Master of Science in Cyber-physical and Social System

Optimization and Operation Research

Constrained optimization - Practical Session

Student: Arun Raveendran Nair Sheela CPS2

Lecturers: Professor Antoine Gourru

Saint-Etienne 2021

Content

Convex Programming Problem –

The Optimization problem of the form:

$$\text{Min } F(x)$$

$$\text{subject to : } G_i(x) <= 0, \text{ where } i = 1, 2, 3, 4, \ldots, m.$$

is called a Convex Programming Problem if F(x) and $G_i(x)$ (i = 1,2,3,4,…..,m) are Convex Functions.

Different formats of Convex Programming Problem –

Optimization Problem	Conditions for Convex Programming Problem
Min F(x) subject to: $G_i(x) <= 0$, (i = 1,2,….,m)	F(x) and $G_i(x)$ are convex function
Max F(x) subject to: $G_i(x) <= 0$, (i = 1,2,….,m)	F(x) is concave and $G_i(x)$ are convex
Min F(x) subject to: $G_i(x) <= 0$, (i = 1, 2,….,m)	F(x) is convex and $G_i(x)$ are concave
Max F(x) subject to: $G_i(x) <= 0$, (i = 1,2,….,m)	F(x) and $G_i(x)$ are concave

Solving Convex Optimization Problem using Python CVXPY library –

CVXPY is an open source Python-embedded modelling language for convex optimization problems. It lets you express your problem in a natural way that follows the math, rather than in the restrictive standard form required by solvers.

Steps for solving Convex Optimization problem using CVXPY library –

Step1 - Install and Import the python CVXPY library.

Step2 - Declare the variable involved in the problem using the "cvxpy.Variable()" command.

Step3 – Declare the constraint(equality and inequality constraints) involved in the problem and append all constraints in a python list

Step4 – Declare the objective function using the command either "cvxpy.Minimise(expression)" or "cvxpy.Maximize(expression)"

Step5 - Define the problem using the objective function and defined constrains using the command "cvxpy.Problem(objective function,constraints)"

Step6 – Solve the problem using the command “cvxpy.solve()”

Step7 – Now the optimal value and the variable value is available in the “problem.value()” and “variable.value()” correspondingly

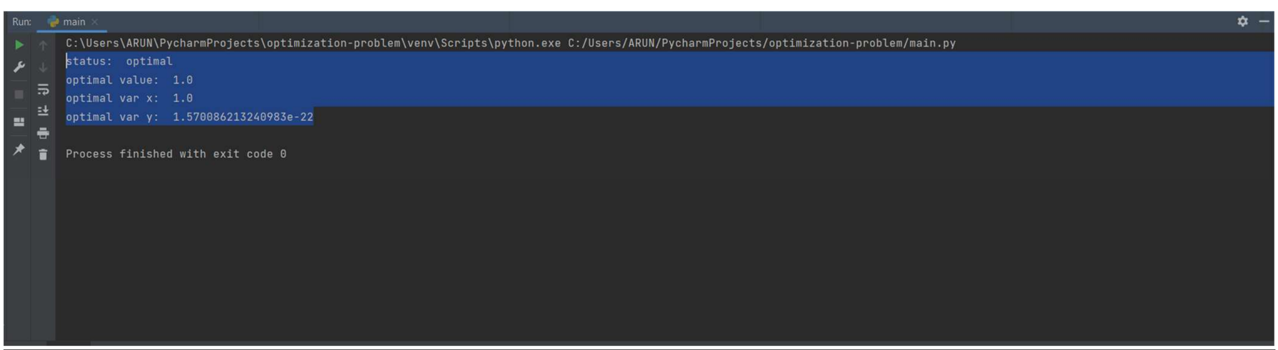
Problem 1 -

1. Min ($(x-y)^2$) , s.t – x +y = 1 and x-y >= 1

Python Program –

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [x + y == 1,
x - y >= 1]

# Form objective.
obj = cp.Minimize((x - y)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.
print("status: ", prob.status)
print("optimal value: ", prob.value)
print("optimal var x: ", x.value)
print("optimal var y: " ,y.value)
```

Output of the above program –



status: optimal

optimal value: 1.0

optimal var x: 1.0

optimal var y: 1.570086213240983e-22

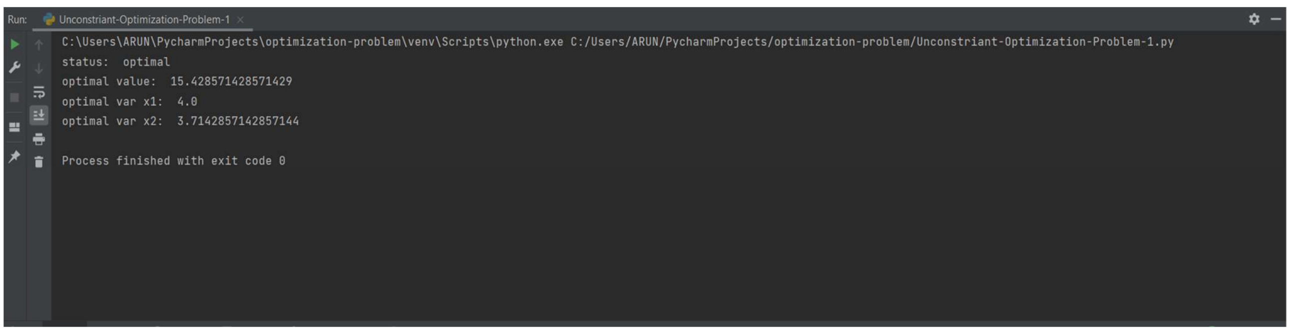
2. Unconstrained Optimization –

1. Problem 1 –
   Min $(x_1 – 4)^2 + 7(x_2 – 4)^2 + 4x_2$ where $x_1, x_2 \in R$

   Python program to solve the above expression using CVXPY library –

```
import cvxpy as cp

# Create two scalar optimization variables.
x1 = cp.Variable()
x2 = cp.Variable()

#form objective.
obj = cp.Minimize((x1-4)**2 + 7*(x2-4)**2 + 4*x2)

prob = cp.Problem(obj)
#solution.
prob.solve()
print("status: ", prob.status)
print("optimal value: ", prob.value)
print("optimal var x1: ", x1.value)
print("optimal var x2: ", x2.value)
```

Output of the Program –



status: optimal

optimal value: 15.428571428571429

optimal var x1: 4.0

optimal var x2: 3.7142857142857144

1) $\min_{x_1, x_2 \in R} (x_1-4)^2 + 7(x_2-4)^2 + 4x_2$ — (1)

To solve this equation
we need to put

$$\frac{\partial f}{\partial x_1} = 0 \quad \text{and} \quad \frac{\partial f}{\partial x_2} = 0$$

$= x_1^2 - 8x_1 + 16 + 7(x_2^2 - 8x_2 + 16) + 4x_2$

$= x_1^2 - 8x_1 + 16 + 7x_2^2 - 56x_2 + 16 + 4x_2$

$= x_1^2 - 8x_1 + 16 + 7x_2^2 - 52x_2 + 16$

$$\frac{\partial f}{\partial x_1} = 2x_1 - 8$$

$$\frac{\partial f}{\partial x_2} = 14x_2 - 52$$

$$2x_1 - 8 = 0$$

$$x_1 = \frac{8}{2} = 4 \quad \text{and} \quad x_2 = \frac{52}{14} = 3.7142$$

here solution is

4 and 3.7142

when we substitute the above value of x_1 and x_2 is equation (1)
we get the optimal solution
which is equal to 15.42777148

Here I proved that the solution get using the library CVXPY are correct by doing on my own

2. Problem 2 –

Min $(x_1^3 + (x_2 - x_3)^2 + x_3^3 + 2)$ where $x_1, x_2, x_3 \in R$

Here first need to prove the above equation is convex

For that first need to find the hessian matrix and then calculate the eigen values of the hessian matrix and prove that all the “lambda “ values are greater than 0

$$\min_{x_1,x_2,x_3 \in R} x_1^3 + (x_2 - x_3)^2 + x_3^3 + 2$$

equation $= x_1^3 + x_2^2 - 2x_2x_3 + x_3^2 + x_3^3 + 2$

To verify the above equation is convex first need to calculate the hessian matrix

$$\frac{\partial f}{\partial x_1} = 3x_1^2$$

$$\frac{\partial f}{\partial x_2} = 2x_2 - 2x_3$$

$$\frac{\partial f}{\partial x_3} = -2x_2 + 2x_3 + 3x_3^2$$

$$H = \nabla^2 g_1 = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial^2 x_3} \end{vmatrix}$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = 0$$

$$\frac{\partial^2 f}{\partial x_1 \partial 3} = 0$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_1} = 0$$

$$\frac{\partial^2 f}{\partial^2 x_2} = 2$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_3} = -2$$

$$\frac{\partial^2 f}{\partial x_3 \partial x_1} = 0$$

$$\frac{\partial^2 f}{\partial x_3 \partial x_2} = -2$$

$$\frac{\partial^2 f}{\partial^2 x_3} = 2 + 6x_3$$

$$= \begin{vmatrix} 6x_1 & 0 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 6x_3 + 2 \end{vmatrix}$$

eigen value of matrix is

$\lambda_1 \Rightarrow 6x_1$

$\lambda_2 \rightarrow \sqrt{9x_3^2 + 4} + 3x_3 + 2$

$\lambda_3 \rightarrow \sqrt{9x_3^2 + 4} + 3x_3 + 2$

~~here some λ_s are negative~~

So this function is not convex (all λ_s should be positive, for a function to be convex)

here there is a possibility that λ_2 become negative.

eigen value online calculator screenshot –

Start from forming a new matrix by subtracting λ from the diagonal entries of the given matrix:

$$\begin{bmatrix} -\lambda + 6x & 0 & 0 \\ 0 & 2 - \lambda & -2 \\ 0 & -2 & -\lambda + 6y + 2 \end{bmatrix}.$$

The determinant of the obtained matrix is $-\left(-\lambda + 6x\right)\left(-\lambda^2 + 6\lambda y + 4\lambda - 12y\right)$ (for steps, see determinant calculator).

Solve the equation $-\left(-\lambda + 6x\right)\left(-\lambda^2 + 6\lambda y + 4\lambda - 12y\right) = 0$.

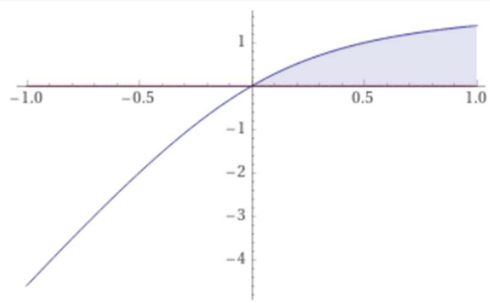
The roots are $\lambda_1 = 3y - \sqrt{9y^2 + 4} + 2$, $\lambda_2 = 3y + \sqrt{9y^2 + 4} + 2$, $\lambda_3 = 6x$ (for steps, see equation solver).

Constraints Calculator by solve the inequality online screenshot –

Input

$3x - \sqrt{9x^2 + 4} + 2 > 0$

Inequality plot



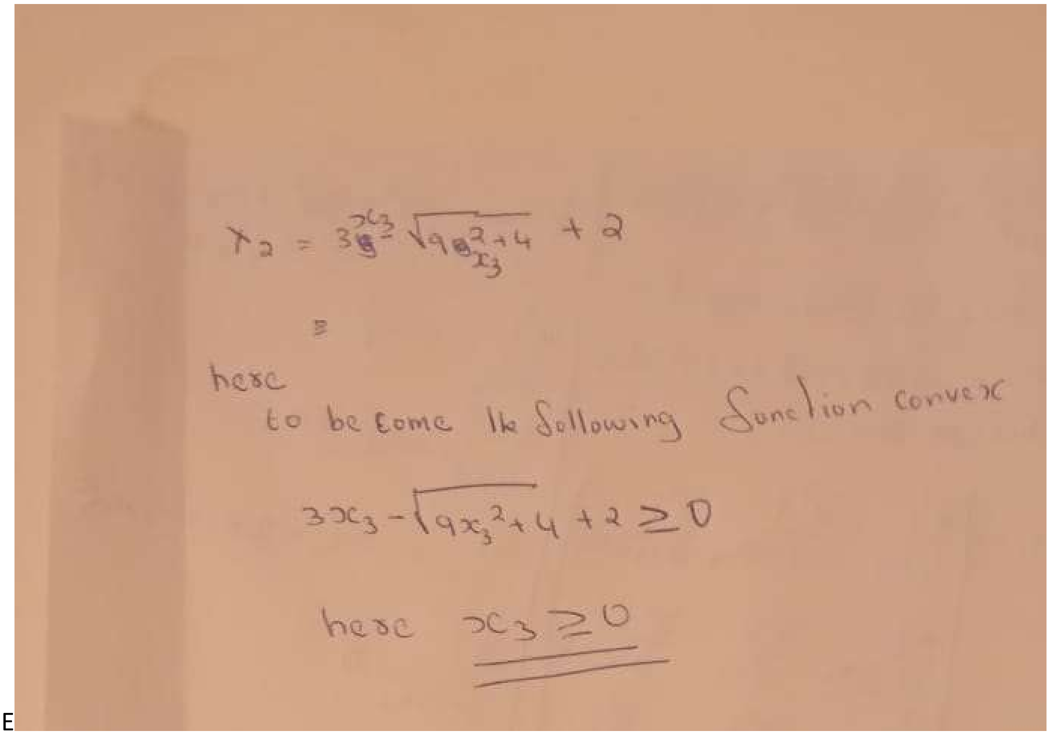
Alternate forms

$3x + 2 > \sqrt{9x^2 + 4}$

$3x - \sqrt{9x^2 + 4} > -2$

Solution

$x > 0$



$x_2 = 3x_3 - \sqrt{9x_3^2+4} + 2$

here
to become the following function convex

$3x_3 - \sqrt{9x_3^2+4} + 2 \geq 0$

here $x_3 \geq 0$

E

Python program to solve the above expression using CVXPY library –

Here I user sympy library to calculate the eigen value and derivative of the function also I used the numpy array

Command - equation.diff(variablename)

```
import cvxpy as cp
from sympy import symbols,Matrix
from numpy.linalg import eig
import numpy as np

#SymPy is a Python library for symbolic mathematics. It aims to become a
full-featured computer algebra system (CAS)
# while keeping the code as simple as possible in order to be
comprehensible and easily extensible.
# SymPy is written entirely in Python.

#Before we can construct symbolic math expressions or symbolic math
equations  with SymPy,
# first we need to create symbolic math variables, also called symbolic
math symbols.
x1 = symbols('x1')
x2 = symbols('x2')
x3 = symbols('x3')

equation = x1**3 + (x2-x3)**2 + x3**3 + 2

# sympy library is used to find the derivative of the function
#calculate the derivative of function "equation" with respect to x1
derivative_x1 = equation.diff(x1)
```

```
print("Derivative of function wrt x1 :",derivative_x1)

#calculate the derivative of function "equation" with respect to x2
derivative_x2 = equation.diff(x2)
print("Derivative of function wrt x2 :",derivative_x2)

#calculate the derivative of functioin "equation" with respect to x3
derivative_x3 = equation.diff(x3)
print("Derivative of function wrt x3 :",derivative_x3)
matrix =  []
#hessain matrix
matrix =
np.array([[derivative_x1.diff(x1),derivative_x1.diff(x2),derivative_x1.diff
(x3)],

[derivative_x2.diff(x1),derivative_x2.diff(x2),derivative_x2.diff(x3)],

[derivative_x3.diff(x1),derivative_x3.diff(x2),derivative_x3.diff(x3)]])

print(matrix)

hessian_matrix = Matrix(matrix)
print("eigen value: ", hessian_matrix.eigenvals())
# here all the eign value are not positive this is not convex

print(hessian_matrix.det())



x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()

obj = cp.Minimize(x1**3 + (x2-x3)**2 + x3**3 + 2)
constraints  = [0<=x3]

prob =cp.Problem(obj,constraints)
prob.solve()

print("status: ", prob.status)
print("optimal value: ", prob.value)
print("optimal var x1: ", x1.value)
print("optimal var x2: ", x2.value)
print("optimal var x3: ", x3.value)
```

Output -

```
Run: Unconstraint-Optimization-Problem-2
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Unconstraint-Optimization-Problem-2.py
Derivative of function wrt x1 : 3*x1**2
Derivative of function wrt x2 : 2*x2 - 2*x3
Derivative of function wrt x3 : -2*x2 + 3*x3**2 + 2*x3
[[6*x1 0 0]
 [0 2 -2]
 [0 -2 6*x3 + 2]]
eigen value:  {6*x1: 1, 3*x3 - sqrt(9*x3**2 + 4) + 2: 1, 3*x3 + sqrt(9*x3**2 + 4) + 2: 1}
72*x1*x3
status:  optimal
optimal value:  2.0000000002159073
optimal var x1:  0.00036328240113486077
optimal var x2:  0.0005517450426201796
optimal var x3:  0.0005517450269280136
```

3. Problem – 3

Min $(X_1 - 2)^2 + 3X_2$, s,t. $-x_1 - x_2 <= -4$

Reformulate this problem using the log-barrier function in CVXPY

Log Barrier function –

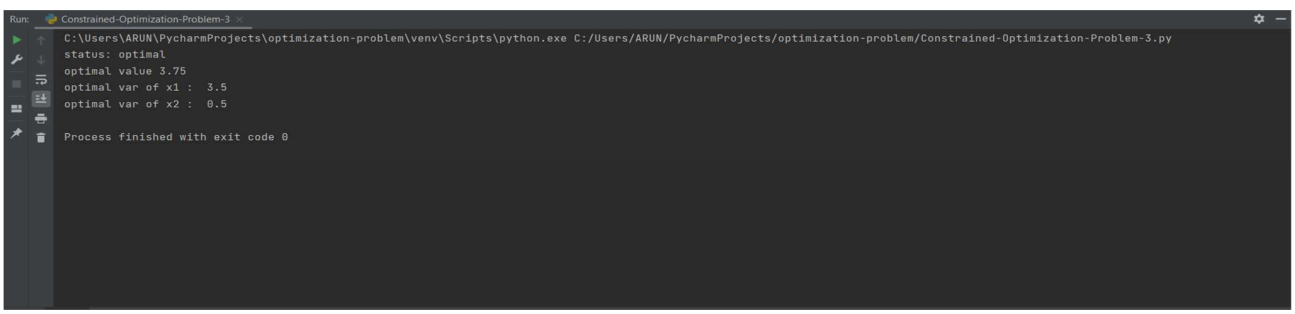
$$F(x) - \frac{1}{t}\left(\log\left(-G_i(x)\right)\right)$$

Where F(x) is the equation to find the solution and G_i (where i = 1,2,3,4 ….) are the constraints

Python Program -

```
import cvxpy as cp

x1 = cp.Variable()
x2 = cp.Variable()

obj = cp.Minimize((x1-2)**2 + 3*x2)
const = [-x1-x2+4<=0]
prob = cp.Problem(obj,const)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var of x1 : ", x1.value)
print("optimal var of x2 : ",  x2.value)
```

Output –



Reformulating the above problem using the log-barrier function is

$$\text{Min } ((X_1 - 2)^2 + 3X_2 - \frac{1}{t}(\log(-(-X_1 - X_2 + 4)))$$

Here t value is 1,100,1000 etc

When k = 1 ,

Program -

```python
import cvxpy as cp

x1 = cp.Variable()
x2 = cp.Variable()

obj = cp.Minimize((x1-2)**2 + 3*x2 - 1*cp.log(-(-x1-x2+4)))

prob = cp.Problem(obj)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var of x1 : ", x1.value)
print("optimal var of x2 : ",  x2.value)
```

Output :

```
Run: Constrained-Optimization-Problem-3
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Constrained-Optimization-Problem-3.py
status: optimal
optimal value 5.848612288668269
optimal var of x1 :  3.500000398379134
optimal var of x2 :  0.8333329278362519

Process finished with exit code 0
```

When k = 100

Program -

```python
import cvxpy as cp

x1 = cp.Variable()
x2 = cp.Variable()

obj = cp.Minimize((x1-2)**2 + 3*x2 - (1/100)*cp.log(-(-x1-x2+4)))

prob = cp.Problem(obj)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var of x1 : ", x1.value)
print("optimal var of x2 : ",  x2.value)
```

Output –

```
Run: Constrained-Optimization-Problem-3
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Constrained-Optimization-Problem-3.py
status: optimal
optimal value 3.8170378247468246
optimal var of x1 :  3.500000442670067
optimal var of x2 :  0.5033328785066005

Process finished with exit code 0
```

Put k = 1000 ,

Program -

```
import cvxpy as cp

x1 = cp.Variable()
x2 = cp.Variable()

obj = cp.Minimize((x1-2)**2 + 3*x2 - (1/1000)*cp.log(-(-x1-x2+4)))

prob = cp.Problem(obj)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var of x1 : ", x1.value)
print("optimal var of x2 : ",  x2.value)
```

Output –

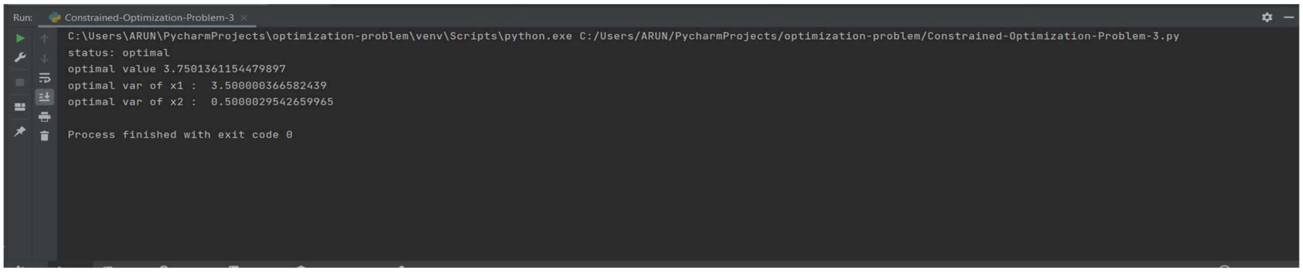
```
Run: Constrained-Optimization-Problem-3
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Constrained-Optimization-Problem-3.py
status: optimal
optimal value 3.7590063675782535
optimal var of x1 :  3.500000963375165
optimal var of x2 :  0.500332323592533

Process finished with exit code 0
```

When put k= 100000,

Program -

```
import cvxpy as cp

x1 = cp.Variable()
x2 = cp.Variable()

obj = cp.Minimize((x1-2)**2 + 3*x2 - (1/100000)*cp.log(-(-x1-x2+4)))

prob = cp.Problem(obj)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var of x1 : ", x1.value)
print("optimal var of x2 : ",  x2.value)
```

Ouput -



```
Run:  Constrained-Optimization-Problem-3
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Constrained-Optimization-Problem-3.py
status: optimal
optimal value 3.7501361154479897
optimal var of x1 :  3.500000366582439
optimal var of x2 :  0.5000029542659965

Process finished with exit code 0
```

As I observed here that the optimal value has no change when the paremeter k passed a certain limit for example, the optimal solution is almost same, when k= 1000 and k = 100000

3 . <u>**Modeling Constrained Problems**</u>

1. Water resources –

Total water requirement in the city - 500,000 litres

Sources – Reservoir and Stream

Cost – 100 Euro per 1000L for Reservoir and 50 Euro per 1000L for stream

Upper Limit – 100,000 L for stream and from reservoir is unlimited

Pollution – 50 ppm for reservoir and 250 ppm for stream (ppm means One ppm is equivalent to **1 milligram of something per litre of water** (mg/l) or 1 milligram of something per kilogram soil (mg/kg))

<u>Expression –</u>

Variables – Reservoir Quantity and Stream Quantity

Total Cost Reservoir = 100 Euro per 1000L

Toal Cost Stream = 50 Euro Per 1000L

Total cost $= \left(\frac{Reservoir\ Quantity * Total\ Cost\ Reservoir}{1000}\right) + \left(\frac{Stream\ Quantity * Total\ Cost\ Stream}{1000}\right)$

$= \left(\frac{Reservoir\ Quantity * 100}{1000}\right) + \left(\frac{Stream\ Quantity * 50}{1000}\right)$

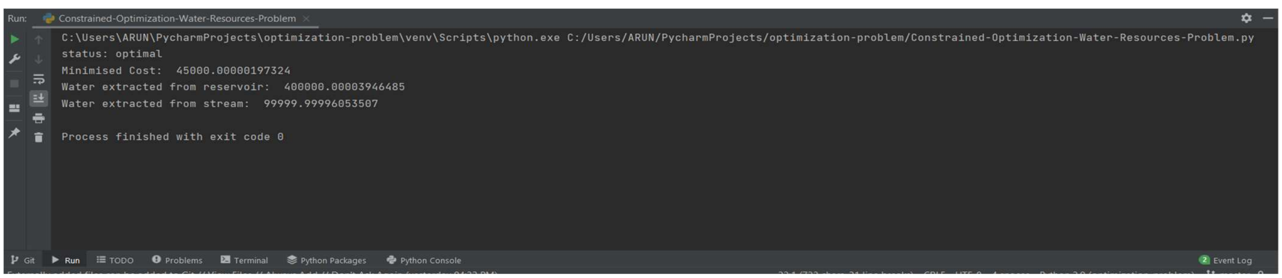
Constraints –

1. Total quantity of water from both the sources should be equal to 500,000
   Reservoir Quantity + Stream Quantity = 500,000
2. Stream Quantity should be less than 100,000L
   Stream Quanity <= 100,000
3. The concentration of pollutants in the water served to the city should bed less than 100ppm
   Total pollutants =
   Reservoir Quantity * 50 + Stream Quantity * 250 = 500,000 * 100
   = Reservoir Quantity * 50 + Stream Quantity * 250 = 500,000,00

Python Program to solve the problem –

```
import cvxpy as cp

# total quantity extracted from reservoir in litres
reservoirQuantity  = cp.Variable()

# total quantity extracted from stream in litres
streamQuantity = cp.Variable()

#total cost calculation formula
totalCost =
cp.Minimize(((reservoirQuantity*100)/1000)+(50*streamQuantity/1000))

constraints = [reservoirQuantity + streamQuantity ==500000, streamQuantity
<=100000,
               reservoirQuantity * 50 + streamQuantity*250 <= 50000000]

probSolve = cp.Problem(totalCost,constraints)
probSolve.solve()

print("status:", probSolve.status)
print("Minimised Cost: ", probSolve.value)
print("Water extracted from reservoir: ", reservoirQuantity .value)
print("Water extracted from stream: ", streamQuantity.value)
```

Output of the Program –



status: optimal

Minimized Cost: 45000.00000197324

Water extracted from reservoir: 400000.00003946485

Water extracted from stream: 99999.99996053507

2. Good Smelling Perfume Problem –

   Calculate the least costly way of mixing the 4 blends of essential oils to produce a new perfume.

   Variables = blendOne, blendTwo, blendThree, blendFour
   Expression to calculate total cost =
   blendOne * 55 + blendTwo * 65 + blendThree * 35 + blendFour * 85

Constraints –

1. The percentage of blend 2 in the perfume must be at least 5% and cannot exceed 20%
   0.05 <= blendTwo and blendTwo <= .20

2. The percentage of blend 3 has to be at least 30%

   0.30 <= blendThree

3. The percentage of blend 1 has to be between 10% and 25%,

   0.10 <= blendOne and blendOne <= 0.25

4. The final percentage of bergamot orange content in the perfume must be at most 50%

   Total bergamot orange content =

   = blendOne * 0.35 + blendTwo * 0.60 + blendThree * 0.35 + blendFour * 0.40

   Total bergamot orange content <=0.50

5 . The final percentage of thymus content has to be between 8% to 13%,
Total thymus content =

= blendOne * 0.15 + blendTwo * 0.05 + blendThree * 0.20 + blendFour * 0.10

0.08 <= Total thymus content and Total thymus content <= 0.13

6. The final percentage of rose content must be at most 35%

   Total rose content =

   = blendOne * 0.30 + blendTwo * 0.20 + blendThree * 0.40 + blendFour * 0.20

   Total rose content <= 0.35

7. The percentage of lily of the valley content has to be at least 19%.

   Total Lilly content =

   = blendOne * 0.20 + blendTwo * 0.15 + blendThree * 0.05 + blendFour * 0.30

   0.19 <= Total Lilly content

8. Total sum of variable should be 100%

   = blendOne + blendTwo + blendThree + blendFour = 1

Python Program to solve the Problem –

```
import cvxpy as cp

# variable declaration for blend
blendOne = cp.Variable()
blendTwo = cp.Variable()
blendThree = cp.Variable()
blendFour = cp.Variable()

# equation for the total cost
totalCost = blendOne * 55 + blendTwo * 65 + blendThree * 35 + blendFour *
```

```
85

calculateTotalCost = cp.Minimize(totalCost)


# total percentage of beragamont Orange content in the blend
beragamotOrange =  0.35 * blendOne + 0.60 * blendTwo + 0.35 * blendThree +
0.40 * blendFour

# total percentage of thymus content in the blend
thymusContent = 0.15 * blendOne + 0.05 * blendTwo + 0.20 * blendThree +
0.10 * blendFour

# total percentage of rose centent in the blend
roseContent =  0.30 * blendOne + 0.20 * blendTwo + 0.40 * blendThree + 0.20
* blendFour

# total percentage of lilly in the blend
lillyContent =0.20 * blendOne + 0.15 * blendTwo + 0.05 * blendThree + 0.30
* blendFour
# constraints
constraints = [0.05 <= blendTwo, blendTwo <= 0.2,
               0.3 <= blendThree,
               0.1 <= blendOne, blendOne <= 0.25,
               beragamotOrange <= 0.5,
               0.08 <= thymusContent, thymusContent <= 0.13,
               roseContent <= 0.35,
               0.19 <= lillyContent,blendOne+blendTwo+blendThree+blendFour
==1
             ]

probSolve = cp.Problem(calculateTotalCost, constraints)
probSolve.solve()

print("status:", probSolve.status)
print("optimal cost", probSolve.value)
print("Percentage of blend One: ", blendOne.value)
print("Percentage of blend Two: ", blendTwo.value)
print("Percentage of blend Three: ", blendThree.value)
print("Percentage of blend Four: ", blendFour.value)
```

Output –

```
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Good-Smelling-Perfume-Problem.py
status: optimal
optimal cost 62.9999999983623
Percentage of blend One:  0.14000000000931784
Percentage of blend Two:  0.14000000001437302
Percentage of blend Three:  0.300000000021845
Percentage of blend Four:  0.41999999995471754

Process finished with exit code 0
```

status: optimal

optimal cost 62.9999999983623

Percentage of blend One: 0.14000000000931784

Percentage of blend Two: 0.14000000001437302

Percentage of blend Three: 0.300000000021845

Percentage of blend Four: 0.41999999995471754

3 . Roadway expenses

B_{rural} = 7000 *(log(1 + x_{rural})

B_{urban} = 7000 * (log(1 + x_{urban})

Variables = xRural ,xUrban

Expression = Maximise (B_{rural} + B_{urban} + x_{rural} + x_{urban})

Python program to solve the problem –

```
import cvxpy as cp

xRural = cp.Variable()
xUrban = cp.Variable()

bRural = 7000 * cp.log(1 + xRural)
bUrban = 5000 * cp.log(1 + xUrban)

equation  = bRural + bUrban - xRural -xUrban

netBenefitCalc = cp.Maximize(equation)

constraints = [xRural + xUrban == 200]

problSolution = cp.Problem(netBenefitCalc,constraints)
problSolution.solve()

print("status:", problSolution.status)
print("Optimal Solution", problSolution.value)
print("xRural : ", xRural.value)
print("xUrban : ", xUrban.value)
```

Output :

```
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/Road-Way-Expenses.py
status: optimal
Optimal Solution 55348.89314799299
xRural :  116.83330317505175
xUrban :  83.1666963297482

Process finished with exit code 0
```

status: optimal

Optimal Solution 55348.89314799299

xRural : 116.83330317505175

xUrban : 83.1666963297482

3.4 Design your own optimization problem –

My Problem is to calculate how much energy should be produced from various sources like Nuclear Energy, Hydro Power Energy, Fossil Fuels, Renewable Energy Resources.

My country have four different ways to produce energy

1. Nuclear Energy
2. Hydro Power Energy
3. Renewable Energy
4. Fossil Fuels

The cost of production for each energy source is different and the total energy needed for my country is 10000kw per day. Also there exist certain criteria that, some energy should take from each energy sources.

This Problem is about calculate the contribution of each energy sources to the total energy with minimal cost.

Sources	Nuclear	Hydro	Renewable	Fossil Fuels
Cost per KW	50 Euro	100 Euro	30 Euro	200 Euro

Total Energy consumption of the country per day – 10,000 kw

Constraints for the above problem -

1. Energy taken from the renewable energy sources should be greater than 10 percentage of the total and less than the 20 percentage of the total.
   10<=renewableEnergy , renewableEnergy <= 20
2. Energy taken from the nuclear energy sources is greater than the 30 percentage of the total.
   30<=nuclearEnergy
3. Energy taken from the hydro energy sources is greater than the 20 percentage of the total
   20<=hydroEnergy
4. Energy taken from the fossil fuel should be greater than 5 percentage and less than 10 percentage of total
   10<=fossilFuel and fossilFuel<=20.

Python program of the above problem using the CVXPY library

```
import cvxpy as cp

#variable declaration
#total energy from fossil fuel in percentage
fossilFuel = cp.Variable()
#total energy from renewable energy resources in percentage
renewableEnergy = cp.Variable()
#total energy from nulcear energy source in percentage
nuclearEnergy = cp.Variable()
#total energy from hydro energy source in percentage
```

```
hydroEnergy = cp.Variable()

#total energey consumption
totalEnergyConsumption = 10000
#cost of individual energy consumption per KW in Euros
costFossilFuelPerKW = 200
costRenewableEnergyPerKW = 30
costNuclearEnergyPerKW = 50
costHydroEnergyPerKW = 100

#total cost of fossil fuel
tCostFossilFuel = ((fossilFuel * totalEnergyConsumption)/100) *
costFossilFuelPerKW

#total cost of renewable energy
tCostRenewableEnergy = ((renewableEnergy * totalEnergyConsumption)/100) *
costRenewableEnergyPerKW

#total cost of nuclear energy
tCostNuclearEnergy = ((nuclearEnergy * totalEnergyConsumption)/100) *
costNuclearEnergyPerKW

#total cost of  hydro energy
tCostHydroEnergy = ((hydroEnergy * totalEnergyConsumption)/100) *
costHydroEnergyPerKW


exp = (tCostFossilFuel + tCostRenewableEnergy + tCostNuclearEnergy +
tCostHydroEnergy)

netUsage = cp.Minimize(exp)
constraints = [fossilFuel+renewableEnergy+nuclearEnergy+hydroEnergy ==
100,10<=renewableEnergy,renewableEnergy<=20,
               30<=nuclearEnergy,20<=hydroEnergy,
5<=fossilFuel,fossilFuel<=10]

probSolve = cp.Problem(netUsage,constraints )
probSolve.solve()

print("status:", probSolve.status)
print("Minimised Cost: ", probSolve.value)
print("Percentage of Total Energy from the Fossil Fuels:  ",
fossilFuel.value)
print("Percentage of  Total Energy from the Renewable Energy Sources: ",
renewableEnergy.value)
print("Percentage of Total Energy from the Nuclear Energy Sources: ",
nuclearEnergy.value)
print("Percentage of  Total Energy from the Hydro Energy Sources: ",
hydroEnergy.value)

#total renewable energy value
totalRenewableEnergyValue  = (renewableEnergy.value *
totalEnergyConsumption)/100
totalFossilEnergyValue  = (fossilFuel.value * totalEnergyConsumption)/100
totalNuclearEnergyValue  = (nuclearEnergy.value *
totalEnergyConsumption)/100
totalHydroEnergyValue  = (hydroEnergy.value * totalEnergyConsumption)/100

#print the total consumtion in kw
print("Total Energy Consumption of Renewable energy in
KW",totalRenewableEnergyValue)
```

```
print("Total Energy Consumption of Fossil fuel in
KW",totalFossilEnergyValue)
print("Total Energy Consumption of Nuclear energy in
KW",totalNuclearEnergyValue)
print("Total Energy Consumption of Hydro energy in
KW",totalHydroEnergyValue)
```

Output of the Program –

```
Run: my-own-problem
C:\Users\ARUN\PycharmProjects\optimization-problem\venv\Scripts\python.exe C:/Users/ARUN/PycharmProjects/optimization-problem/my-own-problem.py
status: optimal
Minimised Cost:  635000.0001969272
Percentage of Total Energy from the Fossil Fuels:   4.99999999498858
Percentage of  Total Energy from the Renewable Energy Sources:  19.999999873847944
Percentage of Total Energy from the Nuclear Energy Sources:  55.000000127401215
Percentage of  Total Energy from the Hydro Energy Sources:  20.000000003860574
Total Energy Consumption of Renewable energy in KW 1999.9999873847946
Total Energy Consumption of Fossil fuel in KW 499.999999498858
Total Energy Consumption of Nuclear energy in KW 5500.000012740122
Total Energy Consumption of Hydro energy in KW 2000.0000003860573

Process finished with exit code 0
```

Values –

status: optimal

Minimised Cost: 635000.0001969272

Percentage of Total Energy from the Fossil Fuels: 4.99999999498858

Percentage of Total Energy from the Renewable Energy Sources: 19.999999873847944

Percentage of Total Energy from the Nuclear Energy Sources: 55.000000127401215

Percentage of Total Energy from the Hydro Energy Sources: 20.000000003860574

Total Energy Consumption of Renewable energy in KW 1999.9999873847946

Total Energy Consumption of Fossil fuel in KW 499.999999498858

Total Energy Consumption of Nuclear energy in KW 5500.000012740122

Total Energy Consumption of Hydro energy in KW 2000.0000003860573

<u>Conclusion –</u>

From this exercise I learned about how to solve the constraint optimization problem using CVXPY python library and also I explored some other python libraries like sympy, for the mathematical calculation which include the differentiation and eigen value calculation. Also the exercise will help solve some application level problems using convex optimization and got chance to design my own problem and solve it using convex optimization. Here I selected the energy consumption problem and objective of this problem is to calculate the optimal cost and percentage of distribution of the energy from various sources.

References –

1. https://problemsolvingwithpython.com/10-Symbolic-Math/10.03-Defining-Variables/
2. https://towardsdatascience.com/hessian-matrix-and-optimization-problems-in-python-3-8-f7cd2a615371
3. https://www.emathhelp.net/en/calculators/linear-algebra/eigenvalue-and-eigenvector-calculator/
4. https://towardsdatascience.com/hessian-matrix-and-optimization-problems-in-python-3-8-f7cd2a615371
5. https://stackoverflow.com/questions/69382757/finding-eigenvalues-of-a-matrix-with-unknown-variables-using-numpy-linalg-eig
6. https://www.wolframalpha.com/
7. http://antoinegourru.com/CM0.pdf