Lab 11 - Curve Fitting

Paige Brady and Austin Jones

02-24-2020

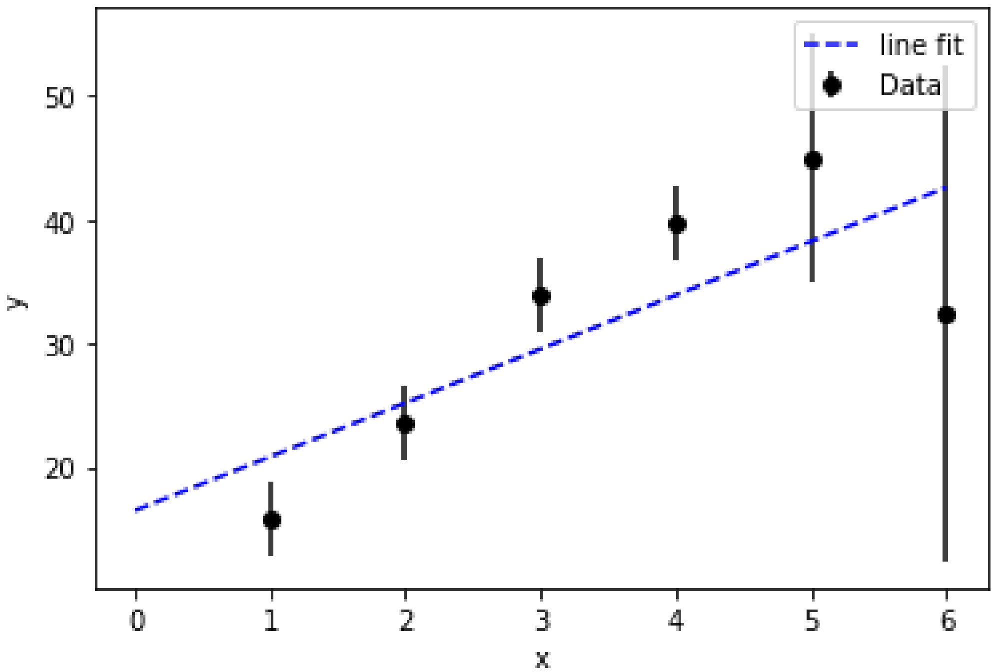
[1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[10]:
```
from scipy import optimize

#define fitting function to be a linear function
def line_func(x, a, b):
    return x*a+b

#Sample data for straight line fine
x_data = np.array([1.0,2.0,3.0,4.0,5.0,6.0])
y_data = np.array([15.9,23.6,33.9,39.7,45.0,32.4])
y_unc = np.array([3.0,3.0,3.0,3.0,10.0,20.0])

#Plotting sample data
plt.errorbar(x_data, y_data,yerr=y_unc,fmt="ko",label="Data")

#initial guesses for parameter values and calculating best fit curve for x_data␣
↪and y_data
guess_a = 1.0
guess_b = 0.0
par, cov = optimize.curve_fit(line_func, x_data, y_data, p0=[guess_a,guess_b])

#Fitted values of a and b
fit_a = par[0]
fit_b = par[1]
print("best fit value of a:  ", fit_a)
print("best fit value of b:  ", fit_b)

#Plotting the best fit line
xf = np.linspace(0.0,6.0,100)
yf = line_func(xf,fit_a,fit_b)
plt.plot(xf, yf,"b--",label="line fit")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
```

```
plt.show()
print("""Jupyter Notebook 11.1: Plot data including error bars and best-fit␣
 ↪function to obtain
a result like that of Fig 11.2 in lab manual version 6. Function is biased␣
 ↪towards poorly measured points.
""")
```

```
best fit value of a:   4.3571428997566874
best fit value of b:   16.499999850851594
```

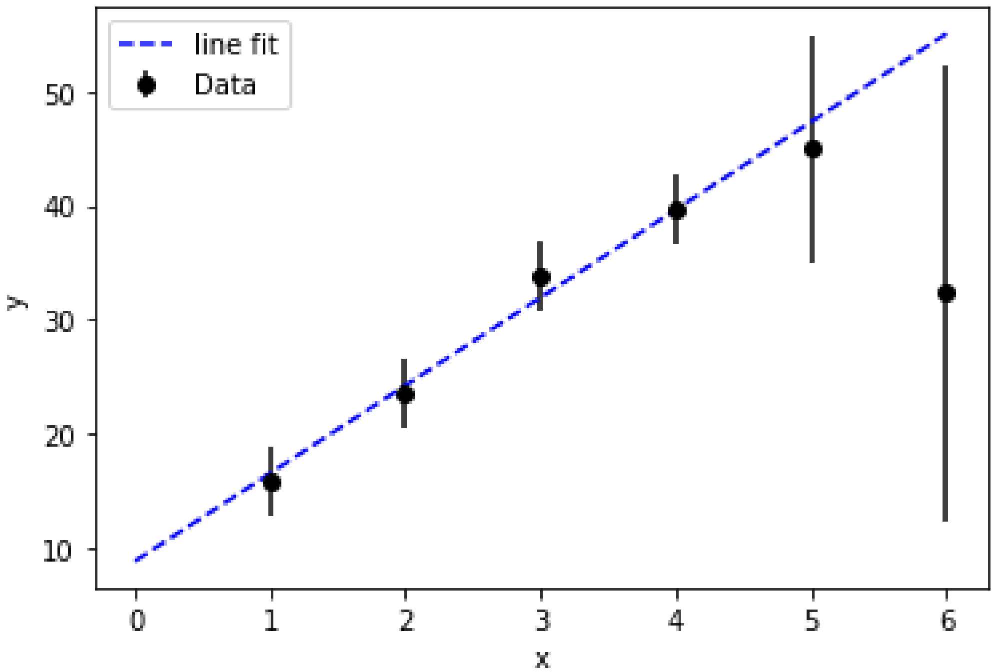



```
Jupyter Notebook 11.1: Plot data including error bars and best-fit function to
obtain
a result like that of Fig 11.2 in lab manual version 6. Function is biased
towards poorly measured points.
```

[11]:
```
#Curve-fit function and optional parameter sigma
par2, cov2 = optimize.curve_fit(line_func, x_data, y_data, sigma=y_unc,␣
 ↪p0=[guess_a,guess_b])

#Sample data for straight line fine
x_data = np.array([1.0,2.0,3.0,4.0,5.0,6.0])
y_data = np.array([15.9,23.6,33.9,39.7,45.0,32.4])
y_unc = np.array([3.0,3.0,3.0,3.0,10.0,20.0])
```

```python
#Plotting sample data
plt.errorbar(x_data, y_data,yerr=y_unc,fmt="ko",label="Data")

#initial guesses for parameter values and calculating best fit curve for x_data␣
 ↪and y_data
guess_a = 1.0
guess_b = 0.0

#Fitted values of a and b
fit_a2 = par2[0]
fit_b2 = par2[1]
print("best fit value of a:  ", fit_a)
print("best fit value of b:  ", fit_b)

#Plotting the best fit line
xf = np.linspace(0.0,6.0,100)
yf = line_func(xf,fit_a2,fit_b2)
plt.plot(xf, yf,"b--",label="line fit")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()

plt.show()
print("""Jupyter Notebook 11.2: Plotting the correct uncertainty y_unc to␣
 ↪produce new plot with data and fit - result
is a fit with no more bias
""")
```

```
best fit value of a:   4.3571428997566874
best fit value of b:   16.499999850851594
```




```
Jupyter Notebook 11.2: Plotting the correct uncertainty y_unc to produce new
plot with data and fit - result
is a fit with no more bias
```

[4]:
```python
def constant_func(x,a):
    return a

#choose y2_data independent of x-values from Gaussian with a mean of 50 and␣
 ↪uncertainty on the y-values to 10
x2_data = np.arange(100)
y2_data = np.random.normal(50.0, 10.0, size=100)

#Best fit constant values
guess_a = 0.0
par3, cov3 = optimize.curve_fit(constant_func, x2_data, y2_data,␣
 ↪p0=[guess_a],absolute_sigma=True)

#Best fit parameter and it's uncertainty
unc = np.sqrt(np.diag(cov3))
fit_a = par3[0]
unc_a = unc[0]

print("""Jupyter Notebook 11.3: Set uncertainty on the y values to 10 and␣
 ↪absolute_sigma = True in the fit. Record the
```

```
uncertainty.
""")


#Printing the results
print("mean of y data: ", np.mean(y2_data))
print("fitted constant: ", fit_a)
print("uncertainty: ", unc_a)
```

```
Jupyter Notebook 11.3: Set uncertainty on the y values to 10 and absolute_sigma
= True in the fit. Record the
uncertainty.

mean of y data:  50.13170251076923
fitted constant:  50.13170251085707
uncertainty:  0.1000000001752797
```

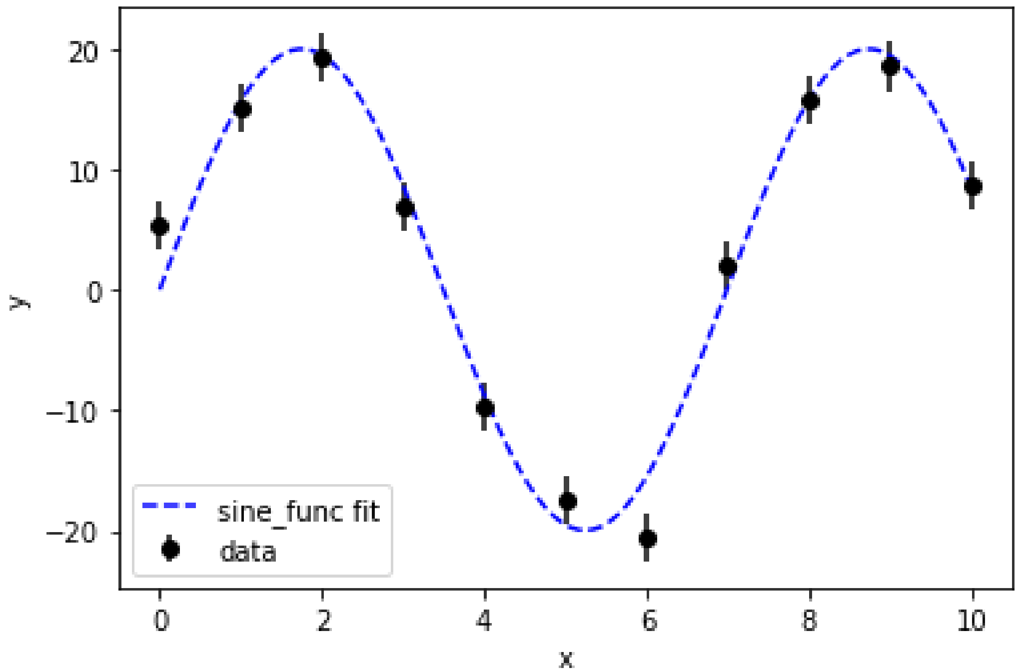
[5]:
```
#fit y3_data to a constant value a:
def constant_func(x,a):
    return a

#choose y3_data independent of x-values from Gaussian with a mean of 50 and␣
↪sigma of 10
x3_data = np.arange(100)
y3_data = np.random.normal(50.0, size=100)

#Best fit constant value
guess_a = 0.0
par4, cov4 = optimize.curve_fit(constant_func, x3_data, y3_data,␣
↪p0=[guess_a],absolute_sigma=True)

#Best fit parameter and it's uncertainty
unc = np.sqrt(np.diag(cov3))
fit_a = par4[0]
unc_a = unc[0]

print("""Jupyter Notebook 11.4: Leave the uncertainties on the y values␣
↪unspecified and absolute_sigma = True in the fit.
""")

#Printing the results
print("mean of y data: ", np.mean(y3_data))
print("fitted constant: ", fit_a)
print("uncertainty: ", unc_a)
```

```
Jupyter Notebook 11.4: Leave the uncertainties on the y values unspecified and
absolute_sigma = True in the fit.
```

```
mean of y data:  50.12609604415761
fitted constant:  50.126096044245415
uncertainty:  0.1000000001752797
```

[6]:
```
def sine_func (x, a, b):
    return a*np.sin(b*x)

#Sample data and setting y uncertainty to sigma = 2
x_sample = np.array([0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0])
y_sample = np.array([5.3,15.0,19.2,6.8,-9.7,-17.4,-20.5,2.1,15.7,18.5,8.6])
y_unc_sine = 2*np.ones(11)

#Plotting the sample data
plt.errorbar(x_sample, y_sample,yerr=y_unc_sine,fmt="ko",label="data")

#initial guesses for parameter values and calculating best fit curve for x_data␣
 ↪and y_data
guess_a = 20.0
guess_b = 2*pi/7
par_sine, cov_sine = optimize.curve_fit(sine_func, x_sample, y_sample,␣
 ↪p0=[guess_a,guess_b],absolute_sigma=True)

#Fitted values of a and b
fit_a_sine = par_sine[0]
fit_b_sine = par_sine[1]
print("best fit value of a:  ", fit_a)
print("best fit value of b:  ", fit_b)

#Plotting the best fit line
xf_sine = np.linspace(0.0,10.0,100)
yf_sine = fit_a_sine*np.sin(fit_b_sine*xf_sine)
plt.plot(xf_sine, yf_sine,"b--",label="sine_func fit")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()

plt.show()
print("""Jupyter Notebook 11.5: Plot sample data listed in lab manual version 6␣
 ↪in section 11.4 with error bars and
best-fit sine wave.
""")
```

```
best fit value of a:   50.126096044245415
best fit value of b:   16.499999850851594
```



Jupyter Notebook 11.5: Plot sample data listed in lab manual version 6 in section 11.4 with error bars and
best-fit sine wave.

```
[ ]:
```