Homework 1 Sol. (OCDR)

Universidad Nacional de Colombia

By:

- Santiago Cadena Álvarez (Electrical and Electronic Engineering student)
- J-J-J. David S. Last. (Electrical Engineering student)

To:

Eng. Eduardo A. Mojica - Nava

1) Unconstrained Optimization

1. Consider Rosenbrock's test function

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

Find the minimum point for this function from the initial estimate $x^0 = [-1, 1]^{\mathrm{T}}$, using the Newton method and the steepest desccent method impemented in Matlab (or Python). Compare the results.

Trvially... The solution is $(x_1, x_2) = (1, 1)$. So, there are two options....

Newton's method requires the gradient (first derivative) and the Hessian (second derivative) of the function.

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k)$$

The steepest descent method updates the current estimate based on the negative gradient of the function.

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Where $\alpha_k$is the step size, which can be determined using a line search.

Hance, using MATLAB...

```
% Define the Rosenbrock function, its gradient, and its Hessian
f = @(x) 100*(x(2) - x(1)^2)^2 + (1 - x(1))^2;
grad = @(x) [-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1)); 200*(x(2)-x(1)^2)];
hessian = @(x) [1200*x(1)^2-400*x(2)+2, -400*x(1); -400*x(1), 200];

% Initial estimate
x0 = [-1; 1];

% Newton's method
x_newton = x0;
for i = 1:100
    x_newton = x_newton - hessian(x_newton) \ grad(x_newton);
end

% Steepest descent method
x_sd = x0;
```

```
alpha = 0.001; % constant step size for simplicity
for i = 1:10000
    x_sd = x_sd - alpha * grad(x_sd);
end

% Display results
disp('Optimal solution using Newton''s method:');
```

```
Optimal solution using Newton's method:
```

```
disp(x_newton);
```

```
     1
     1
```

```
disp('Optimal solution using Steepest Descent method:');
```

```
Optimal solution using Steepest Descent method:
```

```
disp(x_sd);
```

```
    0.9925
    0.9850
```

Results:

- **Newton's Method.**
- $(x_1, x_2) = (1, 1)$
- **Steepest descent Method**
- $(x_1, x_2) = (0.9925, 0.985)$

Newton's Method has converged to the exact minimum of the Rosenbrock's function, which is at $f(x) = 0$, $x_1 = 1$ and $x_2 = 1$. This is expected as Newton's method, when applicable, can be very efficient and converge quadratically to the solution, especially when started close to the optimal point.

Steepest descent Method has come close to the minimum but hasn't reached it exactly... The steepest descent method, while more general and applicable to a wider range of problems, often converges more slowly than methods like Newton's, especially for ill-conditioned problems like the Rosenbrock's function. The constant step size chosen in the implementation (1×10^{-3}) is a simple choice, and in practice, a line search or adaptive step size might be used to improve convergence.

2. Let $f(x) = \frac{1}{2} x^{\mathrm{T}} A x - b^{\mathrm{T}} x$ with

$$A = \begin{bmatrix} 6 & 13 & -17 \\ 13 & 29 & -38 \\ -17 & -38 & 50 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad x^0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Show that f is strictly convex and find the global minimum using the optimality conditions. Use the diagonally scaled steepest descent implemented in Matlab (or Python) to find an iterative solution. Compare the results.

So, A matrix is positive definite if all its eigenvalues are positive. We'll compute the eigenvalues of A to check its definiteness.

For a quadratic function.

$$\nabla f(x) = Ax - b$$

Setting the gradient to zero.

$$Ax^* = b$$

Where x^* is the optimal solution.

In the diagonally scaled steepest descent method, the step size is scaled by the inverse of the diagonal elements of the matrix A.

$$x_{k+1} = x_k - D\nabla f(x_k)$$

Where D is a diagonal matrix with the inverse of the diagonal elements of A.

Implementing in MATLAB...

```
% Given data
A = [6 13 -17; 13 29 -38; -17 -38 50];
b = [1; 2; 3];
x0 = [1; 0; 0];

% Check if A is positive definite
eigenvalues = eig(A);
is_positive_definite = all(eigenvalues > 0);

% Find global minimum using optimality conditions
x_optimal = A \ b;

% Diagonally scaled steepest descent
D = diag(1./diag(A));
x_sd = x0;
for i = 1:100
    gradient = A*x_sd - b;
    x_sd = x_sd - D*gradient;
end

% Display results
disp('Is A positive definite?');
```

```
Is A positive definite?
```

```
disp(is_positive_definite);
```

```
   1
```

```
disp('Optimal solution using optimality conditions:');
```

```
Optimal solution using optimality conditions:
```

```
disp(x_optimal);
```

```
   -5.0000
   39.0000
   28.0000
```

```
disp('Solution using diagonally scaled steepest descent:');
```

```
Solution using diagonally scaled steepest descent:
```

```
disp(x_sd);
```

```
   1.0e+29 *

    1.2340
    0.5644
   -0.4293
```

Results.

- Is A positive definite?
- 1(Yes...)
- Optimal solution using optimality conditions.
- $x^* = \begin{bmatrix} -5 \\ 39 \\ 28 \end{bmatrix}$
- Solution using diagonally scaled steepest descent.
- $x = 1 \times 10^{29} \begin{bmatrix} 1.234 \\ 0.5644 \\ -0.4293 \end{bmatrix}$

Hence,... Yes (as indicated by the value 1). This means that the function is strictly convex. The solution using optimality conditions is the exact global minimum of the function since the function is strictly convex.

Talking about the solution using diagonally scaled steepest descent... The values are extremely large, indicating that the method has diverged. This can happen with the steepest descent method, especially for ill-conditioned problems or if the step size is not chosen appropriately. The diagonally scaled steepest descent method uses the inverse of the diagonal elements of A as the step size, but this might not always guarantee convergence for all problems...

2) Constrained Optimization

3. Consider the problem

$$\min f(x) = -\frac{1}{3}\sum_{i}^{n} x_i^3$$

$$\text{s.t. } \sum_i^n x_i = 0$$

$$\sum_i^n x_i^2 = n,\ i \in \{1, 2, \cdots, n\}$$

where $n > 2$. Write down the first and second order KKT optimality conditions. Verify that they are satisfied at an optimum (minimizers or maximizers? justify).

So, the constraints are.

$$h_1(x) = \sum_i^n x_i = 0$$

$$h_2(x) = \sum_i^n x_i^2 - n = 0$$

Using the Karush-Kuhn-Tucker (KKT) conditions, set up the Lagrangian function, which incorporates the objective function and the constraints using Lagrange multipliers.

$$L(x, \lambda_1, \lambda_2) = -\frac{1}{3}\sum_{i=1}^n x_i^3 + \lambda_1 \sum_{i=1}^n x_i + \lambda_2 \left(\sum_{i=1}^n x_i^2 - n \right)$$

The gradient of the Lagrangian with respect to x and the Lagrange multipliers should be zero.

$$\frac{\partial L(x, \lambda_1, \lambda_2)}{\partial x_i} = -x_i^2 + \lambda_1 + 2\lambda_2 x_i = 0$$

The Hessian of the Lagrangian with respect to x should be positive definite for a minimum and negative definite for a maximum.

$$\frac{\partial^2 L(x, \lambda_1, \lambda_2)}{\partial x_i^2} = -2 + 2\lambda_2$$

Check the second order condition. If the Hessian is positive definite, then the solution is a minimizer. If it's negative definite, then it's a maximizer. Using the first constraint, the sum of all x_i is zero, the coefficient of λ_1 in the gradient of the Lagrangian is zero.

$$\lambda_1 = 0$$

Then.

$$x_i^2 - 2\lambda_2 x_i = 0$$

$$x_i(x_i - 2\lambda_2) = 0$$

Either $x_i = 0$ or $x_i = 2\lambda_2$.If we assume that m out of n values of x_i are non-zero.

$m(2\lambda_2) = 0$

$\lambda_2 = 0$

This leads to a contradiction since all x_i would be zero, but the second constraint states that $\sum_{i=1}^{n} x_i^2 = n$, and $n > 2$.Therefore $x_i = 2\lambda_2$. Using the second constraint.

$n(2\lambda_2)^2 = n$

$4\lambda_2^2 = 1$

$\lambda_2 = \frac{1}{2}$

Using the second order condition.

$$\frac{\partial^2 L(x, \lambda_1, \lambda_2)}{\partial x_i^2} = -2 + 2\left(\frac{1}{2}\right) = -1 < 0$$

Since this value is negative, it indicates that the solution is a maximizer, as said above. So, the function subject to the given constraints has a maximum at $x_i = 1 \, \forall \, i \in \{1, 2, \cdots, n\}$.

4. Form the KKT conditions for the problem and determine the solution

$$\max f(x, y) = (x + 1)^2 + (y + 1)^2$$

$$\text{s. t. } x^2 + y^2 \leq 2$$

$$y \leq 1$$

The constraints are.

$g_1(x, y) = x^2 + y^2 - 2 \leq 0$

$g_2(y) = y - 1 \leq 0$

Set up the Lagrangian function.

$L(x, y, \mu_1, \mu_2) = (x + 1)^2 + (y + 1)^2 + \mu_1(x^2 + y^2 - 2) + \mu_2(y - 1)$

Gradient of the Lagrangian with respect to xx, yy, and the Lagrange multipliers should be zero.

$$\frac{\partial L(x, y, \mu_1, \mu_2)}{\partial x} = 2(x + 1) + 2\mu_1 x = 0$$

$$\frac{\partial L(x, y, \mu_1, \mu_2)}{\partial y} = 2(y + 1) + 2\mu_1 y + \mu_2 = 0$$

Complementary slackness.

$\mu_1 \geq 0$

$\mu_2 \geq 0$

Feasibility.

$x^2 + y^2 \leq 2$

$y \leq 1$

Analyze the constraints and the objective function.

- The constraint $x^2 + y^2 \leq 2$ is a circle of radius $\sqrt{2}$ centered at the origin.
- The constraint $y \leq 1$ is a restriction on the ordinate axis.
- The objective function is to be maximized, and it represents the squared distance from a point to the (x, y) point $(-1, -1)$.

Obtaining x and y from the KKT conditions:

$$x = \frac{-1}{1 + \mu_1}$$

$$y = \frac{-(\mu_2 + 1)}{2(1 + \mu_1)}$$

Considering different cases for λ_1 and λ_2 :

Case $\mu_1 = \mu_2 = 0.$

This means that neither of the constraints are active. This is unlikely given the nature of the problem. So-

$$x = \frac{-1}{1 + 0} = -1$$

$y = \frac{-(0 + 1)}{2(1 + 0)} = -\frac{-1}{2}$. Due to the

Then.

$$f(x, y) = (-1 + 1)^2 + (-1 + 1)^2 = 0$$

Case $\mu_1 > 0, \mu_2 = 0.$

This means the point lies on the circle $x^2 + y^2 = 2$ but not on the line $y = 1$. Then.

$$y = \frac{-(\mu_2 + 1)}{2(1 + \mu_1)} = \frac{-1}{2(1 + \mu_1)}$$

$$x = \frac{-1}{1 + \mu_1} = 2y$$

Using the constraint $x^2 + y^2 = 2.$

$$(2y)^2 + y^2 = 2$$

$$y = \sqrt{\frac{2}{5}}$$

$$\mu_1 = -\sqrt{\frac{5}{2}} - 1$$

This contradicts $\mu_1 > 0$. So, this isn't the solution.

Case $\mu_1 = 0, \mu_2 > 0$.

This means the point does not lie on the circle but lies on the line. So.

$$x = -1$$

$$y = 1$$

$$\mu_2 = -2(1) - 1 = -3$$

This contradicts $\mu_2 > 0$.

But if $y = -1,\ \mu_2 =. 2 - 1 = 1$, so it could be the solution.

$$f(x, y) = (-1 + 1)^2 + (1 + 1)^2 = 4$$

Case $\mu_1 > 0, \mu_2 > 0$**.**

This means the point lies on the intersection of the circle and the line.

$$y = 1$$

Substituting in the equation of the circle.

$$x^2 + 1 = 2$$

$$x = \pm 1$$

Hence.

$$f(x, y) = (-1 + 1)^2 + (1 + 1)^2 = 4$$

$$f(x, y) = (1 + 1)^2 + (1 + 1)^2 = 8$$

Comparing the objective function values for all cases, the maximum value of the objective function occurs at $(x^*, y^*) = (1, 1)$ (from case $\mu_1 > 0, \mu_2 > 0$), and $f(x^*, y^*) = 8$.

5. Write a program using Matlab (or Python) to find an optimal solution of the following problems using the barrier method, the penalty function method, and the method of Multipliers. Compare the results.

(a)

$$\min f(x) = -x_1 - x_2$$

$$\text{s.t. } 1 - x_1^2 - x_2^2 \geq 0$$

$$x_1 - x_2^2 \geq 0$$

Well, then.

$g_1(x) = 1 - x_1^2 - x_2^2 \geq 0$

$g_2(x) = x_1 - x_2^2 \geq 0$

The barrier method transforms the problem by adding a barrier term to the objective function for each inequality constraint. The barrier term tends to infinity as the constraint boundary is approached.

$$f_{\text{barrier}}(x, t) = -x_1 - x_2 - \frac{1}{t}(\ln(g_1(x)) + \ln(g_2(x))),\ t > 0$$

Where t is a parameter that decreases over iterations.

The penalty function method adds a penalty term to the objective function for each constraint violation. The penalty term increases as the constraint is violated more.

$$f_{\text{penalty}}(x, r) = -x_1 - x_2 + r\left(\max^2(0, -g_1(x)) + \max^2(0, -g_2(x))\right),\ r > 0$$

Where r is a penalty parameter that increases over iterations.

The method of multipliers introduces dual variables (Lagrange multipliers) for the constraints and updates them iteratively.

$$L(x, \lambda) = -x_1 + x_2 + \lambda_1 g_1(x) + \lambda_2 g_2(x) + \frac{r}{2}\left(g_1^2(x) + g_2^2(x)\right)$$

λ_i are the Lagrange multipliers.

So, the MATLAB code for these methods...

```
% Define the objective function and constraints
f = @(x) -x(1) - x(2);
g1 = @(x) 1 - x(1)^2 - x(2)^3;
g2 = @(x) x(1) - x(2)^2;

% Initial guess
x0 = [0.5; 0.5];

% Barrier method
t = 1;
for i = 1:10
    f_barrier = @(x) f(x) - (1/t) * (log(g1(x)) + log(g2(x)));
    x_barrier = fminunc(f_barrier, x0);
    t = t * 10;
```

```
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
```

```
Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>
```

```
% Penalty function method
r = 1;
for i = 1:10
    f_penalty = @(x) f(x) + r * (max(0, -g1(x))^2 + max(0, -g2(x))^2);
    x_penalty = fminunc(f_penalty, x0);
    r = r * 10;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.
```

```
fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
```

```
% Method of Multipliers
lambda = [0; 0];
r = 1;
for i = 1:10
    L = @(x) f(x) + lambda(1)*g1(x) + lambda(2)*g2(x) + (r/2) * (g1(x)^2 + g2(x)^2);
    x_multiplier = fminunc(L, x0);
    lambda = lambda + r * [g1(x_multiplier); g2(x_multiplier)];
    r = r * 10;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
```

```
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
```

```
% Display results
disp('Optimal solution using Barrier method:');
```

```
Optimal solution using Barrier method:
```

```
disp(x_barrier);
```

```
    0.7549
    0.7549
```

```
disp('Optimal solution using Penalty function method:');
```

```
Optimal solution using Penalty function method:
```

```
disp(x_penalty);
```

```
    0.7549
    0.7549
```

```
disp('Optimal solution using Method of multipliers:');
```

```
Optimal solution using Method of multipliers:
```

```
disp(x_multiplier);
```

```
    0.6710
    0.8192
```

Results

- **Berrier method.**
- $(x_1, x_2) = (0.7549, 0.7549)$
- **Penalty function method**
- $(x_1, x_2) = (0.7549, 0.7549)$
- **Method of multipliers.**
- $(x_1, x_2) = (0.671, 0.8192)$

The Barrier Method and the Penalty function Method yield the same optimal solution for this problem... This is not always the case, but it's not uncommon for these methods to converge to the same solution, especially for certain types of problems or when the constraints are not too restrictive.

The Method of Multipliers gives a slightly different solution. This could be due to the iterative nature of the method and the specific update rules used for the Lagrange multipliers and the penalty parameter. The Method of Multipliers, being an augmented Lagrangian method, combines aspects of both penalty methods and Lagrangian methods. Depending on the problem and the chosen parameters, it might converge to a different solution than the other methods...

(b)

$$\min f(x) = -x_1 + (x_2 + x_3)^2$$

$$\text{s.t. } x_1 - x_2^2 - 2e^{x_3} \geq 0$$

$$10 - x_1 - x_3^4 \geq 0$$

So.

$g_1(x) = x_1 - x_2^2 - 2e^{x_3} \geq 0$

$g_2(x) = 10 - x_1 - x_3^4 \geq$

Now, implementing the same methods as in **(a)** in MATLAB...

```
% Define the objective function and constraints
f = @(x) -x(1) + (x(2) + x(3))^2;
```

```
g1 = @(x) x(1) - x(2)^2 - 2*exp(x(3));
g2 = @(x) 10 - x(1) - x(3)^4;

% Initial guess
x0 = [5; 0; 0];

% Barrier Method
t = 1;
for i = 1:10
    f_barrier = @(x) f(x) - (1/t) * (log(g1(x)) + log(g2(x)));
    x_barrier = fminunc(f_barrier, x0);
    t = t * 10;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.
```

```
<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
```

```
% Penalty function Method
r = 1;
for i = 1:10
    f_penalty = @(x) f(x) + r * (max(0, -g1(x))^2 + max(0, -g2(x))^2);
    x_penalty = fminunc(f_penalty, x0);
    r = r * 10;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum found.
```

```
Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because the size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
```

```
% Method of multipliers
lambda = [0; 0];
r = 1;
for i = 1:10
    L = @(x) f(x) + lambda(1)*g1(x) + lambda(2)*g2(x) + (r/2) * (g1(x)^2 + g2(x)^2);
    x_multiplier = fminunc(L, x0);
    lambda = lambda + r * [g1(x_multiplier); g2(x_multiplier)];
    r = r * 10;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.
```

```
Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Solver stopped prematurely.

fminunc stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 3.000000e+02.


Solver stopped prematurely.

fminunc stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 3.000000e+02.


Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
```

```
% Display results
disp('Optimal solution using Barrier method:');
```

```
Optimal solution using Barrier method:
```

```
disp(x_barrier);
```

```
   10.0000
         0
         0
```

```
disp('Optimal solution using Penalty function method:');
```

```
Optimal solution using Penalty function method:
```

```
disp(x_penalty);
```

```
   10.0000
         0
         0
```

```
disp('Optimal solution using Method of multipliers:');
```

```
Optimal solution using Method of multipliers:
```

```
disp(x_multiplier);
```

```
    7.2496
   -0.0000
    1.2878
```

Results:

- **Berrier method.**
- $(x_1, x_2, x_3) = (10, 0, 0)$
- **Penalty function method.**
- $(x_1, x_2) = (10, 0, 0)$
- **Method of multipliers.**
- $(x_1, x_2) = (7.2496, 0, 1.2878)$

Well, as expected, the conclusions are the same as in (a),

6. Consider the problem.

$$\min f(x) = \sum_{i}^{n} \left(x_i + e^{-x_i}\right)$$

$$\text{s.t}, g_i(x) = 1 - x_i, \; i \in \{1, \cdots, n\}$$

Using the Dual problem and the optimality conditions, find an optimal solution of the primal problem.

So, the Lagrangian function.

$$L(x, \mu) = \sum_{i=1}^{n} \left(x_i + e^{-x_i}\right) + \sum_{i=1}^{n} \mu_i (1 - x_i)$$

So, x_i can b e expressed in terms of λ_i. Now, get the function $q(\lambda)$ substituing the expression for x_i into the Lagrangian.

$$\max q(\lambda)$$

$$\text{s.t}, \lambda_i \geq 0,\ i \in \{1, \cdots, n\}$$

The KKT condtions for the problem are:

Lagrange optimality:

$$x^* = \arg\min_{x \epsilon R} L(x, \mu^*)$$

$$\nabla L(x, \mu) = 0$$

$$1 - e^{-x_i^*} - \mu_i = 0$$

$$x_i^* = -\ln(1 - \mu_i)$$

Primal feasibility.

$$g_i(x) = 1 - x_i \leq 0$$

$$x_i \geq 1$$

Dual feasibility.

$$\mu_i \geq 0$$

Complementary slackness.

$$\mu \cdot g_i(x) = 0$$

$$\mu_i(1 - x_i) = 0$$

As we saw that $1 - x_i \leq 0$ and $\mu_i \geq 0$, there could be two posibilities, either $\mu_i = 0$ or $x_i = 1$

Now, solving this problem according with duality:

$$L(x_i, \mu) = L(-\ln(1 - \mu_i), \mu) = \sum_{i=1}^{n} \left(-\ln(1 - \mu_i) + e^{\ln(1-\mu_i)}\right) + \sum_{i=1}^{n} \mu_i(1 + \ln(1 - \mu_i))$$

$$L = \sum_{i=1}^{n} -\ln(1 - \mu_i) + 1 - \mu_i + \sum_{i=1}^{n} \mu_i(1 + \ln(1 - \mu_i)) = \sum_{i=1}^{n} \ln(1 - \mu_i)(\mu_i - 1) + n$$

$$q(\mu) = \inf_{x_i \in \Re} L(x_i, \mu) = \sum_{i=1}^{n} \ln(1 - \mu_i)(\mu_i - 1)$$

And the problem is:

$$\max q(\mu), s.t\ \mu \geq 0$$

To accomplish that, it must ensured:

$$\frac{\mathrm{d}}{\mathrm{d}\mu} q\left(\mu_i^*\right) = 0$$

$$\frac{\mathrm{d}^2}{\mathrm{d}\mu^2} q\left(\mu_i^*\right) < 0$$

$$\frac{\mathrm{d}}{\mathrm{d}\mu} q\left(\mu_i^*\right) = \ln\left(1 - \mu_i^*\right) - 1 = 0$$

$$\mu_i^* = 1 - e^{-1}$$

$\frac{\mathrm{d}^2}{\mathrm{d}\mu^2} q\left(\mu_i^*\right) = -\frac{1}{1-\mu_i^*} < 0$, it´s true.

Now, replacing x_i, we obtain:

$$x_i^* = -\ln\left(1 - \mu_i^*\right) = 1 \ \ \forall\, i \in \{1, 2, \cdots, n\}$$

We see, that all the optimality condition are fulfilled.

The optimal value of the objective function hence is

$$f\left(x_i^*\right) = n(1 + e^{-1})$$

7. Write down the Dual problem for the convex problem and verify the primal and dual values are equal.

$$\min f(x) = e^{x_1} + 16e^{x_2}$$

$$\text{s.t.} -x_1 - 2x_2 \leq 0$$

$$x \in \mathbb{R}^2$$

Formulate the Lagrangian function by introducing the Lagrange multipliers.

$$L(x_1, x_2, \mu) = e^{x_1} + 16e^{x_2} + \mu(-x_1 - 2x_2)$$

where $\mu \geq 0$ is the Lagrange multiplier associated with the inequality constraint. Differentiate the Lagrangian with respect to x_1 and x_2 and set them to zero according to the KKT conditions:

$$\frac{\partial L(x_1, x_2, \mu)}{\partial x_1} = e^{x_1} - \mu = 0$$

$$\frac{\partial L(x_1, x_2, \mu)}{\partial x_2} = 16e^{x_2} - 2\mu = 0$$

Express x_1 and x_2 in terms of μ. Substituting these expressions into the Lagrangian will give the dual function.

$$x_1 = \ln(\mu)$$

$$x_2 = \ln\left(\frac{\mu}{8}\right)$$

Hence.

$$q(\mu) = \inf_{x_1, x_2 \in \Re} L(x_1, x_2, \mu) = L\left(\ln(\mu), \ln\left(\frac{\mu}{8}\right), \mu\right) = e^{\ln(\mu)} + 16e^{\ln\left(\frac{\mu}{8}\right)} - \mu\left(\ln(\mu) + 2\ln\left(\frac{\mu}{8}\right)\right)$$

$$q(\mu) = 3\mu - \mu(3\ln(\mu) - 2\ln(8))$$

The dual problem consists in maximing $q(\mu)$ with respect to μ, subject to $\mu \geq 0$.Hence, the following has to be acomplished:

$$\frac{\mathrm{d}}{\mathrm{d}\mu} q(\mu^*) = 0$$

$$\frac{\mathrm{d}^2}{\mathrm{d}\mu^2} q(\mu^*) < 0$$

$$\frac{\mathrm{d}}{\mathrm{d}\mu} q(\mu^*) = -3\ln(\mu^*) + 2\ln(8) = 0$$

$$\mu^* = 8^{\frac{2}{3}} = 4$$

$$\frac{\mathrm{d}^2}{\mathrm{d}\mu^2} q(\mu^*) = -\frac{3}{\mu^*} = -\frac{3}{4} < 0$$

$$q(\mu^*) = 3(4) - 4(3\ln(4) - 2\ln(8)) = 12$$

With this value, x1 and x2 are:

$$x_1 = \ln(4)$$

$$x_2 = \ln\left(\frac{1}{2}\right)$$

Now, replacing these values into f for finding the minimum value:

$$f\left(x_1^*, x_2^*\right) = e^{\ln(4)} + 16e^{\ln\left(\frac{1}{2}\right)} = 12$$

So, we verify that $q(\mu^*) = f\left(x_1^*, x_2^*\right)$ so for the convex function corroborating the weak duality theorem in which if $f^* = q^*$ so there's no gap duality