Process Systems Modeling HW-3

Yash Gokhale (ysg)

Q1]

a) For the standard case:

Formulated MIP is:

$\min \sum_1^t C(t) * y(t) + p(t) * x(t) + h(t) * s(t)$

s.t:

y: Binary variable, x,s: Positive variables

s(t=1)=x(t=1)-d(t=1)

$s(t) = s(t-1) + x(t) - d(t)$ for t>1

$x(t) \leq M * y(t)$, where M=430 (Total demand)

Solving the following in GAMS as RMIP yields:

Z=440 (Optimum)

Solving for an MIP:

Z=530 (Optimum)

b) For disaggregated production amounts:

Introduce an alias for t: tau

Formulated MIP is:

$\min \sum_t^T \sum_\tau (P(t) + \sum_t h(t)) q_{t\tau} + \sum_t^T c(t) * y(t)$

St:

Period: tau>t (Always)

$q(t,\tau) \leq M_{t\tau} y_t$

$M_{t\tau} = d(t)$

$\sum_t^{NT} q(t, tau) = d(tau)$

Solving the RMIP yields:

Z=530 (Optimum)

MIP solution gives:

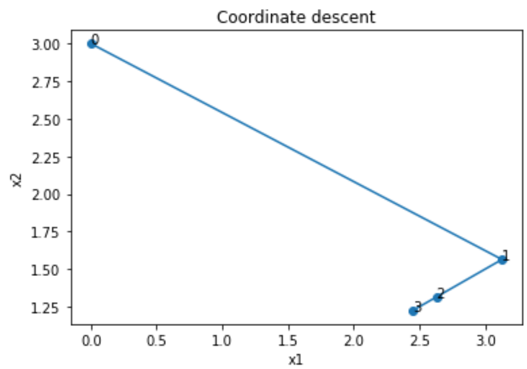
Z=530 (Optimum)

(Corresponding GAMS Files have been submitted)

Q2]

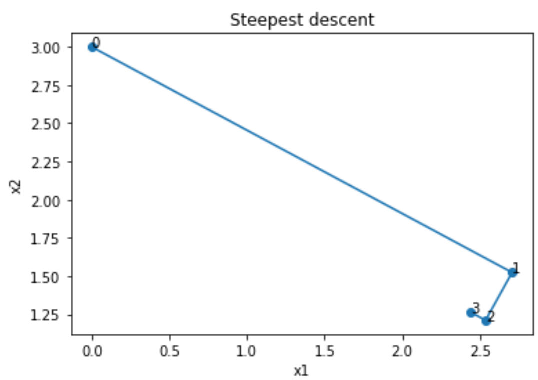
All calculations were done using Python, where relevant codes are attached in the appendix.

a) Cycling Coordinate Method:



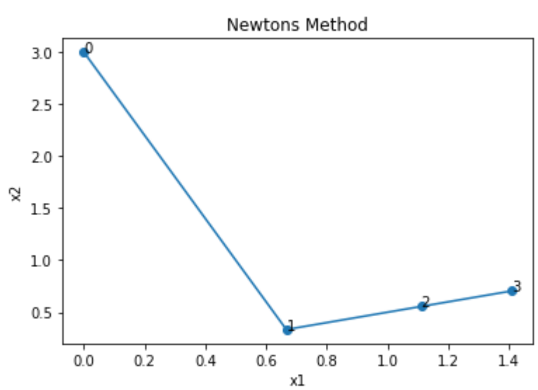
Minimum after 3 iterations at (2.448, 1.224) at function value 0.0405 for the Cycling Coordinate method.

b) Steepest Descent



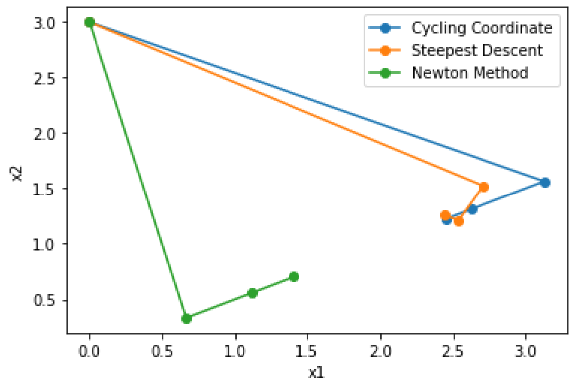
Minimum after 3 iterations at (2.442, 1.262) at function value 0.045

c) Newton's Method



Minimum after 3 iterations at (1.407, 0.704) at function value 0.1233

Comparison plot:



Appendix for Q2:

Coordinate descent

```
x1=0
x2=3
n_iter=6
x1a=np.array([x1])
x2a=np.array([x2])
def obj(x1,x2):
    return (x1-2)**4+(x1-2*x2)**2
def obj2(x2,x1):
    return (x1-2)**4+(x1-2*x2)**2
for i in range(n_iter):
    if i%2==0:
        ff=minimize(obj,0,(x2,))
        x1=ff.x[0]
#         x1a=np.append(x1a,x1)
#         x2a=np.append(x2a,x2)
        print(obj(x1,x2))
    else:
        gg=minimize(obj2,x2,(x1,))
        x2=gg.x[0]
        x1a=np.append(x1a,x1)
        x2a=np.append(x2a,x2)
        print(obj(x1,x2))
```

```
def objective(x1,x2):
    return (x1-2)**4+(x1-2*x2)**2

def dx1(x1,x2):
    return 4*(x1-2)**3+2*(x1-2*x2)

def dx2(x1,x2):
    return 2*(-2)*(x1-2*x2)
```

```
def dx11(x1,x2):
    return 12*(x1-2)**2+2
dx12=-4
dx22=8
```

```
def der(x1,x2):
    u=np.array([dx1(x1,x2),dx2(x1,x2)])
    u=np.reshape(u,(2,1))
    return u.T

def hess(x1,x2):
    p=np.array([[(12*(x1-2)**2+2),-4],[-4,8]])
    return np.linalg.inv(p)
```

Steepest Descent

```
def steep(n_iter,x0):
    x1i,x2i=x0
    xo=np.array([x1i,x2i])
    xo=np.reshape(xo,(2,1))
    t=0.01
    for i in range(n_iter):
        a=np.array([x1i,x2i])
        print(a)
        der1=dx1(a) #derivative1
        der2=dx2(a) #derivate2
        print(der1,der2)
        res=minimize(f,0.01,args=(a)) #f is the objective
        print(res)
        x1i=x1i-res.x*der1
        x2i=x2i-res.x*der2
        print(x1i,x2i)
        temp=np.array([x1i,x2i])
        xo=np.column_stack([xo,temp])
        t=res.x
    return xo
```

Newton's Method

:

```
x10=0
x20=3
x1a=np.array([x10])
x2a=np.array([x20])
o=np.array([objective(x10,x20)])
for i in range(3):
    dx=-1*l*(hess(x10,x20))@der(x10,x20).T
    x10=x10+dx[0][0]
    x20=x20+dx[1][0]
    x1a=np.append(x1a,x10)
    x2a=np.append(x2a,x20)
    o=np.append(o,objective(x10,x20))
```

Q3] $f = \sum_{i=1}^{n-1} \sum_{j=1}^{n} f_{ij} = \sum_{i=1}^{n-1} \sum_{j=1}^{n} \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}}$

① Convexity:

For convexity:

$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y) \quad \forall \lambda \in [0,1]$

$\therefore$ Consider LHS−RHS:

$= \frac{\lambda}{r_x^{12}} + \frac{1-\lambda}{r_y^{12}} - \left[\frac{1}{(\lambda r_x + (1-\lambda) r_y)^{12}}\right]$

We can ascertain that LHS will have a lower value in the denominator than RHS. Thus, its inverse will be higher.

$\therefore$ LHS − RHS ≥ 0

Also: $\frac{d^2 f}{dr_{ij}} = \frac{156}{r_{ij}^{14}} > 0 \Rightarrow H(f(x))$ is positive semi definite.

② Concavity:

Consider $g(x)$ as $\frac{2}{r_{ij}^{6}}$

$\therefore \lambda g(x) + (1-\lambda) g(y) \geq g(\lambda x + (1-\lambda) y) \quad \forall \lambda \in [0,1]$

LHS − RHS $= \frac{2\lambda}{r_x^{6}} + \frac{2(1-\lambda)}{r_y^{12}} - \left[\frac{2}{(\lambda r_x + (1-\lambda) r_y)^{12}}\right]$

Similar to ①, LHS $\geq$ RHS

$\therefore g(x)$ is convex.

Also: $\frac{d^2 g}{dr_{ij}} = \frac{84}{r_{ij}^{8}} \Rightarrow H(g(x))$ is positive semi definite.

$\Rightarrow g(x)$ is convex.

We have $f(x) = -g(x)$

$\therefore f(x) = \frac{-2}{r_{ij}^{6}}$ is concave.

③ For minimizer of f_{ij}:

$$f_{ij} = \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}}, \quad r_{ij} \geq 0 \ldots \lambda_1$$

Lagrangian: $L = \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}} + \lambda_1 (r_{ij})$

Using KKT-1:

$$\frac{dL}{dr_{ij}} = \nabla_r L = 0$$

$$\therefore \frac{-12}{r_{ij}^{13}} + \frac{12}{r_{ij}^{7}} = 0 \ldots (A)$$

And Using KKT-2:

$$\lambda_1 r_{ij} = 0 \Rightarrow \lambda_1 = 0.$$

$$\Rightarrow \frac{12}{r_{ij}^{7}} = \frac{12}{r_{ij}^{13}}$$

$$\therefore r_{ij} = 1$$

From this, $f_{ij} = 1 - 2 = -1$

$\therefore$ Minimizer occurs at:

$r_{ij}^* = 1, \ f_{ij}^* = -1$.

④ $\therefore \frac{d^2 f}{dr_{ij}^2} = \frac{156}{r_{ij}^{14}} - \frac{84}{r_{ij}^{8}}$

To check convexity:

$$\frac{d^2 f}{dr_{ij}^2} \geq 0$$

$$\therefore \frac{-156}{r_{ij}^{14}} \leq \frac{84}{r_{ij}^{8}}$$

$$\therefore r_{ij}^6 \leq \frac{13}{7} \Rightarrow r_{ij} \leq \left(\frac{13}{7}\right)^{1/6}$$

And for concavity:

$$r_{ij} \geq \left(\frac{13}{7}\right)^{1/6} \Rightarrow \text{Hence, proved.}$$

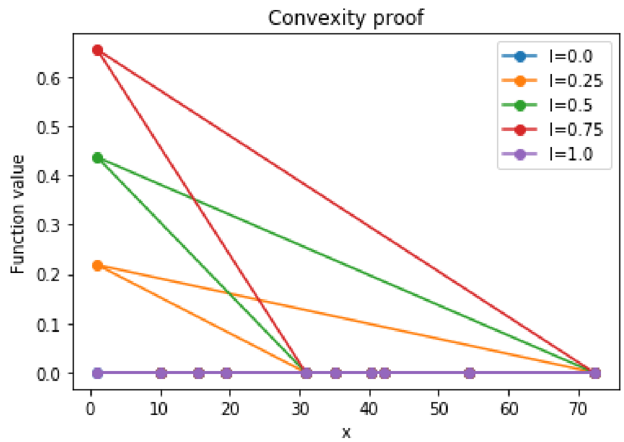
Relevant Graphical Verifications:

For random values of x,y and lambda:

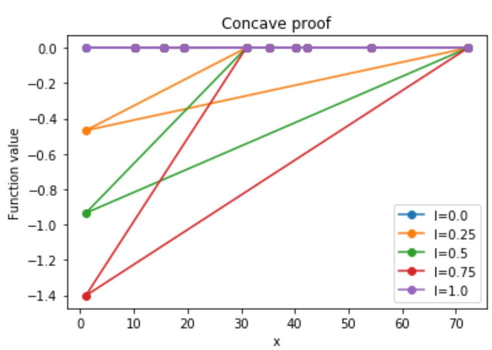
Plotting the function:

```python
def convex(x,y,l):
    p=l*f(x)+(1-l)*f(y)-f(l*x+(1-l)*y)
    return p
```

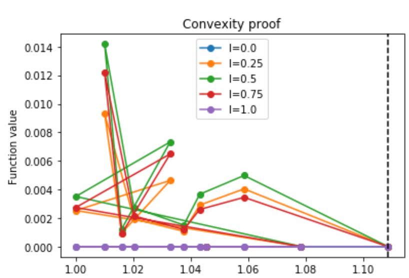
a) Function is always non-negative

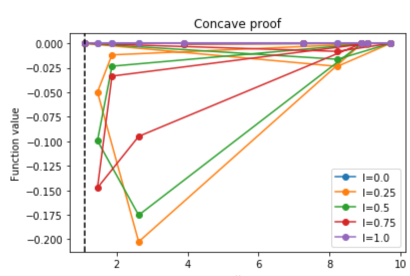


b) Function is always non-positive



d) Boundary of separation





The function value is positive for r<(13/7)^1/6 and negative for the rest.

Hence, it matches the derived result.