Final Project

Abdel-Kader KABA

June 2020

1 Part A

(1) We want to solve the following equation :

$$\min_{x \in \mathbf{R}^n} \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle \quad s.t \quad Cx = d \tag{1}$$

The Lagrangian of this problem is

$$L(x,\lambda) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle + \langle \lambda, Cx - d\rangle$$

And the dual criterion is

$$D(\lambda) = -\frac{1}{2}\langle A^{-1}(C^T\lambda - b), C^T\lambda - b\rangle - \langle \lambda, d\rangle$$

and the steps in Uzawa algorithm are :

$$x^k = A^{-1}(b - C^T\lambda^k)$$
$$\lambda^{k+1} = \lambda^k + \tau(Cx - d)$$

Please find a copy of our code below :

```
def uzawa(matA,matC,vectb,vectd,tau,tol,m,n):
    max_iterations = 1000
    lamb = np.zeros(m)
    x = np.ones(n)
    x_list = [x]
    dist = 1
    dist_list = []
    i = 0
    while i < max_iterations and dist > tol :
        #import ipdb; ipdb.set_trace()
        x_next = np.matmul(np.linalg.inv(matA),(vectb-np.matmul(matC.T,lamb)))
        x_list.append(x_next)
        dist = np.linalg.norm(x_list[-1]-x_list[-2])
        dist_list.append(dist)
        lamb += tau*(np.matmul(matC,x_next)-vectd)
        i += 1
    print("number of iterations",i)
    return dist_list,i,x_list
```

We illustrate that a sufficient condition for Uzawa algorithm is $0 < \tau < \frac{2m_A}{M_C}$. We compare the convergence of Uzawa i.e the distance between x^{k+1} and x^k when τ is smaller than $\frac{2m_A}{M_C}$ and when τ is greater than $\frac{2m_A}{M_C}$. Cf figure 1 and figure 2

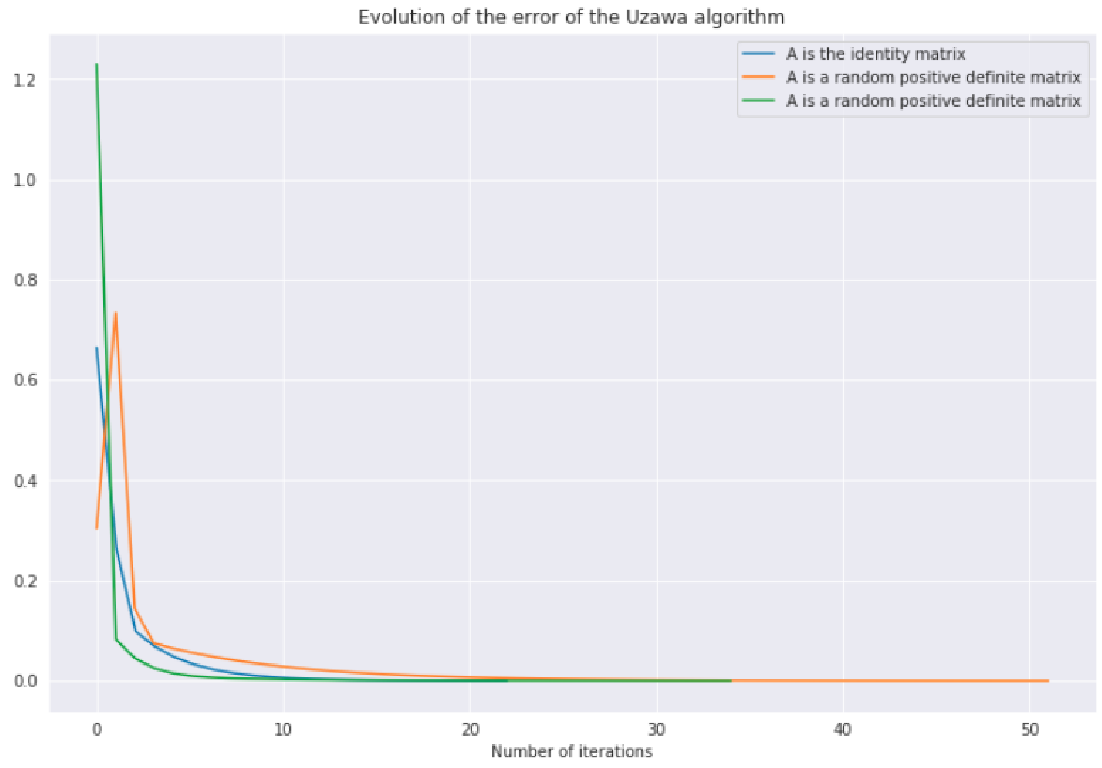


Figure 1: We observe for all our examples that the error decreases hence Uzawa algorithm converges when $0 < \tau < \frac{2m_A}{M_C}$

(2) Now we implement the augmented Lagrangian algorithm. It writes as

$$L_\beta(x,\lambda) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle + \langle \lambda, Cx - d\rangle + \frac{\beta}{2}\|Cx - d\|^2$$

And the unique solution is given by

$$x = (A + \beta C^T C)^{-1}(b - C^T\lambda + \beta C^T d)$$

We provide a copy of our code below :

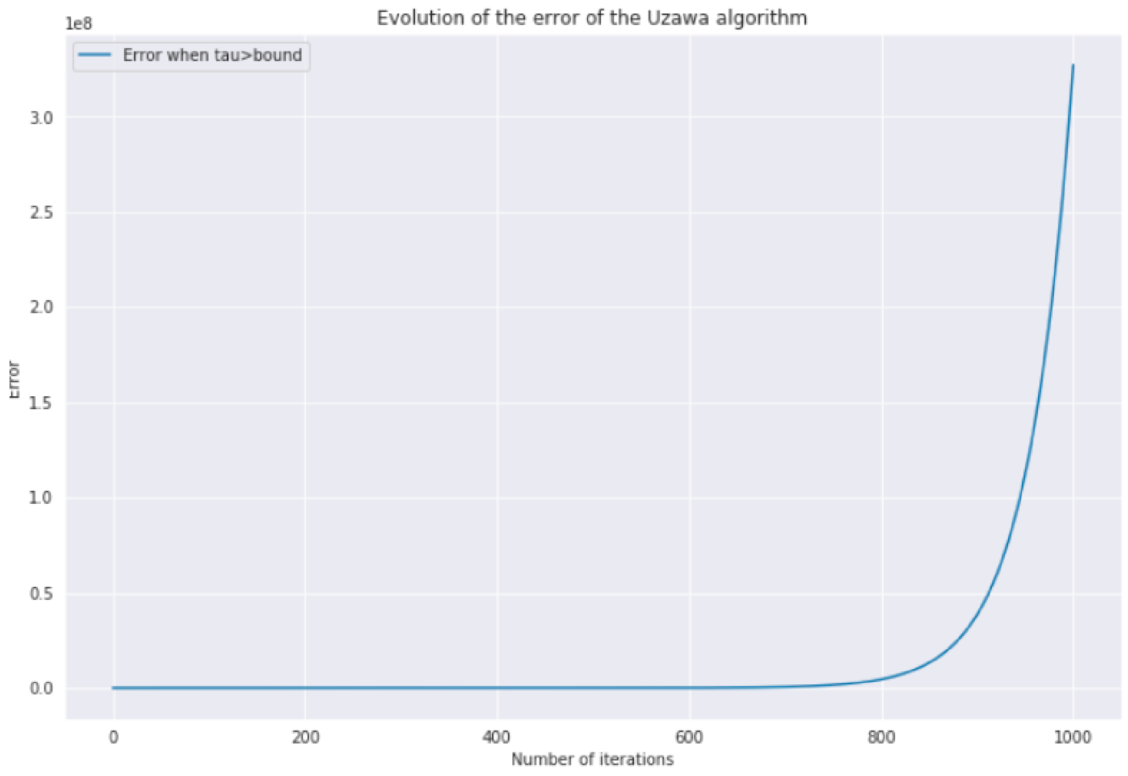


Figure 2: The error goes to infinity, When the step size is bigger than $\frac{2m_A}{M_C}$ we do not have convergence

```
def alagrangian(matA,matC,vectb,vectd,beta,tol,m,n):
    max_iterations = 1000
    lamb = np.zeros(m)
    x = np.ones(n)
    x_list = [x]
    dist = 1
    dist_list = []
    i = 0
    while i < max_iterations and dist > tol :
        #import ipdb; ipdb.set_trace()
        addi = matA+beta*np.matmul(matC.T,matC)
        fact = vectb-np.matmul(matC.T,lamb)+beta*np.matmul(matC.T,vectd)
        x_next = np.matmul(np.linalg.inv(addi),fact)
        x_list.append(x_next)
        dist = np.linalg.norm(x_list[-1]-x_list[-2])
        dist_list.append(dist)
        lamb += beta*(np.matmul(matC,x_next)-vectd)
        i += 1
    print("number of iterations",i)
    return dist_list,i,x_list
```

First we compare the result of the augmented Lagrangian with Uzawa algorithm when $\beta = 1$

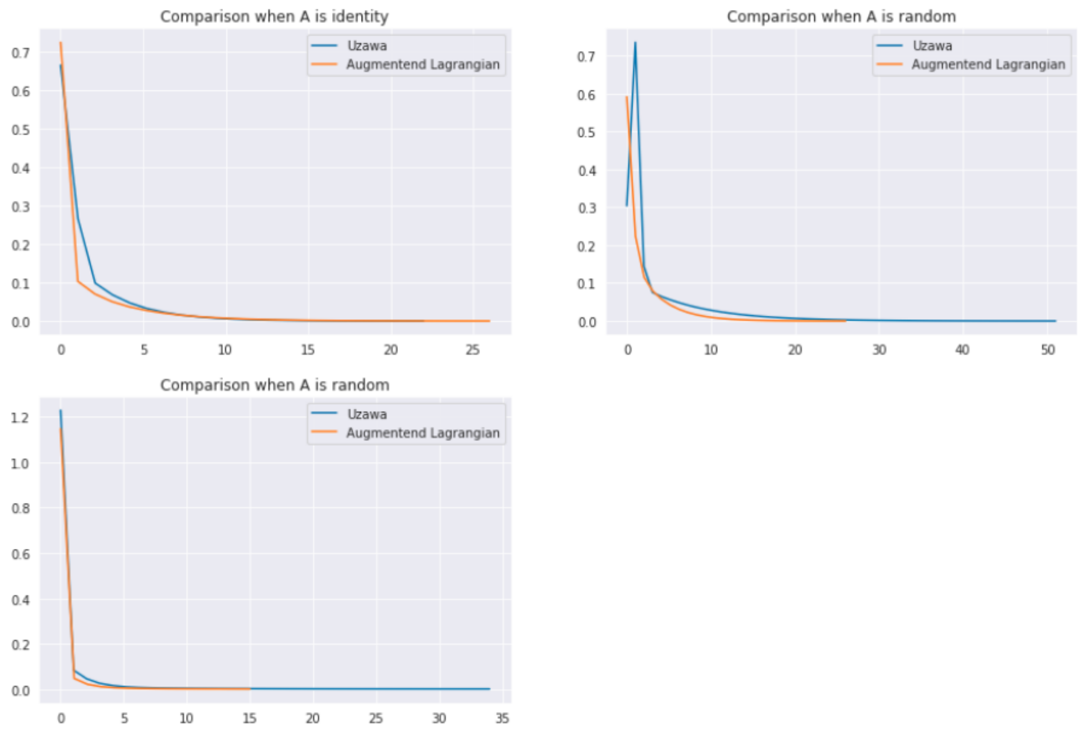


Figure 3: The augmented Lagrangian converges faster than Uzawa Algorithm

Now we increase the value of beta and see how the augmented Lagrangian behaves. We observe two examples, when A is the identity in figure 4 and when A is random in figure 5.

2 Part B

(1) We want to compute the projection onto the set $K = \{(u_1, ..., u_n) \in \mathbb{R}^n, u_{i+1} \geq u_i 1 \leq i < n\}$ From the course we know that it is equivalent to solving the problem :

$$\min_{y \in K} \frac{1}{2} \|y - x\|^2 \quad s.t \;\; y \in K$$

And from B.1), we know how to solve these types of problems. We have the following inputs for our algorithm : $A = I_n, b = -2x, d = 0, C = \begin{pmatrix} 1 & -1 & 0 & ... & 0 \\ 0 & 1 & -1 & 0 & ... \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$. We get this $(n-1) \times n$ matrix because $u_{i+1} \geq u_i$.

We get figure 6:

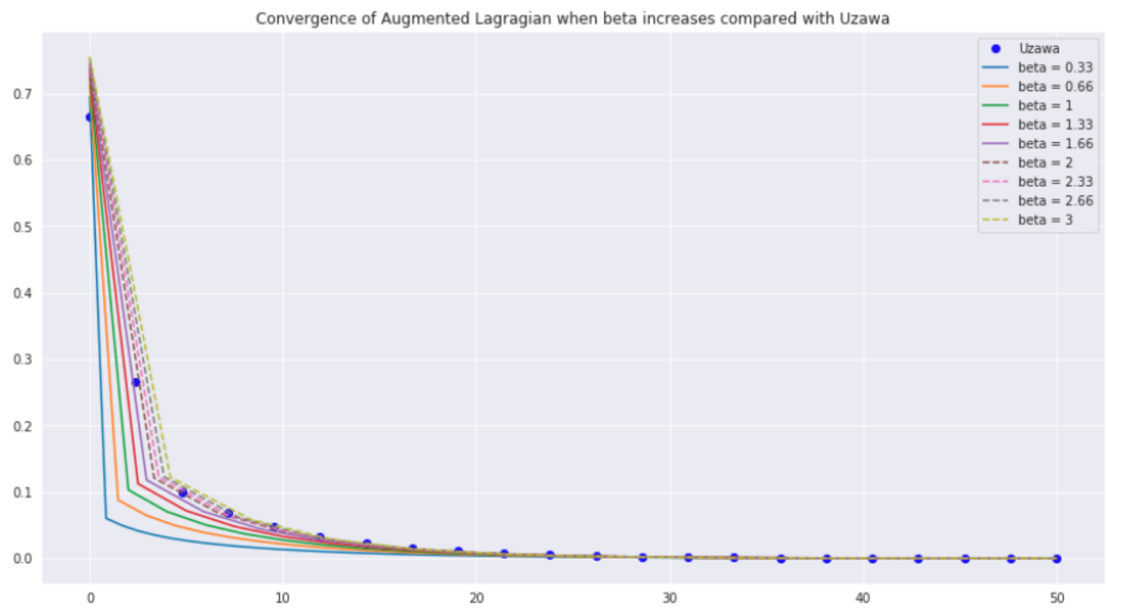


Figure 4: In this case, when Beta is greater than 1.66 the augmented Lagrangian converges slower than Uzawa algorithm

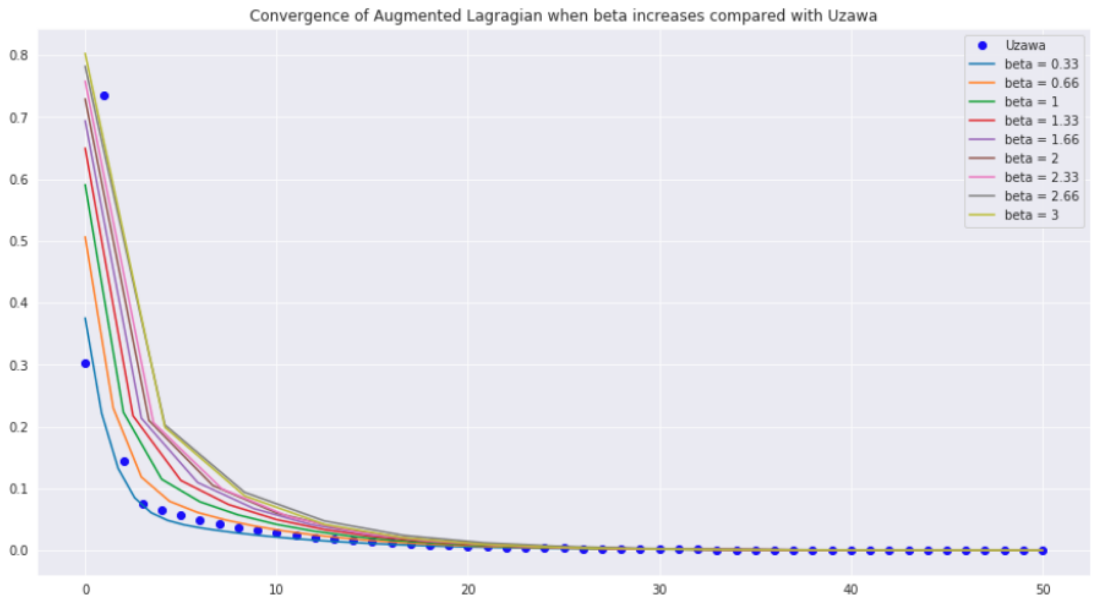
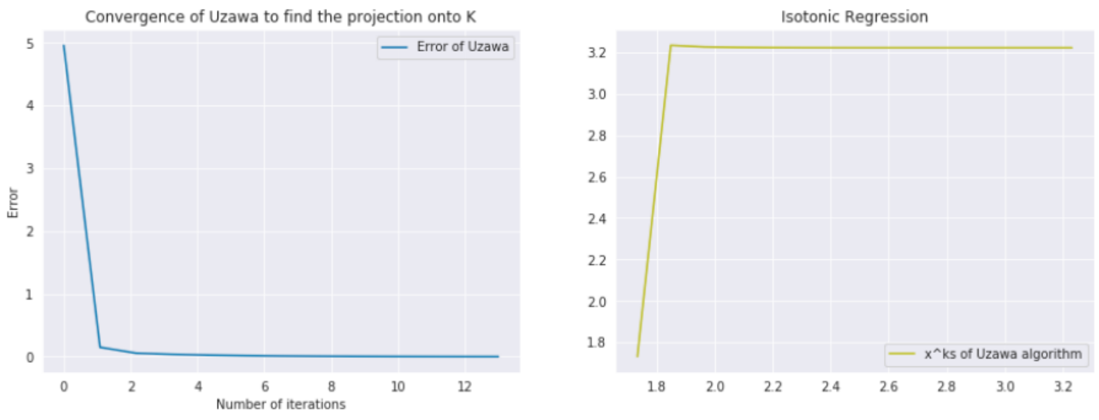


Figure 5: As Beta increases, the augmented Lagrangian converges slower and slower



```
print("The projection onto K is : ", res41[2][-1])
```

```
The projection onto K is :  [-1.85800567 -1.8593657  -1.86072574]
```

Figure 6: On the left we see that Uzawa algorithm converges. On the right is the isotonic regression given by the projected vector x. The projection onto x we find for n=3 is also shown above.

(2) We now want to find the projection onto the set $K' = \{(u_1, ...u_n) \in \mathbb{R}^n, u_i \leq \frac{1}{2}(u_{i-1} + u_{i+1})\}$ By applying the same reasoning as before we have the following inputs for our Uzawa algorithm : $A = I_n, b = -2x, d = 0, C = \begin{pmatrix} -0.5 & 1 & -0.5 & ... & 0 \\ 0 & 0 & -0.5 & 1 & -0.5 \\ 0 & 0 & 0 & -0.5 & 1 \end{pmatrix}$.

We get figure 7

3 Part C

We implemented the subgradient algorithm to solve exercise 5.4. Please find our code below

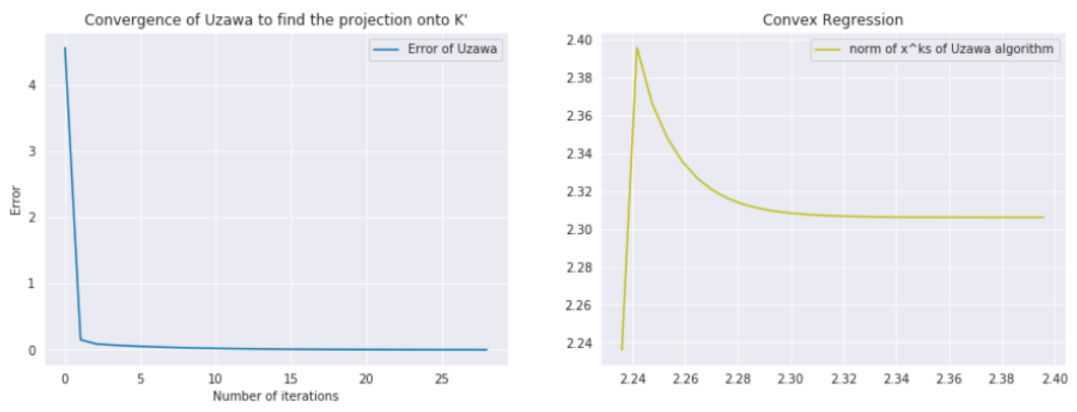
```
def subgradient(tol,n):
max_iterations = 1200
x = np.random.rand(n)
points = [np.random.rand(n) for i in range(n)]
x_list = [x]
dist = 1
dist_list = []
i = 1
while i < max_iterations and dist > tol :
    tau = 1/i
    norms = np.array([np.linalg.norm(x-elmt)**2 for elmt in points])
    r = x - points[norms.argmax()]
    x_next = x - tau*(r/np.linalg.norm(r))
    x_list.append(x_next)
    dist = np.linalg.norm(x_list[-1]-x_list[-2])
```

```
        dist_list.append(dist)
        x = x_next
        i += 1
    print("number of iterations",i)
    return dist_list,i,x_list,points
```

We observe that our algorithm indeed converges, even though it takes more iteration (about 1001) than Uzawa and the gradient method algorithm to converge. We see the rate of converge in figure 8

We now apply our algorithm for $n = 2$ to find the optimal point. Our results are presented in figure 9

We can see that it is in the convex hull of x_1, x_2 and it is equidistant to x_1, x_2 ($x_1 \approx (0.13, 0.88)$ and $x_2 \approx (0.10, 0.57)$ thus $\frac{1}{2}(x_1 + x_2 = (0.115, 0.725) \approx x^*$



```
print("The projection onto K' is : ", res51[2][-1])
```

```
The projection onto K' is :  [-1.0710692  -1.33716733 -1.10097716 -0.8697952  -0.64362144]
```

Figure 7: On the left we see that Uzawa algorithm converges. On the right is the isotonic regression given by the projected vector x. The projection onto x we find for n=5 is also shown above.

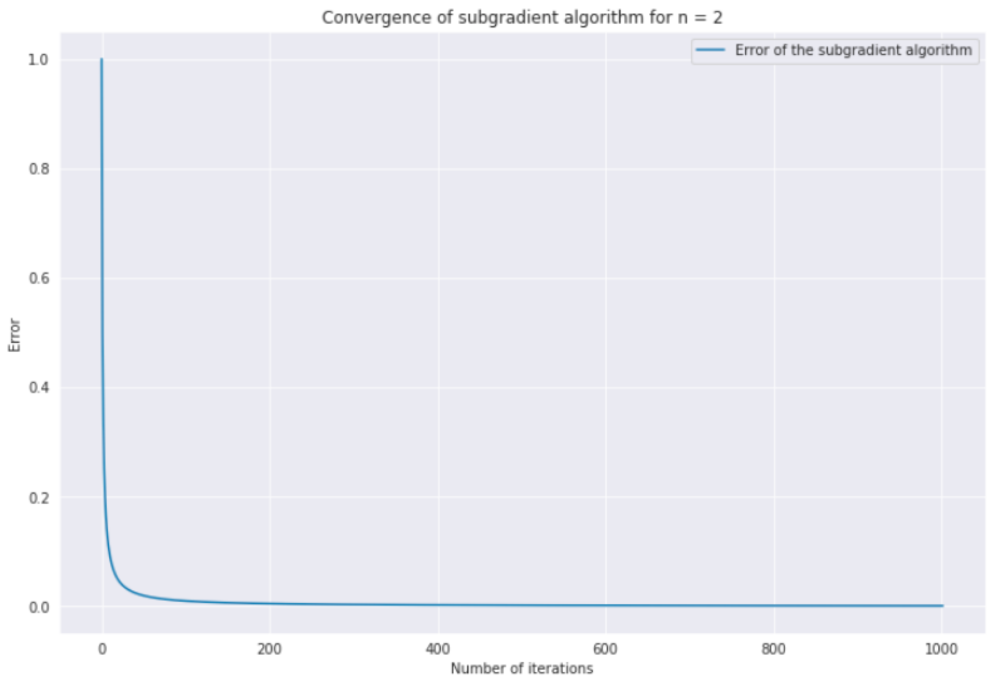
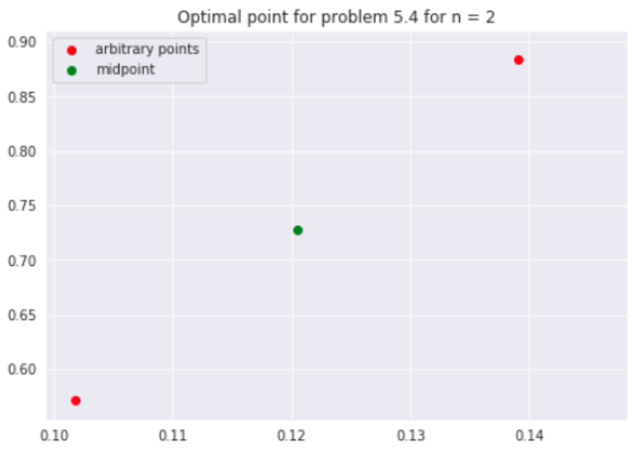


Figure 8: Subgradient algorithm converge after 1000 iterations



```
Coordinates of the points shown above :  [array([0.13911283, 0.88319212]), array([0.10179604, 0.
57139942]), array([0.12045456, 0.72729677])]
```

Figure 9: The optimal point in green satisfies all the properties shown in exercise 5.4