Notebook FakeData.txt FakeData_with_error.txt

Name - Jinal Shah UID - 2019230070

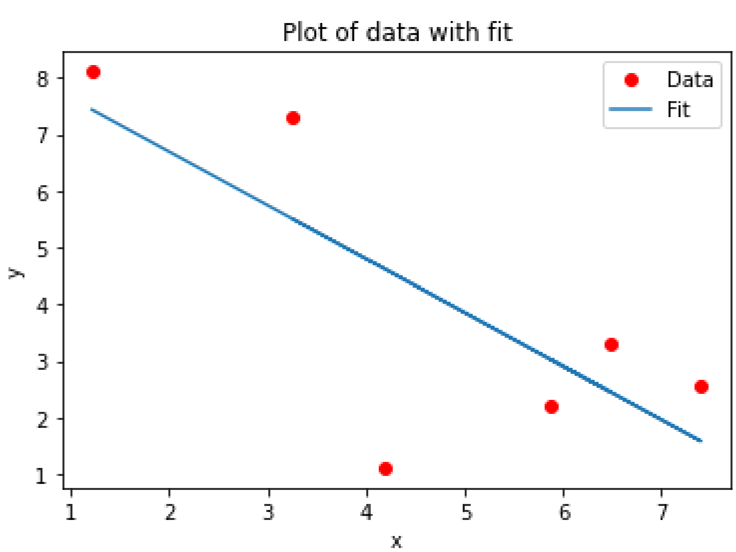
# Exp 4

Aim: To measure fit and error parameters for a model

## 1. Data Fitting

```
from pylab import *
from scipy.optimize import curve_fit
# The first line above is the familar command for importing the entire pylab module. The seco
# Next we have our program read the data from the FakeData.txt file.

xdata,ydata=loadtxt('FakeData.txt',unpack=True)

# This command reads the data from the file FakeData.txt and loads the first column in the fi


# Now we need to define the function we wish to fit. The curvefit() function will actually fi
def linearFunc(x,intercept,slope):
  y = intercept + slope * x
  return y

# The return y statement tells Python to return the value of y whenever the function is calle
# For example, for an intercept of 2, a slope of 3, and x=x= 1, calling linearFunc(1,2,3) giv
linearFunc(1,2,3)
# You can give your function any name you like. I called it linearFunc.

# The next step is to actually do the fit using curvefit().
# We must pass curvefit() the name of the function to fit, the horizontal axis data, and the
# The program returns some arrays containing the best fit parameters and the covariance matri
# matrix to determine the uncertainites in the slope and intercept of the best fit line.

a_fit,cov=curve_fit(linearFunc,xdata,ydata)

# The next two lines assign the best-fit parameters returned by the curve_fit() to the variab
inter = a_fit[0]

slope = a_fit[1]

# Next, the uncertainties in the intercept and slope are computed from the covarience matrix
# to the variables d_inter and d_slope.
d_inter = sqrt(cov[0][0])

d_slope = sqrt(cov[1][1])
```

```
# Create a graph showing the data.

plot(xdata,ydata,'ro',label='Data')

# Compute a best fit y values from the fit intercept and slope.

yfit = inter + slope*xdata

# Create a graph of the fit to the data.

plot(xdata,yfit,label='Fit')

# Display a legend, label the x and y axes and title the graph.

legend()

xlabel('x')

ylabel('y')

title('Plot of data with fit')

# Save the figure to a file

savefig('FakeData.png')

# Show the graph in a new window on the users screen.

show()

# We will use a the print() command to print the best fit parameters and uncertainties. Here
```



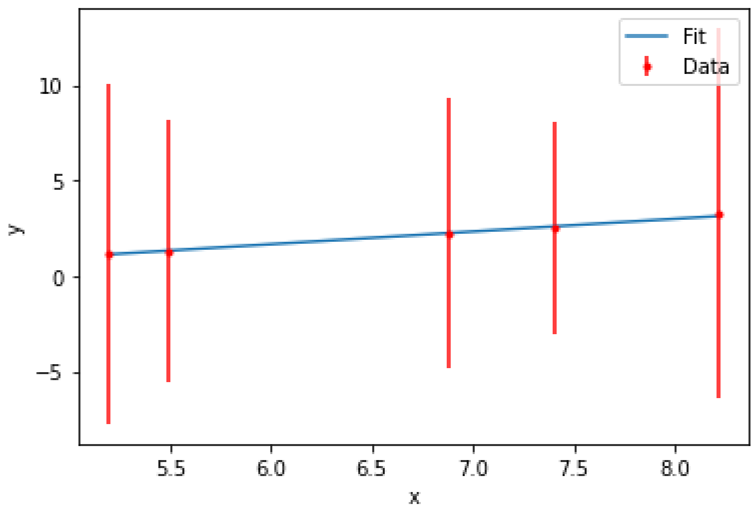
```
# Display the best fit values for the slope and intercept. These print

# statments illustrate how to print a mix of strings and variables.

print(f'The slope = {slope}, with uncertainty {d_slope}')

print(f'The intercept = {inter}, with uncertainty {d_inter}')
```

```
The slope = -0.9455682604628566, with uncertainty 0.4174773565883709
The intercept = 8.579511698507119, with uncertainty 2.163614051105426
```

## 2. Data Fitting with error

```
# Next we have our program read the data from the FakeData_with_error.txt file. It's simple t

xdata,ydata,d_y = loadtxt('FakeData_with_error.txt',unpack=True)

# We pass the uncertainties to curve_fit by adding the argument sigma=d_y to the function cal


a_fit,cov=curve_fit(linearFunc,xdata,ydata,sigma=d_y)

# We get the best fit slope and the uncertainties from a_fit and cov.




inter = a_fit[0]
slope = a_fit[1]
d_inter = sqrt(cov[0][0])
d_slope = sqrt(cov[1][1])

# We use the function errorbar() to plot the data showing error bars on the data points. This

# Create a graph showing the data.

errorbar(xdata,ydata,yerr=d_y,fmt='r.',label='Data')

# Compute a best fit line from the fit intercept and slope.

yfit = inter + slope*xdata

# Create a graph of the fit to the data. We just use the ordinary plot

# command for this.

plot(xdata,yfit,label='Fit')

# Display a legend, label the x and y axes and title the graph.
```

```
# Display a legend, label the x and y axes and title the graph.

legend()
xlabel('x')
ylabel('y')

# Save the figure to a file

savefig('FakeDataPlot_with_error.png')

# Show the graph in a new window on the users screen.

show()

# Now we can display the numerical result.
```




```
print(f'The slope = {slope}, with uncertainty {d_slope}')

print(f'The intercept = {inter}, with uncertainty {d_inter}')

# When we have estimated uncertainties in the data, then we can estimate the goodness of fit

# where for a linear fit y(x)=a+bx For a good fit, χ2r should be approximately equal to one.
```

```
The slope = 0.6656028702881751, with uncertainty 0.03549213604200107
The intercept = -2.3430681719234285, with uncertainty 0.239532487804196
```

```
chisqr = sum((ydata-linearFunc(xdata,inter,slope))**2/d_y**2)

dof = len(ydata) - 2

chisqr_red = chisqr/dof
```

```
print(f'Reduced chi^2 = {chisqr_red}')
```

```
Reduced chi^2 = 1.2633310164063059
```

Double-click (or enter) to edit

Conclusion :

1. In this experiment, i have performed data fitting for the fake data considered here. Data fitting us best fit points are those which are near to the slope.
2. In the second part i performed data fitting for the given data with error. In order to check the fitti Reduced chi^2 value was near 1 so it indicates that the data has been nearly fit. The more the er

✓ 0s completed at 5:20 PM