KAUNAS UNIVERSITY OF TECHNOLOGY
FACULTY OF MATHEMATICS AND NATURAL SCIENCES

Module P160B116 “Optimization methods”
Laboratory work #2 report

Lecturer
Mindaugas Šnipas

Student
Mykolas Paulauskas

KAUNAS, 2021

Contents

Student number for the first laboratory work: 14

Task 1 function #2: $f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$

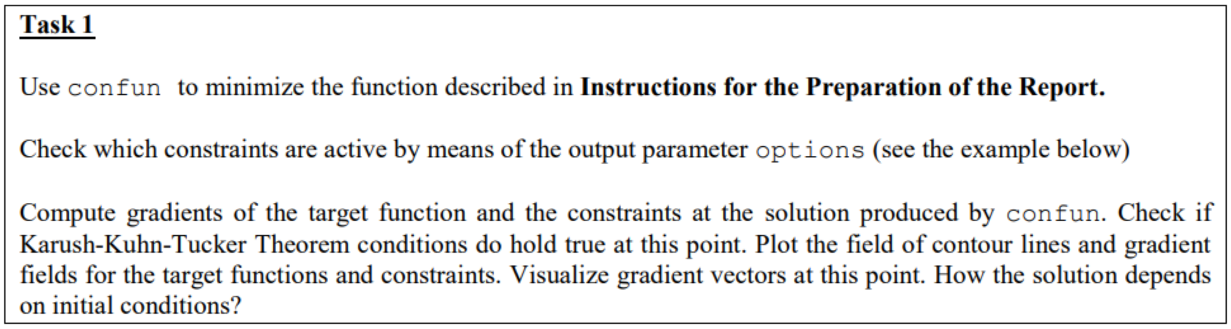
Subject to: $g_1(x,y) = x - y \leq 0$; $g_2(x,y) = x^2 + y^2 - 1.5 \leq 0$.

Figure 1 Function for part 1

Task 2 function #4: $f(x,y) = x^2 + y^2$ Subject to: $h(x,y) = x + y + 2 = 0$

Figure 2 Function for part 2

1. Task 1



Task 1

Use `confun` to minimize the function described in **Instructions for the Preparation of the Report.**

Check which constraints are active by means of the output parameter `options` (see the example below)

Compute gradients of the target function and the constraints at the solution produced by `confun`. Check if Karush-Kuhn-Tucker Theorem conditions do hold true at this point. Plot the field of contour lines and gradient fields for the target functions and constraints. Visualize gradient vectors at this point. How the solution depends on initial conditions?

Figure 3 Task 1

1.1. Results

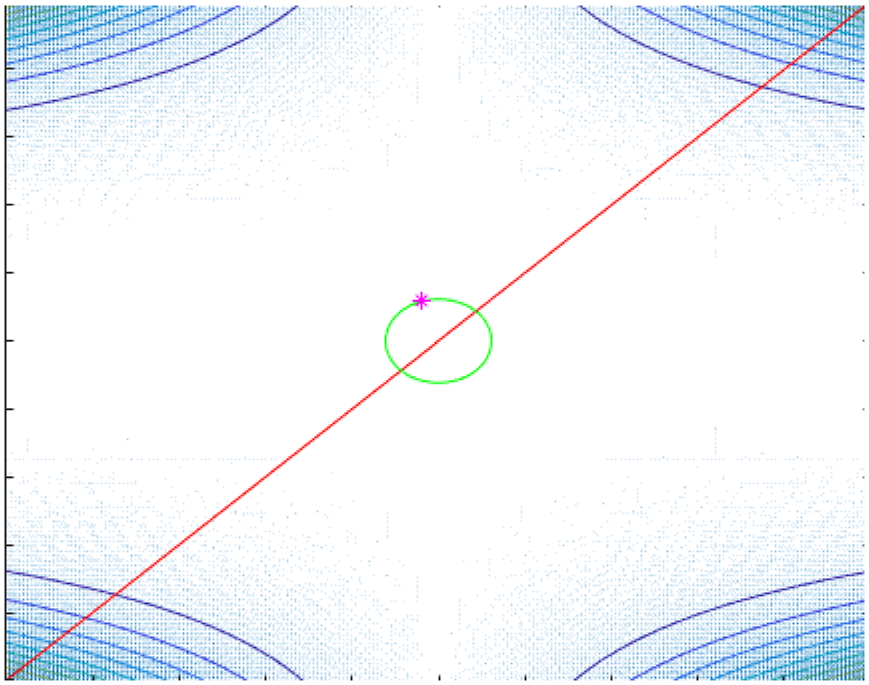


Figure 4 Results for task 1

From the figure shown above we can see both constrains graphed out the red one is g_1 and the green graph is g_2. Local minimum value, which satisfies the constraints, of the target function was computed to be (-0.4036; 1.1563). The constraint g_1 at this value is passive and the constraint g_2 is active ($g_2 = 0$).

KKT conditions apply, because with the computed gradients at the local minimum the lambda values are not all zeros and all are positive.

With different initial conditions the solution remains similar.

1.2. Matlab code

From file **objfun.m**:

```
function f = objfun(x)

f = (1.5 - x(1) + x(1) * x(2))^2 + (2.25 - x(1) + x(1) *
x(2)^2)^2 + (2.625 - x(1) + x(1) * x(2)^3)^2;
```

From file **confun.m**:

```
function [c, ceq] = confun(x)

c = [x(1) - x(2); x(1)^2 + x(2)^2 - 1.5];
```

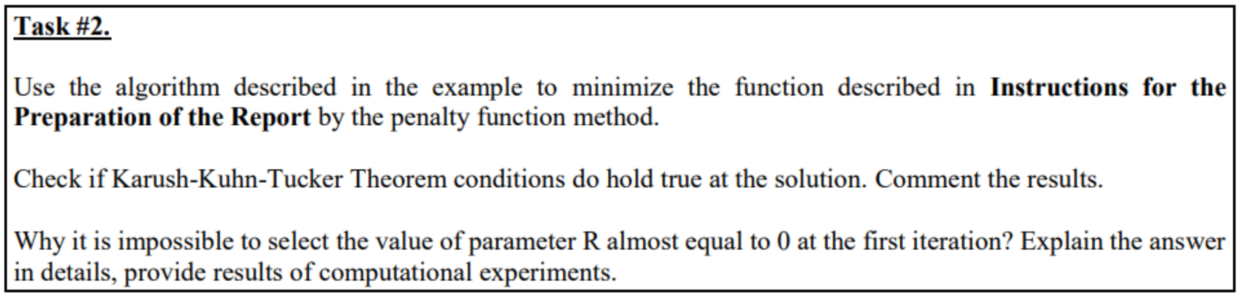
```
% since equality constraints are missing:
ceq = [];
```

From file **main.m**:

```
% use meshgrid to create a rectangular grid
[x, y] = meshgrid(-10:.1:10, -10:.1:10);

% compute function values at the points of the grid
z = (1.5 - x + x.* y).^2 + (2.25 - x + x.* y.^2).^2 + (2.625 - x
+ x.* y.^3).^2;

% compute gradients
% dx – partial derivative in respect of x; dy – partial
derivative in respect of y
[dx, dy] = gradient(z);

% plot contourlines
contour(x, y, z)

% the next plot will be constructed on top of the existing
figure
hold on

% plot gradients
quiver(x, y, dx, dy)

% the next plot will be constructed on top of the existing
figure
hold on

% plot the constraints function g1
x1 = -10:.1:10;
y1 = x1;
plot(x1, y1, 'r')

% the next plot will be constructed on top of the existing
figure
hold on

% plot the constraints functions g2 positive values
x2 = -sqrt(1.5):.00001:sqrt(1.5);
y2 = sqrt(1.5 - x2.^2);
plot(x2, y2, 'g')
```

```
% the next plot will be constructed on top of the existing
figure
hold on

% plot the constraints functions g2 negative values
plot(x2, -y2, 'g')

% make sure that the grid will stay 10x10
xlim([-10 10])
ylim([-10 10])

% make an initial guess:
x0 = [-1 1];

% Setup the optimization parameters:
% turn off large-scale algorithms
% turn on Display options for visualization of transient results
options = optimset('LargeScale', 'off', 'Display', 'iter');

% non-explicit constraints are replaced by []
[x, fval, exitflag, output] = fmincon('objfun', x0, [], [], [],
[], [], [], 'confun', options);

% the next plot will be constructed on top of the existing
figure
hold on

% plot the result X point
plot(x(1), x(2), 'm*')

% finish drawing
hold off

disp(x)
disp(fval)
disp(exitflag)
disp(output.message)
```

2. Task 2



Task #2.

Use the algorithm described in the example to minimize the function described in **Instructions for the Preparation of the Report** by the penalty function method.

Check if Karush-Kuhn-Tucker Theorem conditions do hold true at the solution. Comment the results.

Why it is impossible to select the value of parameter R almost equal to 0 at the first iteration? Explain the answer in details, provide results of computational experiments.

Figure 5 task 2

2.1. Results

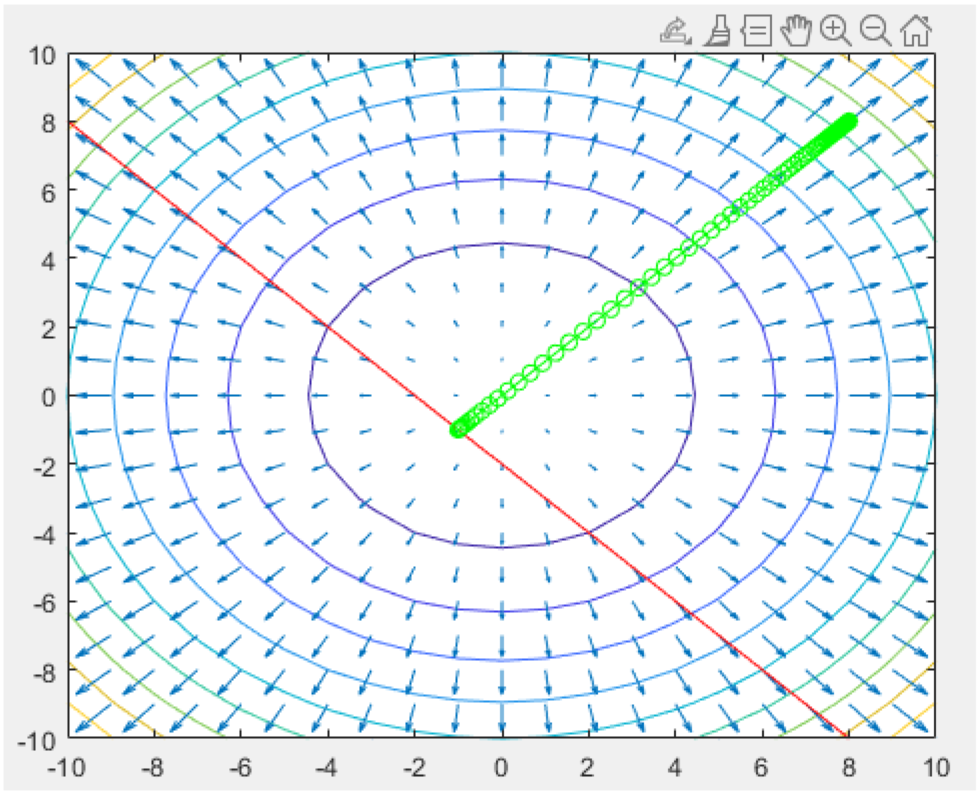


Figure 6 Results task 2

From the figure above we can see that see that the constrained minimum is reached with the value of approximately (-1, -1). At this value the constraint is active (equal to zero) there fore the KKT conditions apply.

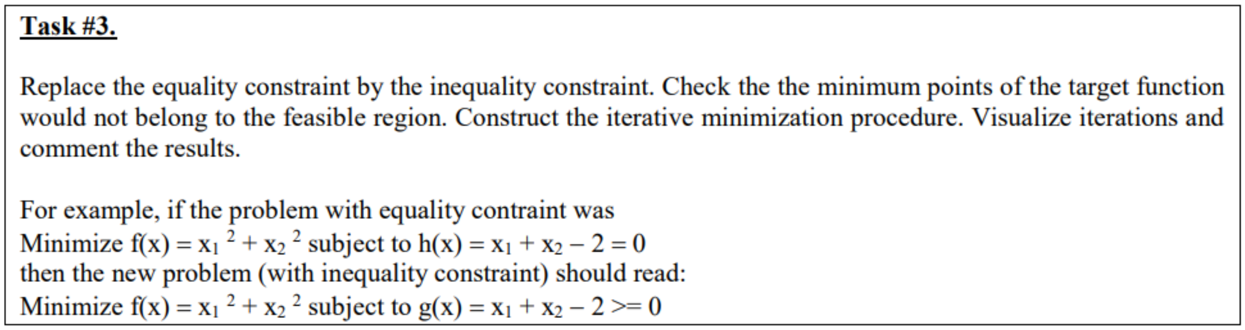
The parameter R cannot be equal to almost zero because by dividing to a value close to zero we receive a value that is very big hence our penalty function step becomes too big to compute any reliable results.

2.2. Matlab code:

```
% use meshgrid to create a rectangular grid
[x, y] = meshgrid(-10:1:10, -10:1:10);

% compute function values at the points of the grid
z = x.^2 + y.^2;

% compute gradients
% dx - partial derivative in respect of x; dy - partial
derivative in respect of y
[dx, dy] = gradient(z);

% plot contourlines
contour(x, y, z)

% the next plot will be constructed on top of the existing
figure
hold on

% plot gradients
quiver(x, y, dx, dy)

% the next plot will be constructed on top of the existing
figure
hold on

% plot the constraints function g1
x1 = -10:1:10;
y1 = -2 - x1;
plot(x1, y1, 'r')

% the next plot will be constructed on top of the existing
figure
hold on

% make sure that the grid will stay 10x10
xlim([-10 10])
ylim([-10 10])
```

```
% penalty parameter R
r = 1;

% precision
epsilon = 0.001;

% starting point
xs = [8 8];

% the step size
gamma = 0.0001;

% limit the number of iterations in order to avoid infinite
loops
max_iterations = 10000;

for i=1:max_iterations

    % gradient of the transformed function
    e1 = 2*xs(1) + 2 * (xs(1) + xs(2) + 2) / r;
    e2 = 2*xs(2) + 2 * (xs(1) + xs(2) + 2) / r;
    grad = [e1 e2];

    % compute the next point
    x = xs - gamma*grad;

    % plot the step
    plot([xs(1), x(1)], [xs(2), x(2)], 'g', [xs(1), x(1)],
[xs(2), x(2)], 'go')

    % distance between the previous and current point
    distance = sqrt((x(1) - xs(1))^2 + (x(2) - xs(2))^2);

    % refresh the variable
    xs = x;

    % update the parameter R
    r = r/5;

    % Check if the current minimum point did move from the
previous minimum
    % point less than epsilon
    if distance < epsilon
        break;
    end
end
```

3. Task 3


Task #3.

Replace the equality constraint by the inequality constraint. Check the the minimum points of the target function would not belong to the feasible region. Construct the iterative minimization procedure. Visualize iterations and comment the results.

For example, if the problem with equality contraint was
Minimize $f(x) = x_1^{\ 2} + x_2^{\ 2}$ subject to $h(x) = x_1 + x_2 - 2 = 0$
then the new problem (with inequality constraint) should read:
Minimize $f(x) = x_1^{\ 2} + x_2^{\ 2}$ subject to $g(x) = x_1 + x_2 - 2 >= 0$

Figure 7 Task 3

3.1. Results

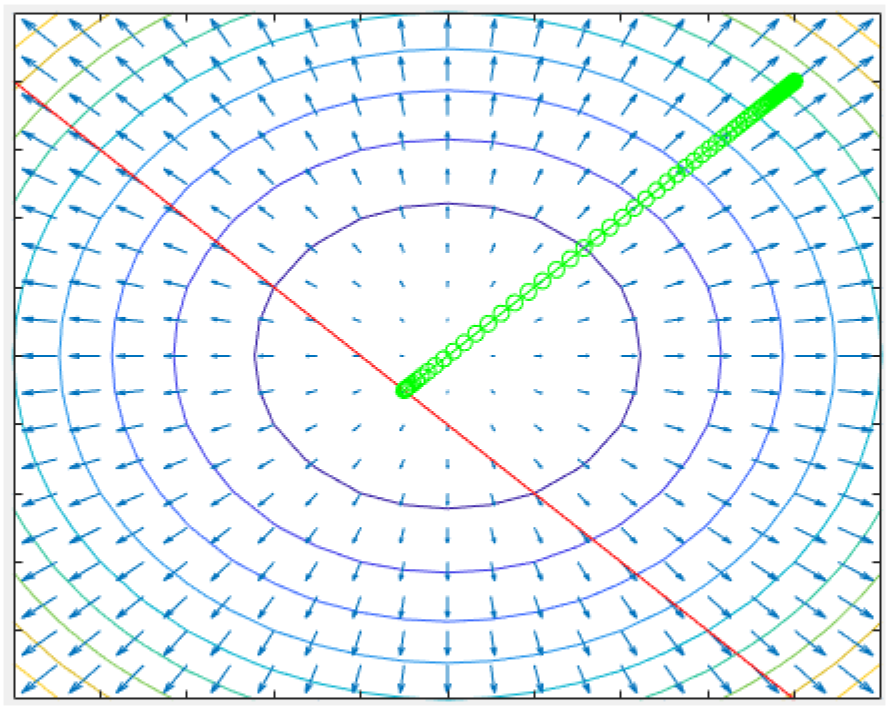


Figure 8 Results of task 3

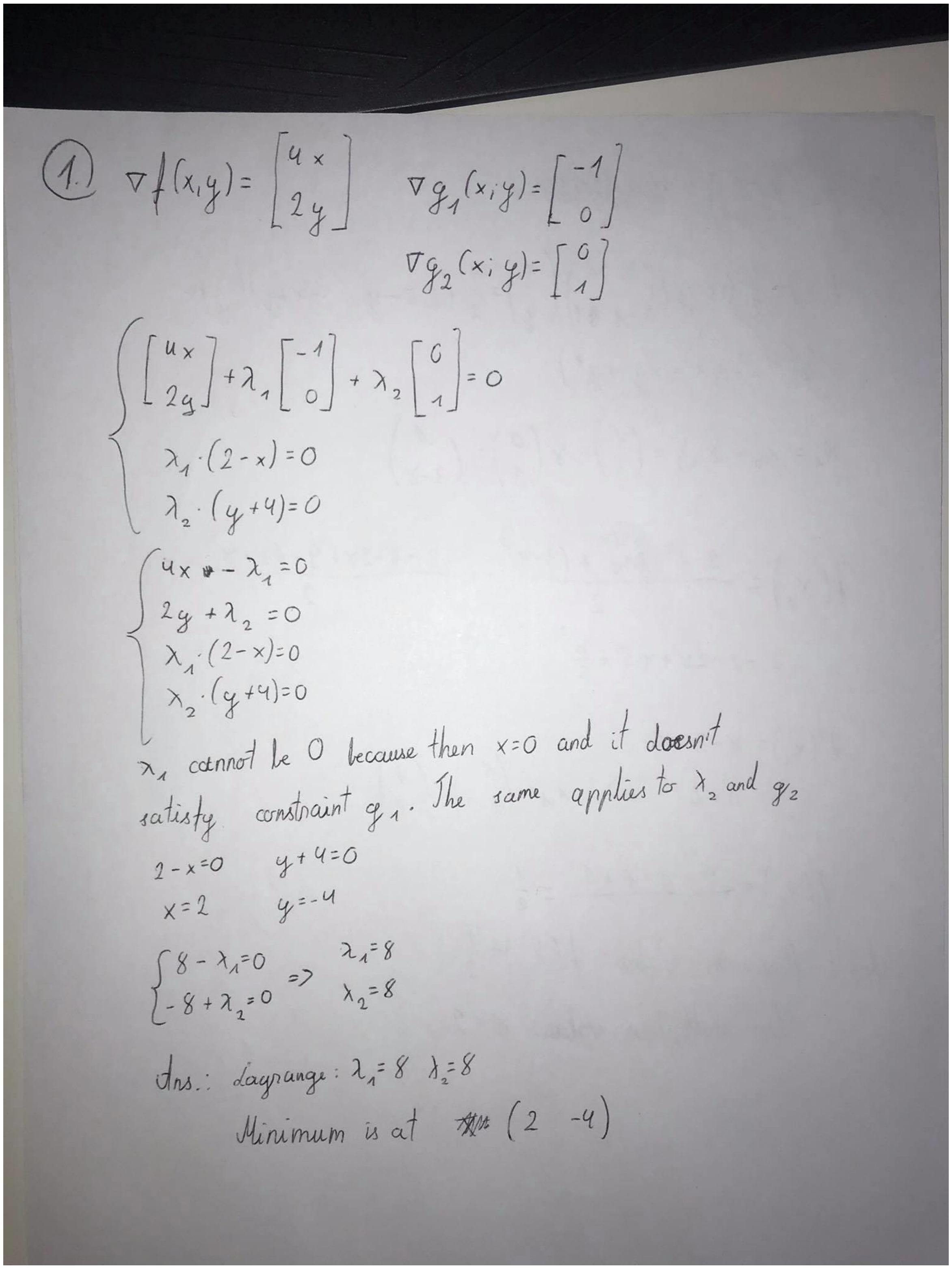
When replacing the equality constraint by the inequality constraint I needed to make the function $h(x, y) = x + y + 2 = 0$ into $g(x, y) = x + y + 2 \leq 0$ because minimumo of the target function is (0, 0). So the result doesn‘t differ a lot from the task 2.

3.2. Matlab code

```
% use meshgrid to create a rectangular grid
[x, y] = meshgrid(-10:1:10, -10:1:10);

% compute function values at the points of the grid
z = x.^2 + y.^2;

% compute gradients
% dx - partial derivative in respect of x; dy - partial
derivative in respect of y
[dx, dy] = gradient(z);

% plot contourlines
contour(x, y, z)

% the next plot will be constructed on top of the existing
figure
hold on

% plot gradients
quiver(x, y, dx, dy)

% the next plot will be constructed on top of the existing
figure
hold on

% plot the constraints function g1
x1 = -10:1:10;
y1 = -2 - x1;
plot(x1, y1, 'r')

% the next plot will be constructed on top of the existing
figure
hold on

% make sure that the grid will stay 10x10
xlim([-10 10])
ylim([-10 10])

% penalty parameter R
r = 1;

% precision
epsilon = 0.001;
```

```
% starting point
xs = [8 8];

% the step size
gamma = 0.0001;

% limit the number of iterations in order to avoid infinite
loops
max_iterations = 10000;

for i=1:max_iterations

    % penalty
    p = max(0, xs(1)+xs(2)+2);

    % gradient of the transformed function
    grad = [2*xs(1) 2*xs(2)] + 2/r*p*[1 1];

    % compute the next point
    x = xs - gamma*grad;

    % plot the step
    plot([xs(1), x(1)], [xs(2), x(2)], 'g', [xs(1), x(1)],
[xs(2), x(2)], 'go')

    % distance between two points
    distance = sqrt((x(1) - xs(1))^2 + (x(2) - xs(2))^2);

    % refresh the variable
    xs = x;

    % update the parameter R
    r = r/1.1;

    % Check if the current minimum point did move from the
previous minimum
    % point less than epsilon
    if distance < epsilon
        break;
    end
end
```

4. Control work



1. $\nabla f(x,y) = \begin{bmatrix} 4x \\ 2y \end{bmatrix}$ $\nabla g_1(x;y) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

$\nabla g_2(x;y) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$$\begin{cases} \begin{bmatrix} 4x \\ 2y \end{bmatrix} + \lambda_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \\ \lambda_1 \cdot (2-x) = 0 \\ \lambda_2 \cdot (y+4) = 0 \end{cases}$$

$$\begin{cases} 4x - \lambda_1 = 0 \\ 2y + \lambda_2 = 0 \\ \lambda_1 \cdot (2-x) = 0 \\ \lambda_2 \cdot (y+4) = 0 \end{cases}$$

λ_1 cannot be 0 because then $x=0$ and it doesn't satisfy constraint g_1. The same applies to λ_2 and g_2

$2-x=0$ $y+4=0$

$x=2$ $y=-4$

$$\begin{cases} 8 - \lambda_1 = 0 \\ -8 + \lambda_2 = 0 \end{cases} \Rightarrow \begin{matrix} \lambda_1 = 8 \\ \lambda_2 = 8 \end{matrix}$$

Ans.: Lagrange: $\lambda_1 = 8$ $\lambda_2 = 8$

Minimum is at $(2\ \ -4)$

$\textcircled{2.}$ 1) $T(x, y, c) = f(x, y) + p(x, y, C) =$

$= 2x^2 + y^2 + c(x+y+5)^2$ when $g_1(x, y) = x + y + 5 = 0$

$T(2, 1, 1) = 2 \cdot 2^2 + 1^2 + 1(2 + 1 + 5)^2 = 73$

2) $f(x, y)$ ~~minimum~~ minimum is $(0, 0)$
because it is a quadratic function
$g(0, 0) = 5$ therefore the new constrait
is $g(x, y) = x + y + 5 \leq 0$ (because $5 \nleq 0$)

$T(x, y, c) = x^2 + y^2 + p(x, y, c)$

$$p(x, y, c) = \begin{cases} 0 \text{ if } x+y+5 \leq 0 \\ c \cdot (x+y+5)^2 \text{ if } x+y+5 > 0 \end{cases}$$

$g(2, 1) = 2 + 1 + 5 = 8 > 0$

$p(2, 1, 1) = 64$

$T(2, 1, 1) = 9 + 64 = 73$