

Below is a function to assist with creating a line out of just a slope and intercept.

```
def abline(slope, intercept):
    """Plot a line from slope and intercept"""
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    plt.plot(x_vals,y_vals, label = 'Best Fit Line')
    return(x_vals, y_vals)
```

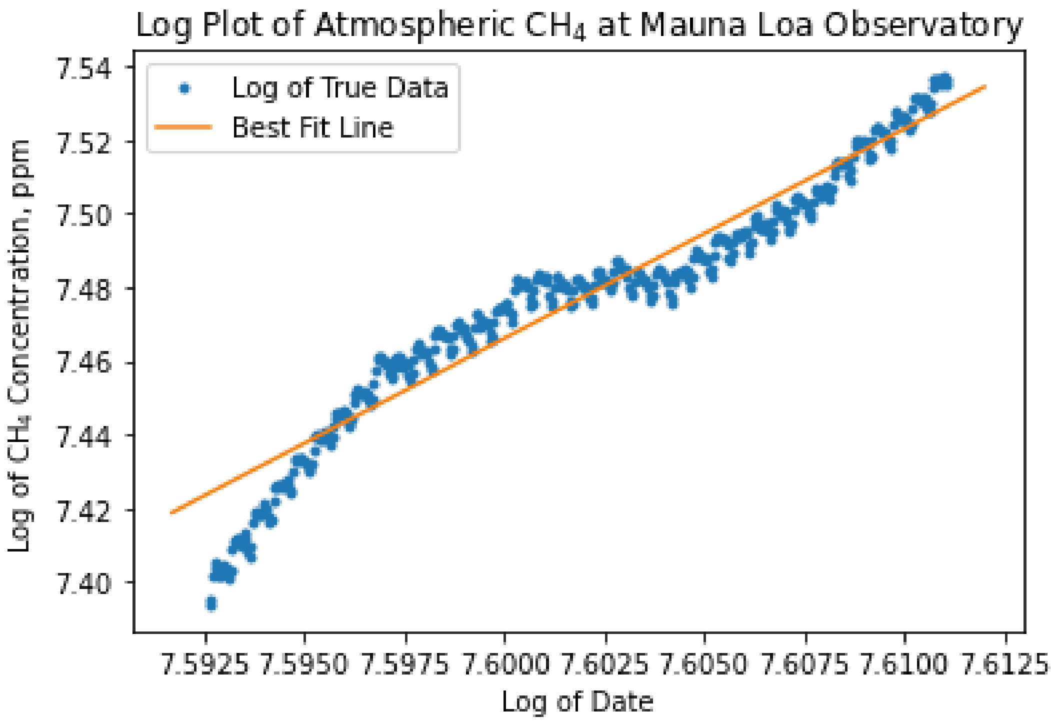
Below I read in the data from the CH4.txt file, and run the modified chi_square_fit function from P3.py.

```
dates,data,err = read_ch4("CH4.txt")
```

```
z = chi_square_fit(dates,data,err)
```

Below, I convert the original data into an equivilant form using the equation $ln(y) = ln(A) + x$ to linearize it. Next, the output from chi_square_fit is plotted with this linearized direct data.

```
def f(g):
    return math.log(g)
def h(yi,j):
    return math.sqrt(1/(yi*math.log(10)))*j

f1 = np.vectorize(f)
f2 = np.vectorize(f)
h1 = np.vectorize(h)

lineDates = f1(dates)
lineData = f2(data)

plt.plot(lineDates,lineData, '.',  label='Log of True Data')
abline(z[1],z[0])

plt.title(r"Log Plot of Atmospheric CH$_{4}$ at Mauna Loa Observatory")
plt.xlabel("Log of Date")
plt.ylabel(r"Log of CH$_{4}$ Concentration, ppm")
plt.legend()


plt.show()
```



The fit is better for the center years to top years, and is ruined a bit by the earlier years far-different slope.