# -\*- coding: utf-8 -\*-

"""

Created on Tue Feb 6 01:18:06 2018

@author: Jiaqi Li

"""

from scipy import \*

from scipy import linalg

thetas = array([65.197917, 66.404320, 87.040195, 141.356044]) # directly observed values

A = zeros((5,4))

A[0:4,0:4] = eye(4)

A[4,0:4] = 1

b = zeros((5,1))

b[0:4,0] = thetas

b[4] = 360.

W = zeros((5,5))

W[0:5,0:5] = eye(5); W[0,0] = 3; W[1,1] = 3; W[2,2] = 3

xbest = linalg.solve(A.T.dot(W).dot(A), A.T.dot(W).dot(b)) #

print("Corrections, in arc-seconds:\n")

print((xbest.T - thetas)\*60\*\*2)

# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

from scipy import array, linspace

from scipy import integrate

from matplotlib.pyplot import \*

import numpy

A = bowditchtxt

At = numpy.transpose(A)

b = conttxt

x = numpy.linalg.inv(At.dot(A)).dot(At).dot(b)

print('\nvalue of d, t, p, n, i')

print(x)

dtpni = numpy.transpose(x)

answer = A.dot(dtpni)-b

book = [0.595, 0.537, 0.693, 1.154, 0.352]

y = A.dot(numpy.transpose(book))-b

figure(1)

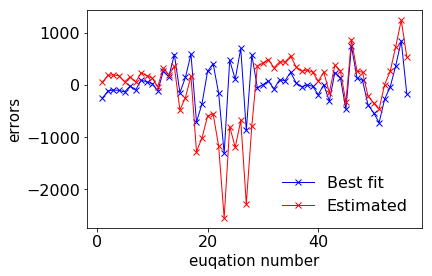
t = linspace(1,56,56)

plot(t,answer,'bx-',t,y,'rx-',linewidth=1)

legend(['Best fit', 'Estimated'], loc=0, framealpha=0)

xlabel('euqation number', fontsize=15)

ylabel('errors', fontsize=15)



value of d, t, p, n, i

[ 0.37647194 -0.41332842 0.3782158 -1.05968075 -0.49088953]

# -\*- coding: utf-8 -\*-

"""

Created on Mon Feb 5 01:57:42 2018

@author: Jiaqi Li

"""

from scipy import \*

from scipy import linalg

from matplotlib.pyplot import \*

import numpy

from astropy.table import Table

deer = deertxt

t = Table(rows=deer, names=('Mean Body Mass', 'Mean Antler Mass'))

print(t)

bw = deer[:,0]

aw = deer[:,1]

A = matrix([[1,1,1,1,1,1,1,1,1,1],[log(bw[0]),log(bw[1]),log(bw[2]),log(bw[3]),log(bw[4]),log(bw[5]),log(bw[6]),log(bw[7]),log(bw[8]),log(bw[9])]])

A = A.T

b = matrix([[log(aw[0]),log(aw[1]),log(aw[2]),log(aw[3]),log(aw[4]),log(aw[5]),log(aw[6]),log(aw[7]),log(aw[8]),log(aw[9])]])

b = b.T

Plaw = linalg.solve(A.T.dot(A), A.T.dot(b))

k = exp(Plaw[0,0])

y = k\*bw\*\*(Plaw[1,0])

#I found that the Arrhenius Law fit the data well

A2 = matrix([[1,1,1,1,1,1,1,1,1,1],[1/bw[0],1/bw[1],1/bw[2],1/bw[3],1/bw[4],1/bw[5],1/bw[6],1/bw[7],1/bw[8],1/bw[9]]])

A2 = A2.T

b2 = matrix([[log(aw[0]),log(aw[1]),log(aw[2]),log(aw[3]),log(aw[4]),log(aw[5]),log(aw[6]),log(aw[7]),log(aw[8]),log(aw[9])]])

b2 = b2.T

Alaw = linalg.solve(A2.T.dot(A2), A2.T.dot(b2))

y2 = exp(Alaw[0,0]+Alaw[1,0]/bw)

print('\n fitted parameter values')

print(y)

figure(1)

plot(bw, aw, 'bo')

plot(bw, y, 'ro-')

legend(['Data','Best fit (Power Law)'], loc=0, framealpha=0.)

xlabel('Body Weigth (kg)', fontsize=15)

ylabel('Antler Weight (kg)', fontsize=15)

figure(2)

plot(bw, aw, 'bo')

plot(bw, y2, 'ro-')

legend(['Data','Arrhenious Law'], loc=0, framealpha=0.)

xlabel('Body Weigth (kg)', fontsize=15)

ylabel('Antler Weight (kg)', fontsize=15)

Mean Body Mass Mean Antler Mass

-------------- ----------------

74.4 1.64

93.4 2.03

110.4 3.16

130.6 3.96

148.9 4.78

170.7 6.21

191.1 7.28

211.8 8.91

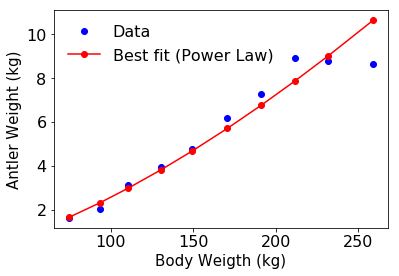
231.7 8.79

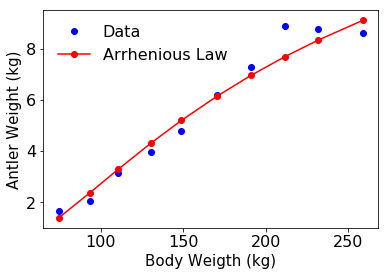
259.1 8.63

fitted parameter values

[ 1.66328127 2.33276098 2.99143667 3.84072106 4.66783663

5.71964556 6.76528754 7.88344493 9.00990247 10.63934108]





# -\*- coding: utf-8 -\*-

"""

Created on Sun Feb 4 18:52:39 2018

@author: Jiaqi Li

"""

from scipy import \*

from scipy import linalg

from matplotlib.pyplot import \*

import numpy

r = stringtxt[:,0]

W = stringtxt[:,1]

A = matrix([[1,1,1,1,1,1,1,1,1,1,1,1]])

A = A.T

b = matrix([[log(W[0])-r[0],log(W[1])-r[1],log(W[2])-r[2],log(W[3])-r[3],log(W[4])-r[4],log(W[5])-r[5],log(W[6])-r[6],log(W[7])-r[7],log(W[8])-r[8],log(W[9])-r[9],log(W[10])-r[10],log(W[11])-r[11]]])

b = b.T

Elaw = linalg.solve(A.T.dot(A), A.T.dot(b))

y = exp(Elaw[0])\*exp(r)

figure(1)

plot(r, W, 'ro')

plot(r, y, 'bo')

legend(['Data','Fit'], loc=0, framealpha=0.)

xlabel('Revolution', fontsize=15)

ylabel('Weight (oz)', fontsize=15)

figure(2)

semilogy(r, y, 'ro-', r, W, 'bo-', linewidth=2)

legend(['Data','Fit'], loc=0, framealpha=0.)

xlabel('Revolution', fontsize=15)

ylabel('Weight (oz)', fontsize=15)

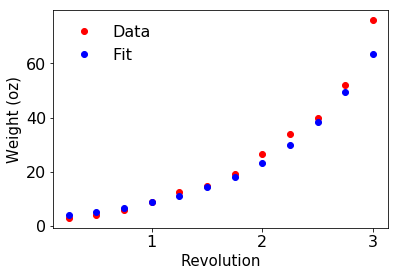
figure(3)

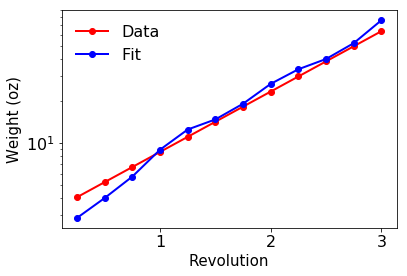
loglog(r, W, 'o', r, y, 'r-')

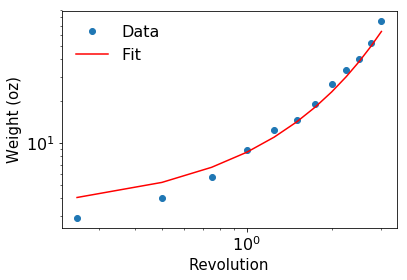
legend(['Data','Fit'], loc=0, framealpha=0.)

xlabel('Revolution', fontsize=15)

ylabel('Weight (oz)', fontsize=15)







# -\*- coding: utf-8 -\*-

"""

Created on Wed Feb 7 02:32:23 2018

@author: Jiaqi Li

"""

from scipy import array, linspace

from scipy import integrate

from matplotlib.pyplot import \*

#60 %

wid1 = linspace(0, 15, 256)

s1 = 100.7

int1 = 436.2

TBL1 = [(s1\*i + int1)/100 for i in wid1]

#65%

wid2 = linspace(0, 15, 256)

s2 = 106

int2 = 390

TBL2 = [(s2\*i + int2)/100 for i in wid2]

x\_0 = 12

y\_60 = (s1\*x\_0+int1)/100

y\_65 = (s2\*x\_0+int2)/100

#The predicted body length of Titanoboa in meter

figure(1)

plot(wid1, TBL1, 'b')

plot(wid2, TBL2, 'r')

plot(x\_0,y\_60,'o',x\_0,y\_65,'o')

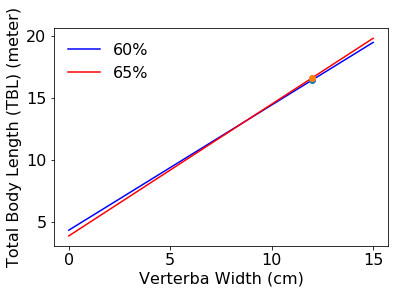
legend(['60%', '65%'], loc=0, framealpha=0.)

xlabel('Verterba Width (cm)')

ylabel('Total Body Length (TBL) (meter)')

print('\nlength due to 60% and 65% extreme values')

print(y\_60,y\_65)



length due to 60% and 65% extreme values

16.446 16.62

# -\*- coding: utf-8 -\*-

"""

Created on Mon Feb 5 01:16:54 2018

@author: Jiaqi Li

"""

from numpy import \*

from scipy import integrate,cos,pi

from matplotlib.pyplot import \*

r1 = log(2.)/1.75

dose = 500\*2

V\_blood = 5000. # 5 liters approximately

mmg\_per\_mg = 1000.

def model1(X, t):

return array([ 100\*(1 - cos(2\*pi\*X[0])/6) - r1\*X[0] ])

x0 = array([dose])

t = linspace(0, 48, 1000)

x = integrate.odeint(model1, x0, t)

figure(1)

plot(t, x[:,0], 'b-', linewidth=3)

ylabel('Plasma content (mg)', fontsize=15)

xlabel('Time (hours from initial dose)', fontsize=15)

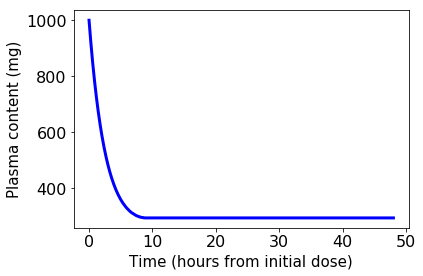
figure(2)

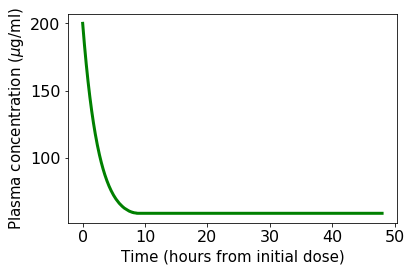
plot(t, x[:,0]\*mmg\_per\_mg/V\_blood, 'g-', linewidth=3, label='Model')

ylabel('Plasma concentration ($\mu$g/ml)', fontsize=15)

xlabel('Time (hours from initial dose)', fontsize=15)

tight\_layout()





# -\*- coding: utf-8 -\*-

"""

Created on Mon Feb 5 01:00:13 2018

@author: Jiaqi Li

"""

from numpy import \*

from scipy import integrate, cos, pi

from matplotlib.pyplot import \*

from astropy.table import Table

r1 = 0.4

r0 = 0.15

m01 = 2.8

dose = 500\*2

V\_blood = 5000.

def f(X, t):

return array([-(r0+m01)\*X[0] + 100\*(1 - cos(2\*pi\*X[0]/6)), \

m01\*X[0] - r1\*X[1] ])

x0 = array([dose, 0.])

t = linspace(0, 48, 1000)

x = integrate.odeint(f, x0, t)

figure(1)

plot(t, x[:,0], 'b-', linewidth=3)

ylabel('Stomach content (mg)')

xlabel('Time (hours from initial dose)')

figure(2)

plot(t, x[:,1]\*1000./V\_blood, 'g-', linewidth=3)

ylabel('Plasma concentration ($\mu$g/ml)')

xlabel('Time (hours from initial dose)')

row\_data = [(x[624,0], x[624,1])]

t = Table(rows=row\_data, names=('Acetaminophen in gut', 'Plasma concentration'))

print('\nThe following is the Acetaminophen in guy and body\n')

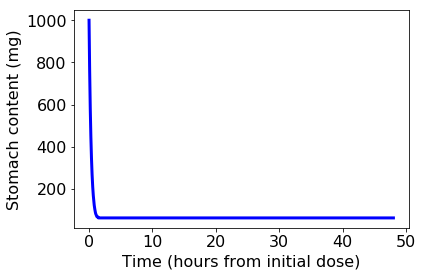
print(t)

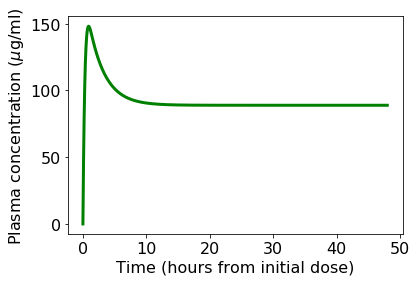
The following is the Acetaminophen in guy and body

Acetaminophen in gut Plasma concentration

-------------------- --------------------

63.4867876972 444.410355504





# -\*- coding: utf-8 -\*-

"""

Created on Sat Feb 3 21:04:18 2018

@author: Jiaqi Li

"""

from scipy import array, linspace

from scipy import integrate

from matplotlib.pyplot import \*

def ok(x, t, sigma, c, p, epsilon):

I = x[0]

V = x[1]

S = array([c\*sigma\*V/p-sigma\*I, (1-epsilon)\*p\*I-c\*V])

return S

def main():

# set up our initial conditions

I0 = 9.2\*10\*\*4

V0 = 10\*\*6

x0 = array([I0, V0])

# Parameters

c = 4.6

sigma = 0.35

epsilon = 0.95

p = 50

# choose the time's we'd like to know the approximate solution

t = linspace(0, 14, 100)

# and solve

x = integrate.odeint(ok, x0, t, args=(sigma,c,p,epsilon))

#Plot I(t) and V (t) on separate subplots using ’semilogy()’

figure(1)

subplot(2,1,1)

semilogy(t, x[:,0], 'b-')

ylabel('infected cells')

subplot(2,1,2)

semilogy(t, x[:,1], 'r-')

xlabel('hours')

ylabel('virus')

figure(2)

plot(x[:,0], x[:,1], 'k-')

xlabel('infected cells')

ylabel('virus')

main()

