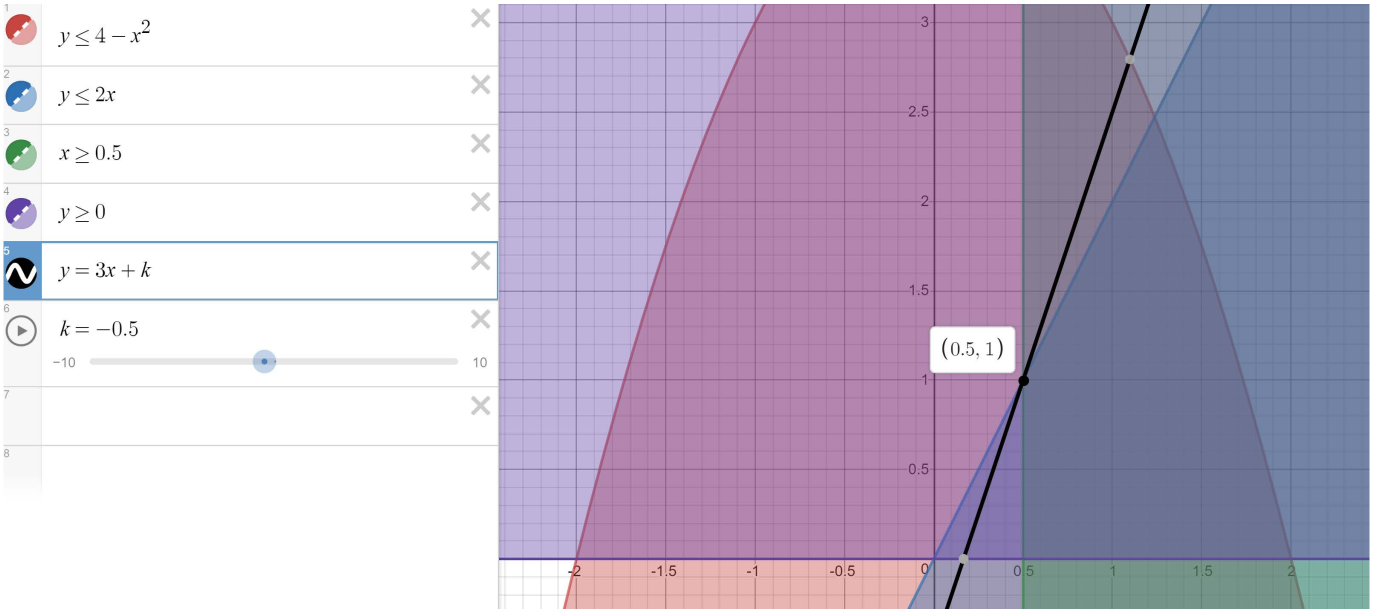
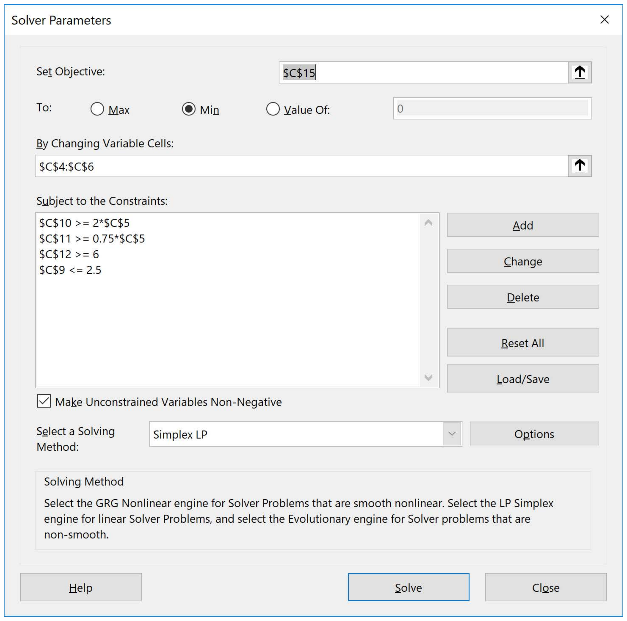
Jacob Needham
A01874339

Problem 1



1. $y \leq 4 - x^2$
2. $y \leq 2x$
3. $x \geq 0.5$
4. $y \geq 0$
5. $y = 3x + k$
6. $k = -0.5$

Problem 2



Solver Parameters

Set Objective: $C$15

To: ○ Max ● Min ○ Value Of: 0

By Changing Variable Cells:
$C$4:$C$6

Subject to the Constraints:
$C$10 >= 2*$C$5
$C$11 >= 0.75*$C$5
$C$12 >= 6
$C$9 <= 2.5

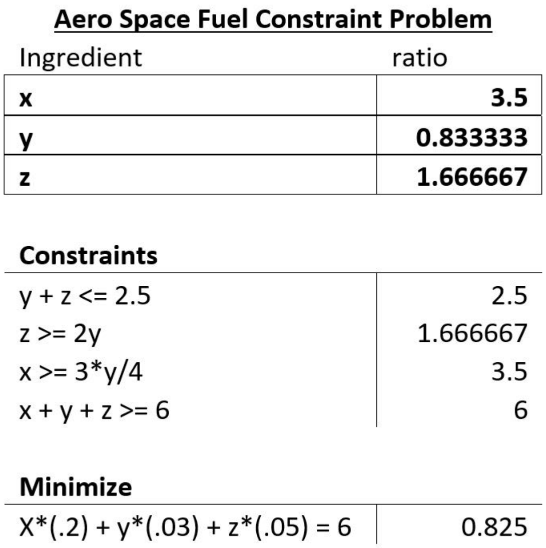
☑ Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

×
Add
Change
Delete
Reset All
Load/Save
Options
Help
Solve
Close



Aero Space Fuel Constraint Problem

Ingredient	ratio
x	**3.5**
y	**0.833333**
z	**1.666667**

Constraints

y + z <= 2.5	2.5
z >= 2y	1.666667
x >= 3*y/4	3.5
x + y + z >= 6	6

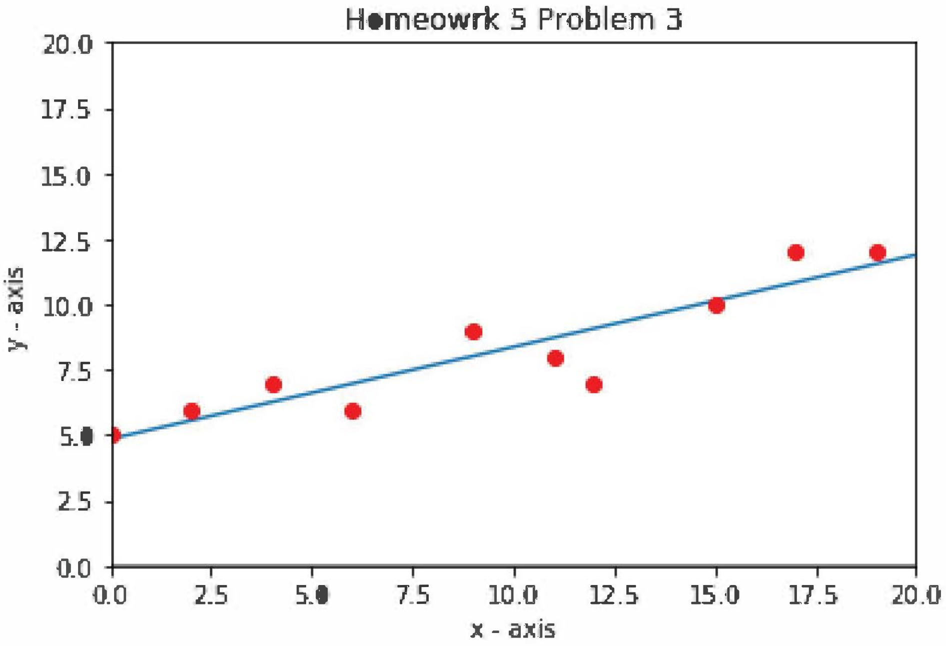
Minimize

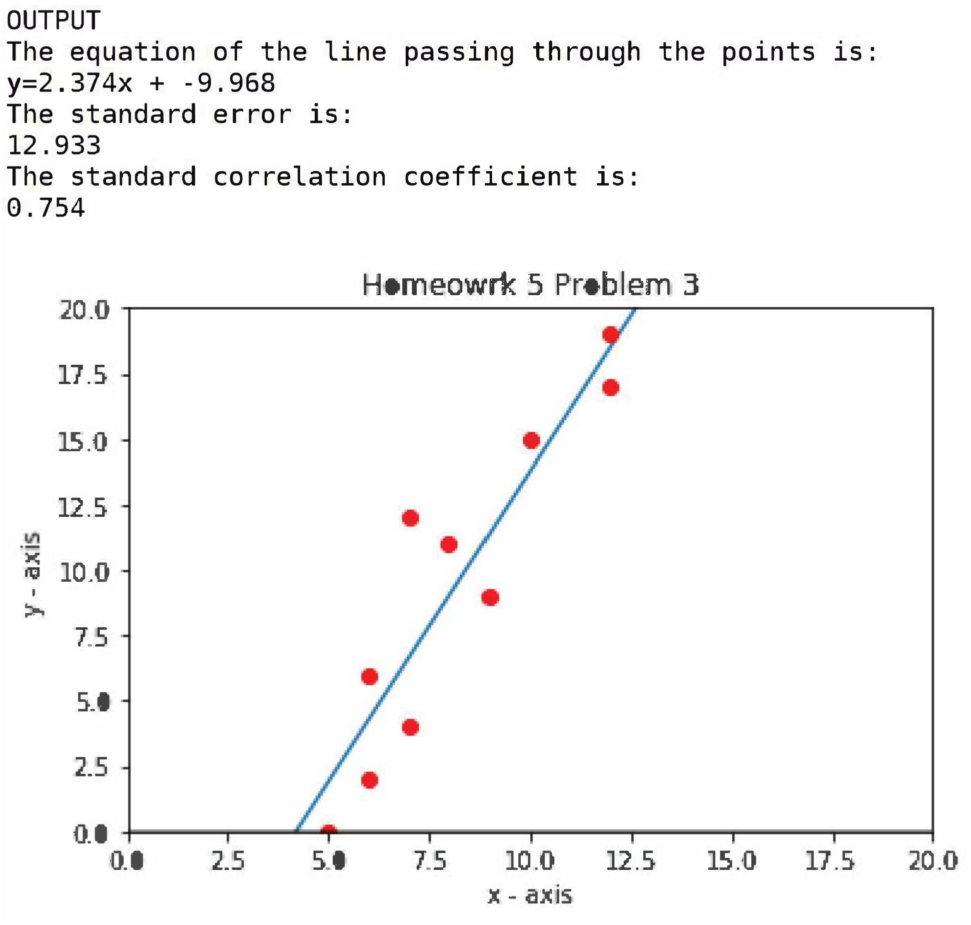
X*(.2) + y*(.03) + z*(.05) = 6	0.825

```
'''
Homework 5 Problem 17.3
Use a least-squares regression to fit a straight line to:

x | 0  2  4  6  9  11  12  15  17  19
---------------------------------------
y | 5  6  7  6  9  8   7   10  12  12

along with the slope and intercept, compute the standard error of the the
estimate and the correlation coefficient. Plot the data and the regression
line. Then repeat the problem, but regress x vs y - that is, switch the
varibles. Interpret your results.

@author: Jacob Needham
'''

'''
Finding best fit function
'''
import math
import numpy as np
import matplotlib.pyplot as graph

#data from problem statement
x = np.matrix([0,2,4,6,9,11,12,15,17,19],dtype='float')
y = np.matrix([5,6,7,6,9,8,7,10,12,12],dtype='float')
m = 2                             #number of basis functions plus a constant
n = 10                            #height of matrix
z = np.matrix([[None for x in range(n)] for y in range(m)],dtype='float')


#Hard coding each part of the basis function
def a0(x):
    y = 1
    return y

def a1(x):
    y = x
    return y


#This populates the z matrix
for row in range(m):
    for column in range(n):
        if row == 0:
            z[0,column] = a0(x[0,column])
        if row == 1:
            z[1,column] = a1(x[0,column])

#orients matrix for output
z=z.transpose()
y=y.transpose()

#this solves for the solution vector that contains coefficients a1-an
#solution vector A = (z(t)*z)^-1 * z(t) * y
A = np.dot(np.dot(np.linalg.inv(np.dot(z.transpose(),z)),z.transpose()),y)
```

```
a0 = A[0,0]
a1 = A[1,0]

#Printing the solution equation
print("The equation of the line passing through the points is:")
print("y=",round(a1,3),"x + ",round(a0,3),sep="")

#line of best fit function
def function(x):
    y = a0 + a1*x
    return y


'''
Finding standard error and correlation coefficient
'''
#Standard Error
Sr = 0
for i in range(n):
    Sr += (function(y.item(i)-a0-a1*x.item(i)))**2
Sy = math.sqrt(Sr/(n-2))

print("The standard error is:")
print(round(Sy,3))

#Correlation Coefficient
yBar = y.sum()/n
St = 0
for i in range(n):
    St += (function(y.item(i)-yBar))**2
r = math.sqrt((St-Sr)/St)
print("The standard correlation coefficient is:")
print(round(r,3))



'''
Plot
'''

#graph
def plotSpace ():
    #setting up function
    x = np.arange(-100,100,.01)
    y = function(x)
    graph.ylim(0,20)
    graph.xlim(0, 20)
    #plotting function
    graph.plot(x, y)
    #plotting data series
    graph.plot([0,2,4,6,9,11,12,15,17,19],[5,6,7,6,9,8,7,10,12,12],'ro')

    #setting up graph
    graph.xlabel('x - axis')
    graph.ylabel('y - axis')
    graph.title('Homeowrk 5 Problem 3')
    #plotting axis
    graph.plot(x, x*0 + 0 , linewidth = .5, color = 'black')
    graph.plot(x*0 +0, y, linewidth =.5, color = 'black')

    #grpahing
    graph.show()

plotSpace()

'''
DISCUSSION
The second half of the assingment is identical, however x and y are flipped.
It is intreting to note that beacuse the correlation coefficient measures
the difference between the average y of the points and the current point, it
is much more sensative to the relative height of the points. The line may be
just as 'tightly' fitted to the data, but becaue the point are more spread in
the part of the assignment, it may not be represented in the coefficients.
'''
```

```
OUTPUT
The equation of the line passing through the points is:
y=0.352x + 4.852
The standard error is:
5.437
The standard correlation coefficient is:
0.154
```



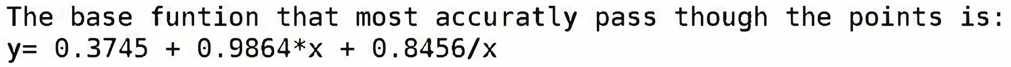


```
OUTPUT
The equation of the line passing through the points is:
y=2.374x + -9.968
The standard error is:
12.933
The standard correlation coefficient is:
0.754
```

```
'''
Homework 5 Problem 17.15
The following data are provided:

x | 1    2    3    4    5
-----------------------------
y | 2.2  2.8  3.6  4.5  5.5

You want to use leat-squares regression to fit these data with the following
model, y = a + bx + c/x. Determine the coefficients by setting up and solving
Eq. (17.25).

@author: Jacob Needham
'''

import numpy as np

x = np.matrix([1,2,3,4,5], dtype = 'float')
y = np.matrix([2.2,2.8,3.6,4.5,5.5], dtype = 'float')
m = 3                             #number of basis functions plus a constant
n = 5                             #height of matrix
z = np.matrix([[None for x in range(n)] for y in range(m)],dtype='float')


#Hard coding each part of the basis function
def a0(x):
    y = 1
    return y

def a1(x):
    y = x
    return y

def a2(x):
    y = 1/x
    return y

#Populating the z matrix
for row in range(m):
    for column in range(n):
        if row == 0:
            z[0,column] = a0(x[0,column])
        if row == 1:
            z[1,column] = a1(x[0,column])
        if row == 2:
            z[2,column] = a2(x[0,column])

#orients matrix for output
z=z.transpose()
y=y.transpose()

#this solves for the solution vector that contains coefficients a1-an
#solution vector A = (z(t)*z)^-1 * z(t) * y
A = np.dot(np.dot(np.linalg.inv(np.dot(z.transpose(),z)),z.transpose()),y)

a0 = A[0,0]
a1 = A[1,0]
```

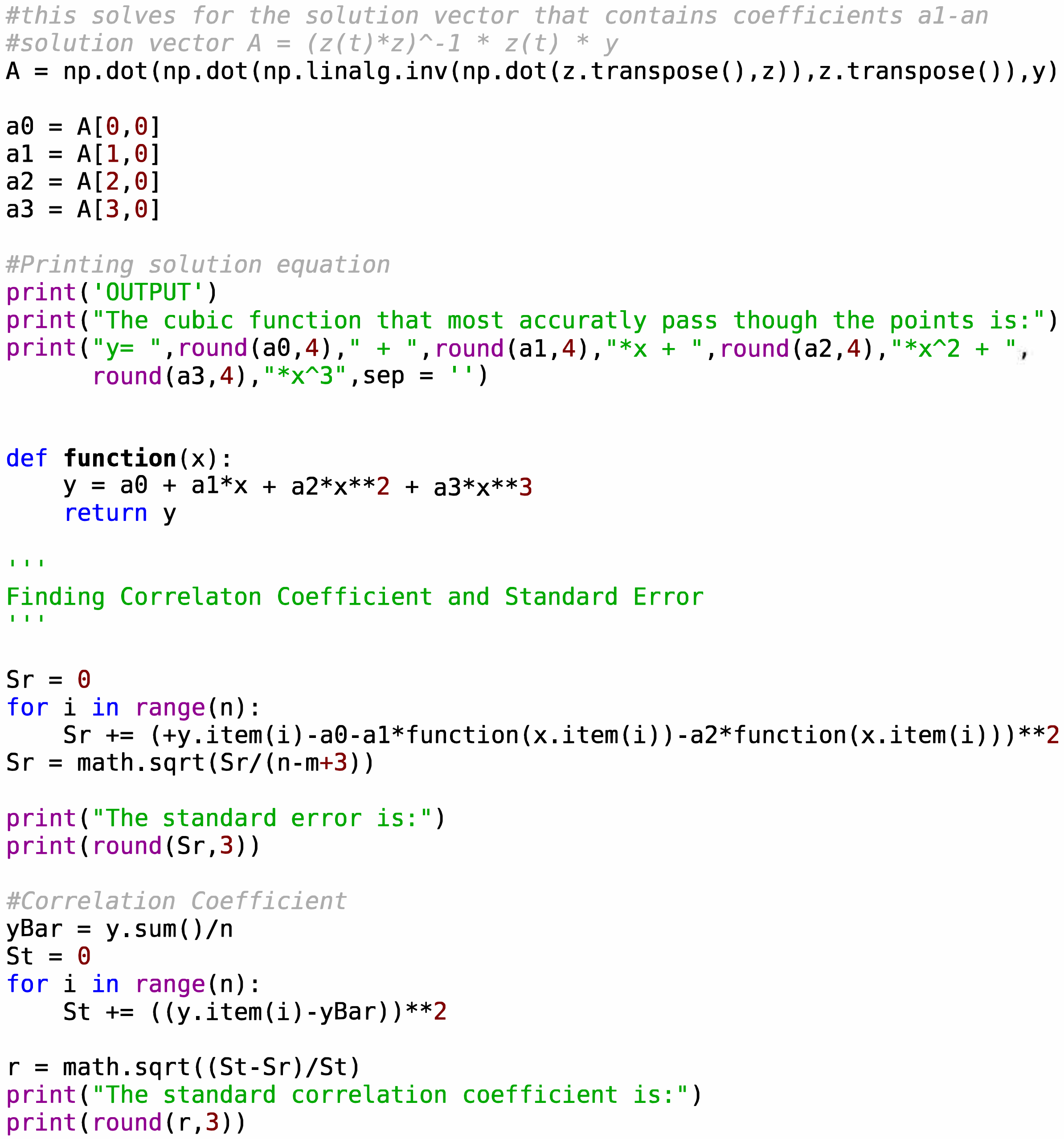
```
a2 = A[2,0]

#Printing solution equation
print("The base funtion that most accuratly pass though the points is:")
print("y= ",round(a0,4)," + ",round(a1,4),"*x + ",round(a2,4),"/x",sep = '')
```

OUTPUT

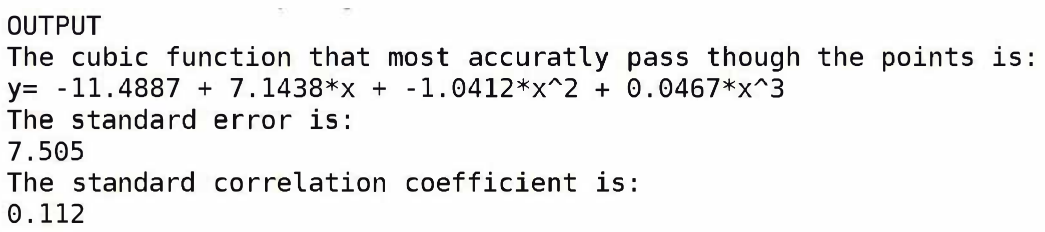


```
The base funtion that most accuratly pass though the points is:
y= 0.3745 + 0.9864*x + 0.8456/x
```

```
'''
Homework 5 Problem 17.17
Fit the cubic equation to the following data:

x | 3    4    5    7    8    9    11   12
------------------------------------------
y | 1.6  3.6  4.4  3.4  2.2  2.8  3.8  4.6

along with the coefficients, determine r^2 and Sy/x.

@author Jacob Needhm
'''

import numpy as np
import math

x = np.matrix([3,4,5,7,8,9,11,12], dtype = 'float')
y = np.matrix([1.6,3.6,4.4,3.4,2.2,2.8,3.8,4.6], dtype = 'float')
m = 4                            #number of basis functions plus a constant
n = 8                            #height of matrix
z = np.matrix([[None for x in range(n)] for y in range(m)],dtype='float')


#Hard coding each part of the basis function
def a0(x):
    y = 1
    return y

def a1(x):
    y = x
    return y

def a2(x):
    y = x**2
    return y

def a3(x):
    y = x**3
    return y


#This populates the z matrix
for row in range(m):
    for column in range(n):
        if row == 0:
            z[0,column] = a0(x[0,column])
        if row == 1:
            z[1,column] = a1(x[0,column])
        if row == 2:
            z[2,column] = a2(x[0,column])
        if row == 3:
            z[3,column] = a3(x[0,column])

#orients matrix for output
z=z.transpose()
y=y.transpose()
```



```
#this solves for the solution vector that contains coefficients a1-an
#solution vector A = (z(t)*z)^-1 * z(t) * y
A = np.dot(np.dot(np.linalg.inv(np.dot(z.transpose(),z)),z.transpose()),y)

a0 = A[0,0]
a1 = A[1,0]
a2 = A[2,0]
a3 = A[3,0]

#Printing solution equation
print('OUTPUT')
print("The cubic function that most accuratly pass though the points is:")
print("y= ",round(a0,4)," + ",round(a1,4),"*x + ",round(a2,4),"*x^2 + ",
      round(a3,4),"*x^3",sep = '')


def function(x):
    y = a0 + a1*x + a2*x**2 + a3*x**3
    return y

'''
Finding Correlaton Coefficient and Standard Error
'''

Sr = 0
for i in range(n):
    Sr += (+y.item(i)-a0-a1*function(x.item(i))-a2*function(x.item(i)))**2
Sr = math.sqrt(Sr/(n-m+3))

print("The standard error is:")
print(round(Sr,3))

#Correlation Coefficient
yBar = y.sum()/n
St = 0
for i in range(n):
    St += ((y.item(i)-yBar))**2

r = math.sqrt((St-Sr)/St)
print("The standard correlation coefficient is:")
print(round(r,3))
```



```
OUTPUT
The cubic function that most accuratly pass though the points is:
y= -11.4887 + 7.1438*x + -1.0412*x^2 + 0.0467*x^3
The standard error is:
7.505
The standard correlation coefficient is:
0.112
```