EGR 103L – Fall 2019

Laboratory 7 - Curve Fitting

Marcus Deans (md374)
EGR103L, Section 4, Tuesdays 11:45-2:35
10 November, 2019

I have adhered to the Duke Community Standard in completing this assignment. I understand that a violation of the Standard can result in failure of this assignment, failure of this course, and/or suspension from Duke University.

Contents

List of Figures

1 Chapra 14.5

Case	Equation	S_t	S_r	r^2	s	r
y vs. x	$y = 0.359x + 4.89$	5.410e01	5.795	8.930e-01	8.510e-01	9.450e-01
x vs. y	$x = 2.49y - 11.1$	3.745e02	4.012e01	8.930e-01	2.239	9.450e-01

The two models, effectively being reversed version of one another, yielded similar results. The coefficient of determination was identical for both at 8.930e-01, demonstrating that both models had equal correlation to the actual data values. This showed the equality between the two models in that both were equally "good" at predicting the results. The Sum of Residual Squares S_t and Sum of Squares of Estimate Residuals S_r were different for the two models. It was noted that the difference was primary by one order of magnitude, with the values for x vs. y being approximately ten times larger than the respective value for the y vs. x model. This had the result of a substantially larger standard error of the estimate for the x vs. y plot, with $s_{x/y}$ again being ten times larger than $s_{y/x}$. This would suggest that the case of y vs. x is a superior model in minimizing the magnitude of these statistical measures.

2 Chapra 14.7

S_t	S_r	r^2	R, N m mol^{-1} K^{-1} or J mol^{-1} K^{-1}
2.455e07	8.018e03	9.997e-01	8.3162 J mol^{-1} K^{-1}

As per the Committee on Data for Science and Technology (CODATA), the value of the Gas Constant R is 8.3145 J mol^{-1} K^{-1}. Therefore, the estimate calculated via the program was highly accurate, yielding a result accurate to the hundredths place. While the difference between the two is nonetheless relevant, particularly for large calculations, it shows the viability of the polyfit method used to calculate this result. The percent accuracy, calculated by $\frac{(experimental-ideal)}{ideal} * 100$ found that the percent error of the calculation was 0.02%. Again, this shows the calculation was quite accurate.

3 Chapra 14.27

Case	Equation	S_t	S_r	r^2	$i(3.5)$, A
Polyfit	$i = 2.81x - 0.592$	3.380e02	2.920e-01	9.99e-01	9.240
Genlin	$i = 2.71x$	3.380e02	2.400	9.930e-01	9.830

When the intercept was set to zero in the second model, thereby reducing the equation to a single term, the predictive power in the model declined in that the coefficient of determination decreased from 0.999 to 0.993. This difference, albeit small relative to the overall value, still demonstrates the difference that this term creates and the constantly increasing accuracy with a higher number of terms. The intercept provided an "anchor" for the line that comprised the rest of the graph, and allowed the polyfit model to better fit the data. The current prediction for 3.5V changed similarly, with the polyfit model yielding a value of 9.24 as opposed to 9.83. This difference of >0.5 Amperes is somewhat substantial and would cause substantial impact on the given circuit. Overall, this exercise reinforced the veracity of using a higher-term model to fit the graph. Also, the decision was made to increase the range of the x-model values to comprise values from 0 to 10 in order to better illustrate the "zero-intercept" nature of one model as opposed to the polyfit model.

4 Chapra 15.10

Based on the least squares fit of the given model,

$$p(t) = 4.14e^{-1.5t} + 2.90e^{-0.3t} + 1.53e^{-0.05t}$$

The statistical and information required by the problem is:

S_t	S_r	r^2
2.040e01	8.000e-02	9.960e-01

This is a mathematically sound model in that the overall organism count (comprised of the sum of each individual organism's growth rate) is closely modeled by the equation, with a coefficient of determination of 0.996. While there is some room for improvement, potentially with higher order terms or different modeling methods, this is overall an effective modeling method that accurately represents the situation.

5 Chapra 15.10 Alternate

Based on the least squares fit of the given model,

$$p(t) = 5.63e^{-1.5t} - 4.43e^{-0.3t} + 7.89e^{-0.2t}$$

The statistical and specific information required by the problem is:

S_t	S_r	r^2
2.040e01	8.000e-02	9.960e-01

Mathematically, this is an effective model as previously described, particularly above regarding Chapra 15.10. The coefficient of determination is equal to the that of Chapra 15.10's model at 0.996. However, this model is scientifically *inaccurate*, given that one of the organisms is shown with a negative concentration, which is impossible. Such cases satisfiy the mathematical model but do not take into account the facts on the ground and the physical existence of organisms on the plate.

6 Chapra 15.11

The model equation is:

$$P = P_m \frac{I}{I_{sat}} e^{-\frac{I}{I_{sat}}-1} = 238.71 \frac{I}{221.82} e^{-\frac{I}{221.82}-1}$$

with P_m measured in (mg m^{-3} d^{-1}) and I_{sat} measured in (μE m^{-2} s^{-1}). The statistical and information required by the problem is:

S_t	S_r	r^2
2.837e04	1.116e03	9.610e-01

For the initial guess, I started with a completely random choice based roughly on values that seemed to be consistent with the provided data; ie. values that would "fit in" by being within the range of the x-values of a given pair, and also within the y-values of that same pair. This yielded a guess of [200,200]. The graph was created, and based on the observed peak of the graph, the estimate was revised to be closer to the visual result. I realized that the graph had indeed been able to yield its complete (ie. fully accurate in the scope of the model and predicting as best possible given the parameters) value based on the initial estimate, and the revised estimate was not in fact necessary. Mathematically, this model was sound with its high coefficient of determination of 0.961, representing high correlation between the actual values and the model and showing its efficacy as a predictor.

7 Chapra 15.18

The model equation is:

$$P = \frac{RT}{V} + \frac{RTA_1}{V^2} + \frac{RTA_2}{V^3}$$

Data table:

Using mL for V				Using L for V					
Initial Guesses		Estimates		Initial Guesses		Estimates		Estimates (converted)	
A_1 (mL)	A_2 (mL)^2	A_1 (mL)	A_2 (mL)^2	A_1 (L)	A_2 (L)^2	A_1 (L)	A_2 (L)^2	A_1 (mL)	A_2 (mL)^2
0	0	-237.92	0	0	0	-0.23056	-0.12389	-230.56	-12389e1
1e2	0	-237.92	0	1e-01	0	-0.23056	-0.12389	-230.56	-12389e1
0	1e2	-230.56	-12389e1	0	1e-4	-0.23056	-0.12389	-230.56	-12389e1
1e2	1e2	-230.56	-12389e1	1e-01	1e-4	-0.23056	-0.12389	-230.56	-12389e1

I would choose the A_1 and A_2 values obtained with an initial guess of $1e-01L$, $1e-04L$. The resulting values appear to be the most consistent with the rest of the calculations. Specifically, the A_1 value is common not only with all of the calculations in litres (including A_1 guesses of both 0 and $1e-01$), but also the same as the calculated A_1 in two of the mL based calcualtions, when $A_2 = 1e2$. Furthermore, the calculated A_2 value for the initial guess of $1e-01$, $1e-04$ was again the same as for the rest of the L based calculations. It bears marked resemblance to the A_2 value found in the mL calcualtion, when the initial guess for A_2 was $1e2$. However, that calculation is different by several orders of magnitude. More investigation would be necessary to ascertain which initial guess is indeed correct, going beyond simple relative comparisons. The values for the calculations vary due to the use of nonlinear regression, which is premised upon an initial guess that considerably affects the result. Therefore, the optimization procedure ascertains a different mathematical model depending on where the model "starts at", which is inputted as the initial guess.

A Codes

A.1 Chapra 14.5

```
"""
[Chapra 14.5]
[Marcus Deans]
[3 November 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]

"""

import numpy as np
import matplotlib.pyplot as plt
from fitting_common import make_plot
from fitting_common import calc_stats

x = [0,2,4,6,9,11,12,15,17,19]
y = [5,6,7,6,9,8,8,10,12,12]

cal1 = np.polyfit(x,y,1)
est1 = np.polyval(cal1,x)
st1, sr1, r21 = calc_stats(y,est1,False)
syx = np.sqrt(sr1/8)
xmodel = np.linspace(0,19, 100)
ymodel1 = np.polyval(cal1,xmodel)

cal2 = np.polyfit(y, x,1)
est2 = np.polyval(cal2,y)
st2, sr2, r22 = calc_stats(x,est2,False)
sxy = np.sqrt(sr2/8)
ymodel2 = np.linspace(5,12,100)
xmodel2 = np.polyval(cal2,ymodel2)

fig , ax = make_plot(x,y,est1,xmodel,ymodel1,1)
ax.set(xlabel = "X-Values", ylabel="Y-Values")
fig.tight_layout()
ax.grid(False)
fig.savefig("Y vs. X.png")
fig.savefig("Y vs. X.eps")

fig , ax = make_plot(y,x,est2,ymodel2,xmodel2,2)
ax.set(xlabel = "Y-Values", ylabel="X-Values")
fig.tight_layout()
ax.grid(False)
fig.savefig("X vs. Y.png")
fig.savefig("X vs. Y.eps")

print("Y vs. X:")
print(cal1)
print("St - Sum of Residual Squares: {:.3e}".format(st1))
print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr1))
print("S(y/x) - Standard Error of Estimate y/x: {:.3e}".format(syx))
print("R Squared Value: {:.3e}".format(r21))
```

```
54 print("R Value: {:.3e}".format(np.sqrt(r21)))
55 print("\n")
56 print("X vs. Y:")
57 print(cal2)
58 print("Sum of Residual Squares: {:.3e}".format(st2))
59 print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr2))
60 print("Standard Error of Estimate x/y: {:.3e}".format(sxy))
61 print("R Squared Value: {:.3e}".format(r22))
62 print("R Value: {:.3e}".format(np.sqrt(r22)))
```

A.2 Chapra 14.7

```
"""
[Chapra 14.7]
[Marcus Deans]
[3 November 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]

"""
import matplotlib.pyplot as plt
import numpy as np
from fitting_common import calc_stats
from fitting_common import make_plot

#fixed volume of 1kg of nitrogen, volume is 10 m^3
#one mole of gas is 28g
#PV = nRT
# P = nRT/V
# P = nR/V * T
# P = R(1000/28*10) * T
#slope is R10/28
n = 1000/28
v = 10
t = [233,273,313,353,393,433]
p = [6900, 8100, 9350, 10500, 11700, 12800]

def yfun(xe, coefs):
    return coefs[0]*xe

xv = np.reshape(t,(-1,1))
yv = np.reshape(p,(-1,1))
phi_mat = np.block([[(xv**1)]])
pvec = np.linalg.lstsq(phi_mat, yv, rcond=None)[0]
# print(pvec)
# print(pvec[0])
xmodel = np.linspace(233,433, 100)
yhat = yfun(t,pvec)
ymodel = yfun(xmodel, pvec)

st, sr, r2 = calc_stats(p,yhat,0)

fig, ax = make_plot(t,p, yhat,xmodel,ymodel,1)
ax.set(xlabel = "Temperature in Degrees Kelvin, $K$", ylabel="Pressure in Newtons
    per Square Metre, N/$m^2$")
fig.tight_layout()
ax.grid(False)
fig.savefig("Pressure vs. Temperature.png")
fig.savefig("Pressure vs. Temperature.eps")

print("St - Sum of Residual Squares: {:.3e}".format(st))
print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr))
print("R Squared Value: {:.3e}".format(r2))
print("Value of R in equation: ")
print(pvec[0]*(28/100))
```

A.3 Chapra 14.27

```
"""
[Chapra 14.27]
[Marcus Deans]
[3 November 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]

"""
import matplotlib.pyplot as plt
import numpy as np
from fitting_common import calc_stats

x = [2,3,4,5,7,10]
y = [5.2, 7.8, 10.7, 13, 19.3, 27.5]
xmodel = np.linspace(0,10, 100)

def yfun(xe, coefs):
    return coefs[0]*xe

# Reshape data for block matrices
xv = np.reshape(x, (-1, 1))
yv = np.reshape(y, (-1, 1))
phi_mat = np.block([[xv ** 1]])
pvec = np.linalg.lstsq(phi_mat, yv, rcond=None)[0]

# %% Generate estimates and model
zeroest = yfun(x, pvec)
st2, sr2, r22 = calc_stats(y, zeroest,False)
zeromod = yfun(xmodel,pvec)

polycoef = np.polyfit(x,y,1)
polyest = np.polyval(polycoef,x)
st1, sr1, r21 = calc_stats(y,polyest,False)
polymod = np.polyval(polycoef,xmodel)

fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x, y, "ko", label="Data")
ax.plot(xmodel, polymod, "b-", label="Polyfit Model")
ax.plot(xmodel, zeromod, "r--", label="Zero-Intercept Model")
ax.set(xlabel = "Potential Difference in Volts, $V$", ylabel="Current in Amperes,
    $A$")
ax.grid(False)
ax.legend(loc="best")
fig.tight_layout()
fig.savefig("Polyfit vs. Zero-Intercept.png")
fig.savefig("Polyfit vs. Zero-Intercept.eps")

# val=1
print("Polyfit Model Statistics")
print("St - Sum of Residual Squares: {:.3e}".format(st1))
print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr1))
print("R Squared Value: {:.3e}".format(r21))
print(st1)
```

```
55 print(sr1)
56 print(r21)
57 print("The equation for the polyfit model uses the coefficients: ")
58 print(polycoef)
59 print("Zero-Intercept Model Statistics")
60 print("St - Sum of Residual Squares: {:.3e}".format(st2))
61 print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr2))
62 print("R Squared Value: {:.3e}".format(r22))
63 print(st2)
64 print(sr2)
65 print(r22)
66 print("The equation for the zero-intercept model uses the coefficient: ")
67 print(pvec)
68 # val = float(yfun(3.5,pvec))
69 print("The prediction of the current at 3.5V with the polyfit model is {:.3f}".
       format(np.polyval(polycoef,3.5)))
70 print("The prediction of the current at 3.5V with the zero-intercept model is {:.3f}
       ".format(float(yfun(3.5,pvec))))
```

A.4 Chapra 15.10

```
"""
[Chapra 15.10]
[Marcus Deans]
[3 November 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]

"""

import numpy as np
import matplotlib.pyplot as plt
from fitting_common import calc_stats

x = np.array([0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 9.0])
y = np.array([6.0, 4.4, 3.2, 2.7, 2.0, 1.9, 1.7, 1.4, 1.1])

# Ae^(-1.5t) + Be^(-0.3t) + Ce^(-0.05t)

xmodel = np.linspace(0.5,9.0,100)
def yfun(xe, coefs):
    return coefs[0]*(np.exp(-1.5*xe)) + coefs[1]*(np.exp(-0.3*xe)) + coefs[2]*(np.
    exp(-0.05*xe))
def getA(xe, coefs):
    return coefs[0]*(np.exp(-1.5*xe))
def getB(xe, coefs):
    return coefs[1]*(np.exp(-0.3*xe))
def getC(xe, coefs):
    return coefs[2]*(np.exp(-0.05*xe))
# Reshape data for block matrices
xv = np.reshape(x, (-1, 1))
yv = np.reshape(y, (-1, 1))
phi_mat = np.block([[np.exp(-1.5*xv), np.exp(-0.3*xv),np.exp(-0.05*xv)]])
# print(phi_mat)
pvec = np.linalg.lstsq(phi_mat, yv, rcond=None)[0]
# print(pvec)

# %% Generate estimates and model
yhat = yfun(x, pvec)
ymodel = yfun(xmodel, pvec)
Amodel = getA(xmodel, pvec)
Bmodel = getB(xmodel, pvec)
Cmodel = getC(xmodel, pvec)

# %% Calculate statistics
st, sr, r2 = calc_stats(y, yhat, 0)

# %% Generate plots
fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x, y, "o", color = "magenta", label="Data")
ax.plot(xmodel, ymodel, "k-", label="Model")
ax.plot(xmodel, Amodel, "b-", label="A")
ax.plot(xmodel, Bmodel, "g-", label="B")
ax.plot(xmodel, Cmodel, "r-", label="C")
```

```
55 ax.set(xlabel = "Time in Days", ylabel="Concentration in Organisms/$cm^3$", title="
       Models of Growth Rates of Seaweed on Plate")
56 ax.grid(True)
57 ax.legend(loc="best")
58 fig.tight_layout()
59 fig.savefig("Time vs. Concentration Alpha.jpg")
60 fig.savefig("Time vs. Concentration Alpha.eps")
61
62 print("Statistics")
63 # print("y = {0:.3f}$e^-1.5t$ + {0:.3f}$e^-0.3t$ + {0:.3f}$e^-0.2t$".format(pvec[0],
       pvec[1],pvec[2]))
64 print("A")
65 print(pvec[0])
66 print("B")
67 print(pvec[1])
68 print("C")
69 print(pvec[2])
70 print("St - Sum of Residual Squares: {:.3e}".format(st))
71 print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr))
72 print("R Squared Value: {:.3e}".format(r2))
73 print("\n")
```

A.5 Chapra 15.10 Alternate

```
"""
[Chapra 15.10]
[Marcus Deans]
[3 November 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]

"""

import numpy as np
import math as m
import matplotlib.pyplot as plt
from fitting_common import calc_stats

x = np.array([0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 9.0])
y = np.array([6.0, 4.4, 3.2, 2.7, 2.0, 1.9, 1.7, 1.4, 1.1])

# Ae^(-1.5t) + Be^(-0.3t) + Ce^(-0.05t)

xmodel = np.linspace(0.5,9.0,100)
def yfun(xe, coefs):
    return coefs[0]*(np.exp(-1.5*xe)) + coefs[1]*(np.exp(-0.3*xe)) + coefs[2]*(np.
    exp(-0.2*xe))
def getA(xe, coefs):
    return coefs[0]*(np.exp(-1.5*xe))
def getB(xe, coefs):
    return coefs[1]*(np.exp(-0.3*xe))
def getC(xe, coefs):
    return coefs[2]*(np.exp(-0.2*xe))
# Reshape data for block matrices
xv = np.reshape(x, (-1, 1))
yv = np.reshape(y, (-1, 1))
phi_mat = np.block([[np.exp(-1.5*xv), np.exp(-0.3*xv),np.exp(-0.2*xv)]])
# print(phi_mat)
pvec = np.linalg.lstsq(phi_mat, yv, rcond=None)[0]
# print(pvec)

# %% Generate estimates and model
yhat = yfun(x, pvec)
ymodel = yfun(xmodel, pvec)
Amodel = getA(xmodel, pvec)
Bmodel = getB(xmodel, pvec)
Cmodel = getC(xmodel, pvec)

# %% Calculate statistics
st, sr, r2 = calc_stats(y, yhat, 0)

# %% Generate plots
fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x, y, "o", color = "magenta", label="Data")
ax.plot(xmodel, ymodel, "k-", label="Model")
ax.plot(xmodel, Amodel, "b-", label="A")
ax.plot(xmodel, Bmodel, "g-", label="B")
```

```
55 ax.plot(xmodel, Cmodel, "r-", label="C")
56 ax.set(xlabel = "Time in Days", ylabel="Concentration in Organisms/$cm^3$", title="
       Alternate Models of Growth Rates of Seaweed on Plate")
57 ax.grid(True)
58 ax.legend(loc="best")
59 fig.tight_layout()
60 fig.savefig("Time vs. Concentration Bravo.jpg")
61 fig.savefig("Time vs. Concentration Bravo.eps")
62
63 print("Statistics")
64 # print("y = {0:.3f}$e^-1.5t$ + {0:.3f}$e^-0.3t$ + {0:.3f}$e^-0.2t$".format(pvec[0],
       pvec[1],pvec[2]))
65 print("A")
66 print(pvec[0])
67 print("B")
68 print(pvec[1])
69 print("C")
70 print(pvec[2])
71 print("St - Sum of Residual Squares: {:.3e}".format(st))
72 print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr))
73 print("R Squared Value: {:.3e}".format(r2))
74 print("\n")
```

A.6 Chapra 15.11

```
"""
[Chapra 15.11]
[Marcus Deans]
[5 November 2019]
I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
import numpy as np
import matplotlib.pyplot as plt
from fitting_common import calc_stats
import scipy.optimize as opt

x = np.array([50,80,130,200,250,350,450,550,700])
y = np.array([99,177,202,248,229,219,173,142,72])
xmodel = np.linspace(50,700, 100)

# %% Perform calculations
def yfun(x, *coefs):
    return coefs[0]*(x/coefs[1])*np.exp((-x/coefs[1])+1)
    # return coefs[0] * x ** 1 + coefs[1] * x ** 0

popt = opt.curve_fit(yfun, x, y, [250, 200])[0]
# print(popt)

# %% Generate estimates and model
yhat = yfun(x, *popt)
ymodel = yfun(xmodel, *popt)

# %% Calculate statistics
st, sr, r2 = calc_stats(y, yhat, 0)

fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x, y, "rD", label="Data")
ax.plot(xmodel, ymodel, "k-", label="Model")
ax.set(xlabel = "Optimal Solar Radiation $I_{sat}$", ylabel="Maximum Photosynthesis
    Rate $P_m$", title="Solar Radiation vs. Photosynthesis Rate")
fig.tight_layout()
fig.savefig("Solar Radiation vs. Photosynthesis Rate.jpg")
fig.savefig("Solar Radiation vs. Photosynthesis Rate.eps")

print("Statistics")
print("St - Sum of Residual Squares: {:.3f}".format(st))
print("Sr - Sum of Squares of Estimate Residuals: {:.3f}".format(sr))
print("R Squared Value: {:.3f}".format(r2))
print("\n")
print("Maximum Photosynthesis Rate: ")
print(popt[0])
print("Optimal Solar Radiation: ")
print(popt[1])
```

A.7 Chapra 15.18

```
"""
[Chapra 15.18]
[Marcus Deans]
[5 November 2019]
I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
import numpy as np
from fitting_common import calc_stats
import scipy.optimize as opt
import matplotlib.pyplot as plt

y = np.array([0.985,1.108,1.363,1.631])
# x = np.array([25000.0,22200.0,18000.0,15000.0])
x = np.array([25.0,22.2,18.0,15.0])
# xmodel = np.linspace(14000.0,26000.0,100)
xmodel = np.linspace(15.0,25.0,100)
"""
(PV/RT) = 1 + A1/V + A2/v^2
P/RT = 1/v + A1/V^2 + A2/V^3
P = RT/v + RTA1/v^2 + RTA2/V^3
"""
# r = 82.05
r = 0.08205
t = 303.0
# %% Perform calculations
def yfun(x, *coefs):
    return ((r*t)/x) + ((r*t*coefs[0])/x**2) + ((r*t*coefs[1])/x**3)

popt = opt.curve_fit(yfun, x, y, [0.1, 0.0001])[0]
# print(popt)

# %% Generate estimates and model
yhat = yfun(x, *popt)
ymodel = yfun(xmodel, *popt)

# %% Calculate statistics
st, sr, r2 = calc_stats(y, yhat, 0)

print("Coefficients:")
print(popt[0])
print(popt[1])

print("Statistics")
print("St - Sum of Residual Squares: {:.3e}".format(st))
print("Sr - Sum of Squares of Estimate Residuals: {:.3e}".format(sr))
print("R Squared Value: {:.3e}".format(r2))
print("\n")

fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x, y, "rs", label="Data")
ax.plot(xmodel, ymodel, "k-", label="Model")
ax.set(xlabel = "Volumes in Litres, $mL$", ylabel="Pressure in Atmospheres, atm")
fig.tight_layout()
```

```
56 fig.savefig("Test Graph.eps")
57 fig.savefig("Test Graph.jpg")
```

B Figures

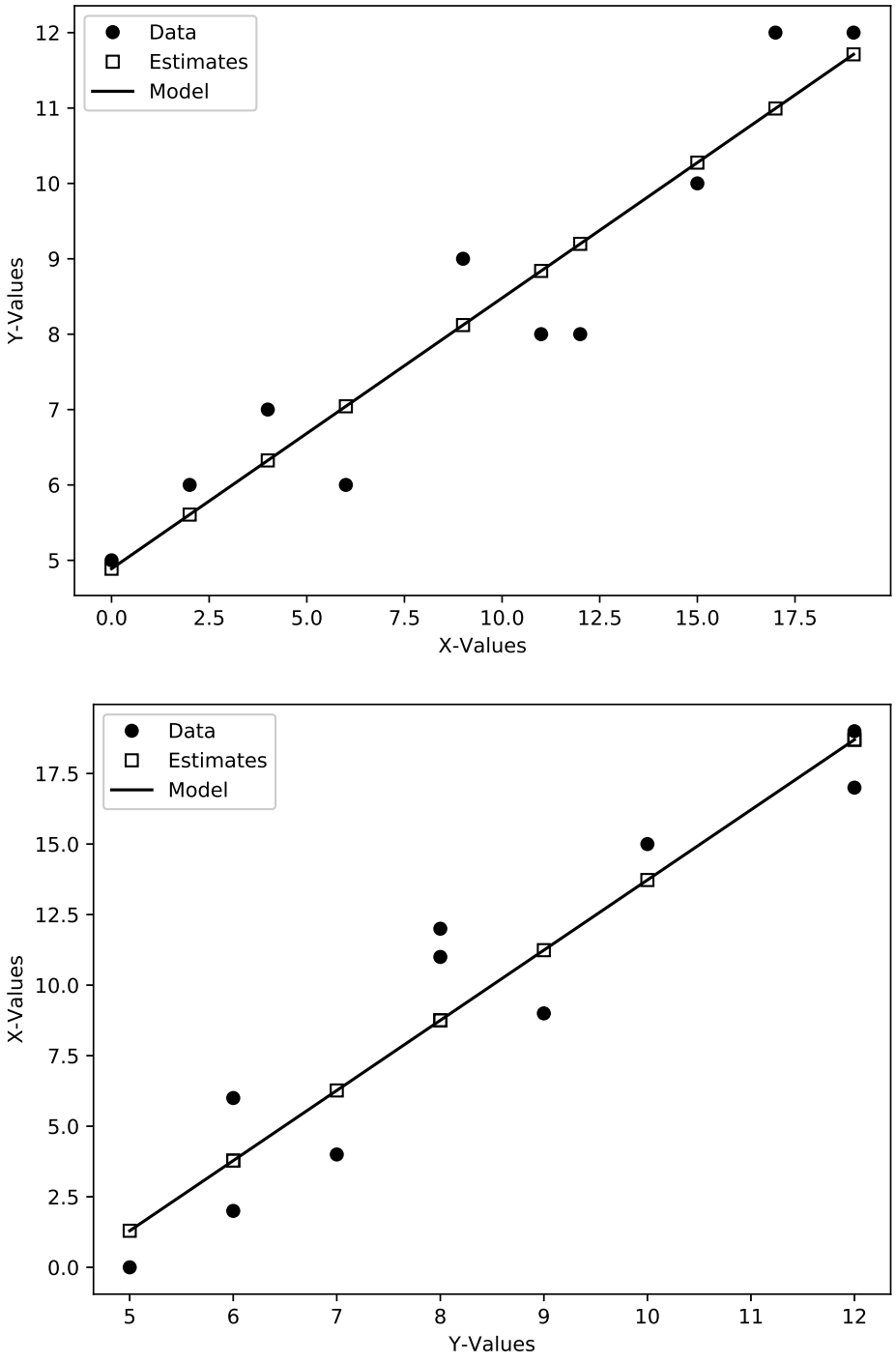


Figure 1: Chapra 14.5

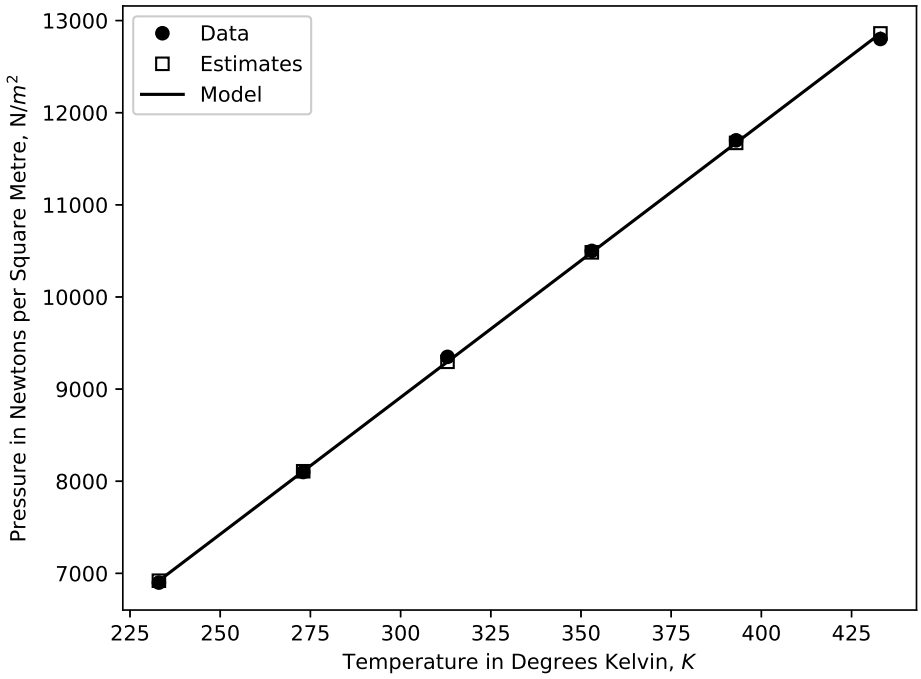


Figure 2: Chapra 14.7

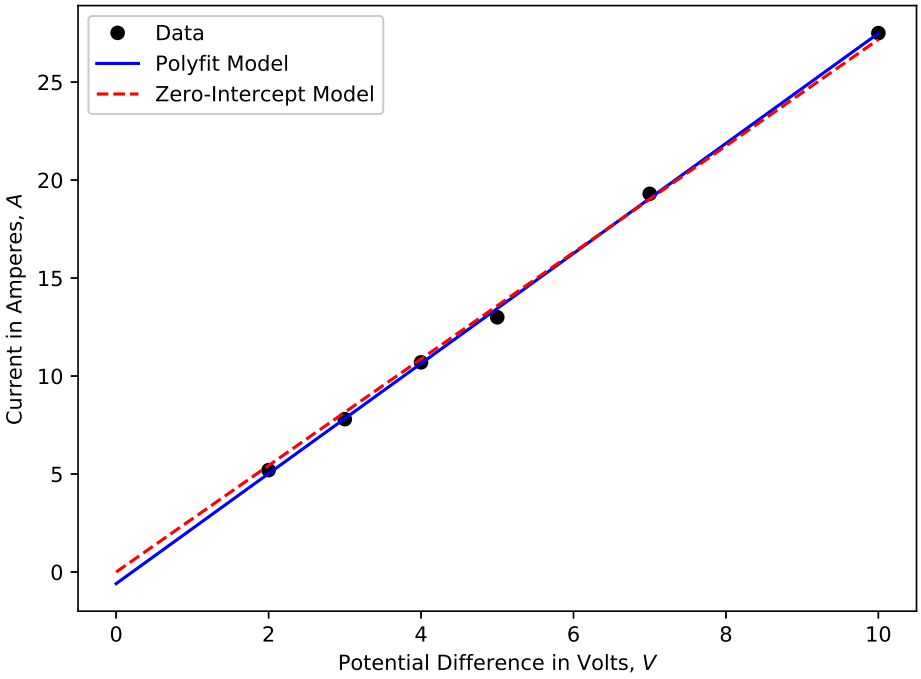
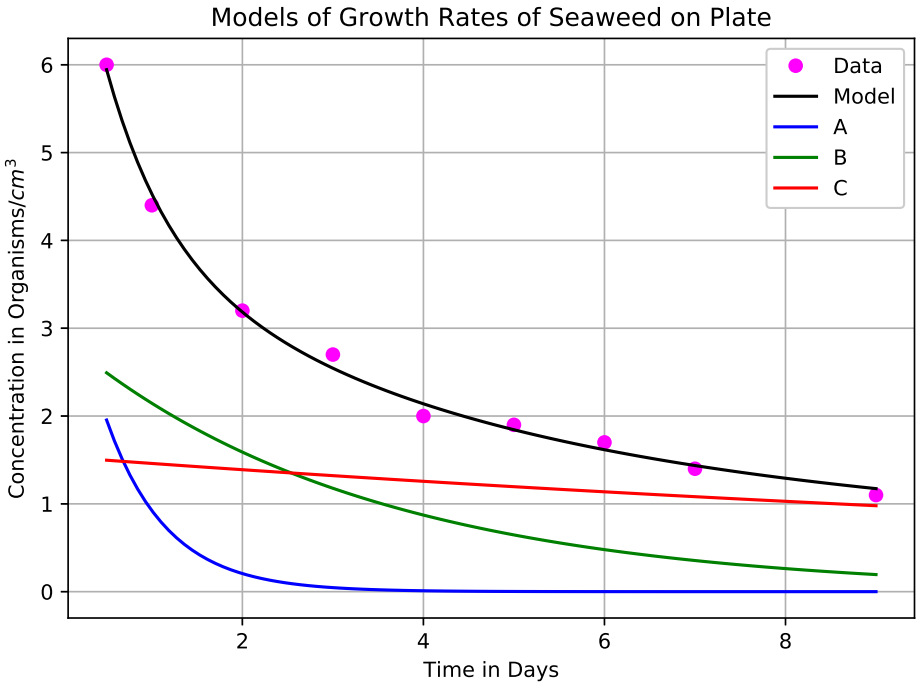


Figure 3: Chapra 14.27



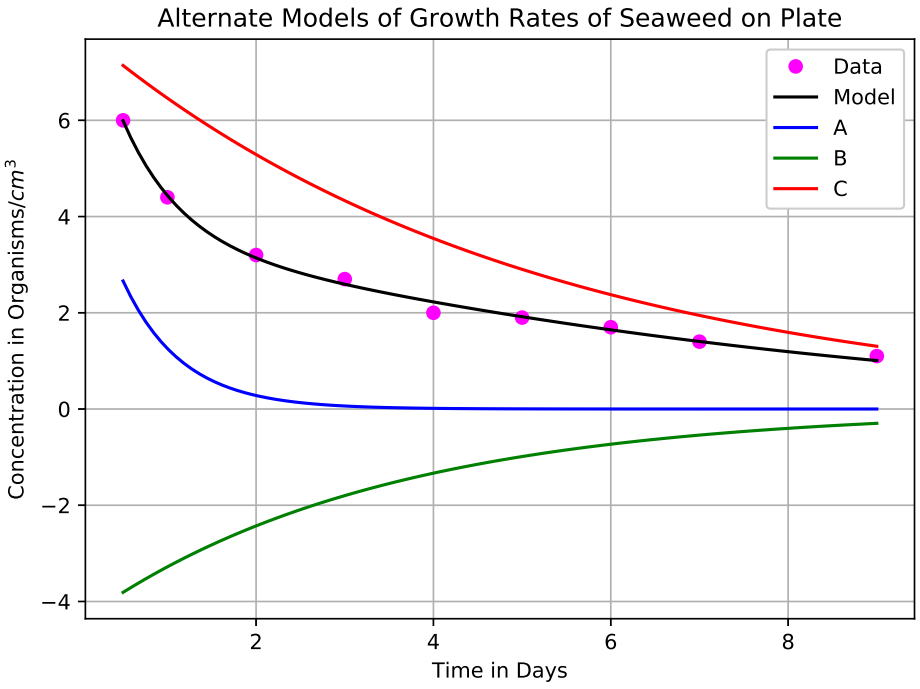


Figure 4: Chapra 15.10 and 15.10 Alternate

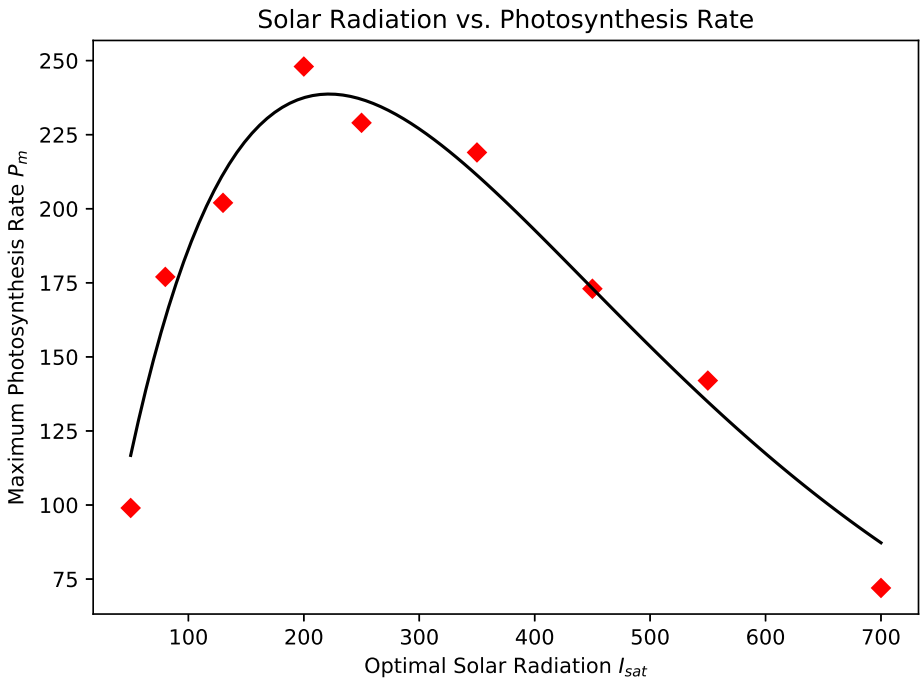


Figure 5: Chapra 15.11

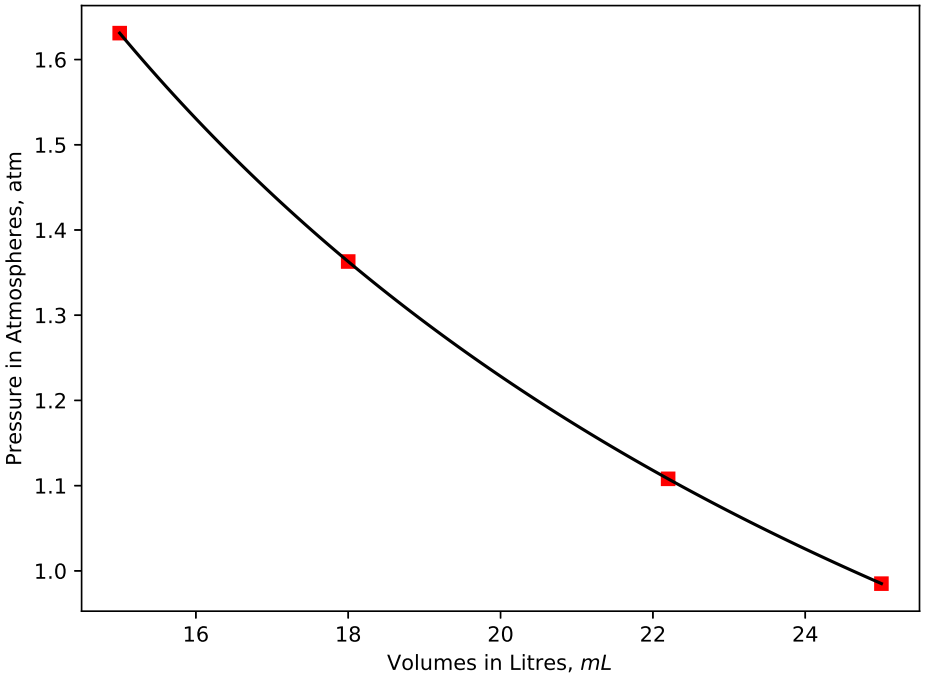


Figure 6: Chapra 15.18