OPTIMIZATION TECHNIQUES
MC418 Lab

Submitted by -

ANEESH PANCHAL

2K20/MC/21

Department of Applied Mathematics,

Delhi Technological University

Delhi Technological University

Vision

To be a world class university through Education, Innovation and Research for the service of humanity.

Mission

- To establish centres of excellence in emerging areas of Science, Engineering, Technology, Management and allied areas.
- To foster an ecosystem for Incubation, Product Development, Transfer of Technology and Entrepreneurship.
- To create environment of Collaboration, Experiment, Imagination and Creativity.
- To develop human potential with Analytical Abilities, Ethics and Integrity.
- To provide environment friendly, reasonable and sustainable solutions for local and global needs.

Department of Applied Mathematics

Vision

To emerge as a centre of excellence and eminence by imparting futuristic technical education with solid mathematical background in keeping with global standards, making our students technologically and mathematically competent and ethically strong so that they can readily contribute to the rapid advancement of society and mankind.

Mission

- To achieve academic excellence through innovative teaching and learning practices.
- To improve the research competence to address social needs.
- To inculcate a culture that supports and reinforces ethically, professional behaviours for harmonious and prosperous society.
- Strive to make students to understand, appreciate and gain mathematical skills and develop logic, so that they are able to contribute intelligently in decision making which characterises our scientific and technological age.

MC418 – Optimization Techniques
Course Outcomes (COs)

CO1

Define and use optimization concepts to model the real-world applications as an optimization problem.

CO2

Apply optimization methods to engineering problems, including developing a model, defining an optimization problem, applying optimization methods, exploring the solution, and interpreting results.

CO3

Formulate continuous problems into unconstrained and constrained optimization problems on the basis of the conditions provided.

CO4

Identify computing derivatives methods for direct and adjoint cases.

CO5

Identify the optimization techniques to determine a robust design for a given real-world problem and justify the technique for solving it.

CO6

Employ basic optimization algorithms in a computational setting and apply existing optimization software packages to solve and analyze the inter-disciplinary real-world problems and for higher study and research.

INDEX

Experiment 1

Aim:

Program to check whether a given function is convex or not.

Theory:

Let X be convex subset of a real vector subspace and let $f: X \to R$ be a function, then f is convex if and only if For every $0 \leq t \leq 1$ and all $x_1, x_2 \in X$

$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$

Problem:

Check the convexity of x^2 in [1,2]

Code:

```
syms x f(x)
x1 = input('Lower Limit: ');
x2 = input('Upper Limit: ');
f(x) = input('Function to check f(x): ');
flag = 0;
for i=0:0.01:1
    if f((1-i)*x1 + i*x2) > (1-i)*f(x1) + i*f(x2)
        fprintf('\nFunction is not Convex Function\n\n')
        flag = -1;
        break;
    end
end
if flag == 0
    fprintf('\nFunction is Convex Function\n\n')
end
```

Output:

```
>> Exp1
Lower Limit: 1
Upper Limit: 2
Function to check f(x): x*x

Function is Convex Function

>>
```

Conclusion:

The given function x^2 is Convex in [1,2] which can be easily be seen from the graph of x^2.
Here in this code, we use the for loop by dividing [1,2] in 100 equal parts.

Experiment 2

Aim:

Program to check convexity of function by Hessian Matrix.

Theory:

Suppose $f: R^n \to R$ is a function taking as input a vector $x \in R^n$ and outputting a scalar $f(x) \in R$. If all 2nd order partial derivatives of f exist, then the Hessian matrix H of f is a square nxn matrix,

$$\mathbf{H}_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex] \vdots & \vdots & \ddots & \vdots \\[2.2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n$ be the eigen values of H_f then,

1. If $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n \geq 0$ then f is convex function
2. If $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n > 0$ then f is strictly convex function
3. If $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n \leq 0$ then f is concave function
4. If $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n < 0$ then f is strictly concave function
5. Else convexity or, concavity of f can't be determine

Problem:

Check the convexity of $4x^2 + y^2 + 4xy$

Code:

```
syms x y
f = input('Function to check f(x,y): ');
Hessian_Matrix = [diff(f,x,2), diff(diff(f,x),y); diff(diff(f,y),x),
diff(f,y,2)];
EigenValues = eig(Hessian_Matrix);
x0 = input('Convexity to check at x: ');
y0 = input('Convexity to check at y: ');
EigenValues = subs(subs(EigenValues,x,x0),y,y0);

if (EigenValues(1)==0 && EigenValues(2)>0) || (EigenValues(2)==0 &&
EigenValues(1)>0)
    fprintf('\nFunction is Convex Function\n\n');
elseif (EigenValues(1)==0 && EigenValues(2)<0) || (EigenValues(2)==0 &&
EigenValues(1)<0)
    fprintf('\nFunction is Concave Function\n\n');
elseif (EigenValues(1)>0 && EigenValues(2)>0) || (EigenValues(2)>0 &&
EigenValues(1)>0)
    fprintf('\nFunction is Strictly Convex Function\n\n');
elseif (EigenValues(1)<0 && EigenValues(2)<0) || (EigenValues(2)<0 &&
EigenValues(1)<0)
    fprintf('\nFunction is Strictly Concave Function\n\n');
else
    fprintf('\nCant Determine\n\n');
end
```

Output:

```
>> Exp2
Function to check f(x,y): 4*x^2 + y^2 + 4*x*y
Convexity to check at x: 0
Convexity to check at y: 0

Function is Convex Function

>>
```

Conclusion:

As the given function is quadratic hence it is either globally convex, globally concave or not determinable.
The Hessian Matrix for the given function,

$H_f = \begin{bmatrix} 8 & 4 \\ 4 & 2 \end{bmatrix}$

for which we get eigen values as,

$\lambda_1 = 0$ and $\lambda_2 = 10$

i.e. the given function is Convex Function.

Experiment 3

Aim:

Program to check whether the given optimization problem is convex or not.

Theory:

A convex optimization problem is a problem where all of the constraints are convex functions, and the objective is a convex function if minimizing, or a concave function if maximizing.
A convex optimization problem is a problem where all of the constraints are concave functions, and the objective is a convex function if minimizing, or a concave function if maximizing.
That is,

1. Min $f_i(x)$ is convex and $g_i(x)$ is convex where $g_i(x) \leq 0$
2. Min $f_i(x)$ is convex and $g_i(x)$ is concave where $g_i(x) \geq 0$
3. Max $f_i(x)$ is concave and $g_i(x)$ is convex where $g_i(x) \leq 0$
4. Max $f_i(x)$ is concave and $g_i(x)$ is concave where $g_i(x) \geq 0$

Problem:

Check whether the given problem is Convex Programming Problem,

$Min\ z = 4x + 3y$

Subject to,

$x + y - 4 \leq 0$

$xy - 1 \leq 0$

$x, y \geq 0$

Code:

```
syms x y z
m = input("Ïnput Number of variables: ");
obj = input("Input Objective Function: ");
n = input("Ïnput Number of constraints: ");
const(1) = input("Ïnput constraint: ");
for i=2:n
    const(i) = input("Ïnput constraint: ");
end
objval = Convex(obj,0,0,0,m);
count = 0;
for i=1:n
    count = count + Convex(const(i),0,0,0,m);
end
if objval==1 && count==-n
    fprintf('\nIt is Convex Programming Problem\n\n');
elseif objval==-1 && count==n
    fprintf('\nIt is Convex Programming Problem\n\n');
elseif objval==1 && count==n
    fprintf('\nIt is Convex Programming Problem\n\n');
elseif objval==-1 && count==-n
    fprintf('\nIt is Convex Programming Problem\n\n');
else
    fprintf('\nIt is NOT Convex Programming Problem\n\n');
end
```

Convex.m

```
function hess = Convex(f,x0,y0,z0,n)
    syms x y z
    if n==2
        var = [x,y];
        val = [x0,y0];
        Hessian_Matrix = [diff(f,x,2), diff(diff(f,x),y); diff(diff(f,y),x), 
diff(f,y,2)];
    elseif n==3
        var = [x,y,z];
        val = [x0,y0,z0];
        Hessian_Matrix = [diff(f,x,2), diff(diff(f,x),y), diff(diff(f,x),z); 
diff(diff(f,y),x), diff(f,y,2), diff(diff(f,y),z); diff(diff(f,z),x), 
diff(diff(f,z),y), diff(f,z,2)];
    else
        var = x;
        val = x0;
        Hessian_Matrix = diff(f,x,2);
    end
    hess = 0;
    EigenValues = eig(Hessian_Matrix);
    EigenValues = subs(EigenValues,var,val);
    for i=1:n
        if EigenValues(i)>=0
            hess = hess + 1;
        elseif EigenValues(i)<=0
            hess = hess - 1;
        end
    end
    if hess==n
        hess = 1;
    elseif hess==-n
        hess = -1;
    else
        hess = 0;
    end
end
```

Output:

```
>> Exp3
Input Number of variables: 2
Input Objective Function: 4*x + 3*y
Input Number of constraints: 2
Input constraint: x+y-4
Input constraint: x*y-1

It is NOT Convex Programming Problem

>>
```

Conclusion:

The given objective function $4x + 3y$ and constraint $x + y - 4$ are Convex as these are equation of straight line. But the constraint $xy - 1$ is neither convex nor concave (using experiment 2 (Hessian Matrix)).

Experiment 4

Aim:

Program to derive the KKT conditions for given optimization problem.

Theory:

Karush–Kuhn–Tucker (KKT) conditions are the conditions below used for solving Nonlinear Programming Problem (mainly Quadratic Programming Problem) using Restricted Simplex Method.

$Max\ f(x)$

subject to,

$g_i(x) \leq 0$

$x \geq 0$

The conditions will be generated by,

$-\nabla f\ = -c\ -\ 2Dx$

$-c - 2Dx + \lambda A^T - \mu I = 0$

$$\begin{bmatrix} -2D & A^T & -I & 0 \\ A & 0 & 0 & I \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \\ s \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix}$$

$\mu_j x_j\ = 0$

$\lambda_i s_i\ = 0$

$x, \lambda, \mu, s \geq 0$

Problem:

Find the KKT conditions for the given problem,

$Max\ z = x + y - x^2 + 2xy - 2y^2$

subject to,

$2x\ +\ y \leq 1$

$x, y \geq 0$

Code:

```
syms x y z
m = input("Ïnput Number of variables: ");
maxeqn = input("Input Max Objective Function: ");
n = input("Ïnput Number of constraints: ");
consLHS(1)  = input("Ïnput constraint LHS: ");
consRHS(1)  = input("Ïnput constraint RHS: ");
for i=2:n
    consLHS(i)  = input("Ïnput constraint RHS: ");
    consRHS(i)  = input("Ïnput constraint RHS: ");
end
if m==2
    var = [x,y];
    z = [0,0];
elseif m==3
    var = [x,y,z];
    z = [0,0,0];
else
    var = x;
    z = 0;
end
```

```
J = jacobian(maxeqn,var);
D = (1/2)*jacobian(J,var);
C = transpose(subs(J,var,z));
A = jacobian(consLHS,var);
b = transpose(consRHS);
fprintf("KKT Conditions are:\n");
KKTLHS = [-2*D, transpose(A), -eye(ndims(D(1))), zeros(ndims(D(1)),size(A,1));
A, zeros(size(A,1)), zeros(size(A,1),ndims(D(1))), eye(size(A,1))]
KKTRHS = [C; b]
```

Output:

```
>> Exp4
Input Number of variables: 2
Input Max Objective Function: x + y - x^2 + 2*x*y - 2*y^2
Input Number of constraints: 1
Input constraint LHS: 2*x + y
Input constraint RHS: 1

KKT Conditions are:

KKTLHS =

[  2, -2, 2, -1,  0, 0]
[ -2,  4, 1,  0, -1, 0]
[  2,  1, 0,  0,  0, 1]

KKTRHS =

 1
 1
 1

>>
```

Conclusion:

Jacobian for the objective function f is,

$\nabla f = J$ = [2y - 2x + 1, 2x - 4y + 1]

And hence the other submatrices for the same is,

D = [-1, 1; 1, -2]

c = [1; 1]

A = [2, 1]

b = 1

and hence the final KKT conditions can be found which are,

$[\,2, -2, 2, -1, 0, 0;\ -2, 4, 1, 0, -1, 0;\ 2, 1, 0, 0, 0, 1][x;\ y;\ \lambda;\ \mu_1;\ \mu_2;\ s] = [1;\ 1;\ 1]$

Experiment 5

Aim:

Program to solve the given optimization problem by Lagrangian Method.

Theory:

Let $(x,\lambda) \in R^n X R^m$ exists such that $\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0$
Let $z(x) = \{z \in R^n : z^T \nabla g(x) = 0\}$
Let $H_L(x,\lambda)$ denotes the Hessian of Lagrange function at (x,λ)
Furthermore let $z^T H_L(x,\lambda) z > 0$ for every $z \in z(x)$
Then x is the local minimum point of f such that $g_i(x) = 0$
In similar manner let $z^T H_L(x,\lambda) z < 0$ for every $z \in z(x)$
Then x is the local maximum point of f such that $g_i(x) = 0$

Problem:

Find the solution of the given problem using Lagrange method,
Maximize $z = (x^3/3) - (3y^2/2) + 2x$
subject to,
$x - y = 0$

Code:

```
syms x y z lambda z1 z2 z3
m = input("Ïnput Number of variables: ");
func = input("Input Max Objective Function: ");
n = input("Ïnput Number of constraints: ");
cons(1) = input("Ïnput constraint: ");
if n>1
    for i=2:n
        cons(i) = input("Ïnput constraint: ");
    end
end
L = func + lambda*lhs(cons);
if m==2
    var = [x,y];
    cz = [z1,z2];
    dx = diff(L,x) == 0;
    dy = diff(L,y) == 0;
    dl = diff(L,lambda) == 0;
    sys = [dx;dy;dl];
    [xval,yval,lval] = solve(sys,[x y lambda],'Real',true);
    res = vpa([xval,yval,lval],5);
    sol = vpa([xval,yval],5)
    eqn = cz*transpose(jacobian(lhs(cons),var))==0;
    zsol(1) = 1;
    zsol(2) = subs(solve(eqn,z2),z1,zsol(1));
elseif m==3
    var = [x,y,z];
    cz = [z1,z2,z3];
    dx = diff(L,x) == 0;
    dy = diff(L,y) == 0;
    dz = diff(L,z) == 0;
    dl = diff(L,lambda) == 0;
    sys = [dx;dy;dz;dl];
    [xval,yval,zval,lval] = solve(sys,[x y z lambda],'Real',true);
    res = vpa([xval,yval,zval,lval],5);
    sol = vpa([xval,yval,zval],5)
    eqn = cz*transpose(jacobian(lhs(cons),var))==0;
    zsol(1) = 1;
    eqn = subs(eqn,z1,zsol(1));
    [zsol(2),zsol(3)] = solve(eqn,[z2,z3],'Real',true);
else
    var = x;
    cz = z1;
    dx = diff(L,x) == 0;
    dl = diff(L,lambda) == 0;
    sys = [dx;dl];
    [xval,lval] = solve(sys,[x lambda],'Real',true);
    res = vpa([xval,lval],5);
    sol = vpa(xval,5)
    eqn = cz*transpose(jacobian(lhs(cons),var))==0;
    zsol(1) = solve(eqn,z1);
end
HL = Hessian(func,m);
Val = zsol*HL*transpose(zsol);
for i=1:size(res,1)
    if subs(Val,var,sol(i,:))>0
        fprintf('\nLocal Minimum Point\n');
        sol(i,:)
        fprintf('\n');
    elseif subs(Val,var,sol(i,:))<0
        fprintf('\nLocal Maximum Point\n');
        sol(i,:)
        fprintf('\n');
    else
        res(i)
        fprintf('\nCant Determine\n\n');
    end
end
```

Hessian.m

```
function Hessian_Matrix = Hessian(f,n)
    syms x y z
    if n==2
        Hessian_Matrix = [diff(f,x,2), diff(diff(f,x),y); diff(diff(f,y),x),
diff(f,y,2)];
    elseif n==3
        Hessian_Matrix = [diff(f,x,2), diff(diff(f,x),y), diff(diff(f,x),z);
diff(diff(f,y),x), diff(f,y,2), diff(diff(f,y),z); diff(diff(f,z),x),
diff(diff(f,z),y), diff(f,z,2)];
    else
        Hessian_Matrix = diff(f,x,2);
    end
end
```

Output:

```
>> Exp5
Input Number of variables: 2
Input Objective Function: ((x^3)/3) - ((3*y^2)/2) + 2*x
Input Number of constraints: 1
Input constraint: x - y == 0
```

```
sol =

[ 1.0, 1.0]
[ 2.0, 2.0]

Local Maximum Point
ans =

[ 1.0, 1.0]


Local Minimum Point
ans =

[ 2.0, 2.0]

>>
```

Conclusion:
The given objective function have local maximum point at $(1,1)$ with Maximum objective value as $z = 0.8333$.
The given objective function have local minimum point at $(2,2)$ with Minimum objective value as $z = 0.6666$.

Experiment 6

Aim:

Program to solve the quadratic programming problem by Wolfe Method.

Theory:

Wolfe Method uses KKT conditions to solve the Quadratic programming problem. Along with restricted entry in the basis matrix such that ($\mu_j x_j = 0$ and $\lambda_i s_i = 0$)
We will be using wolf inbuilt function for solving restricted entry simplex tableau.
Documentation of the wolf function is available at,
https://in.mathworks.com/matlabcentral/fileexchange/27397-quadratic-programming-by-wolf-s-method
Bapi Chatterjee (2023). Quadratic programming by Wolf's method MATLAB Central File Exchange. Retrieved March 19, 2023.

Problem:

Solve the following Quadratic Programming Problem using Wolfe Method,
$Max\ z = 6x + 6y - x^2 - y^2 - 18$
subject to,
$x + y \leq 2$
$x - y \leq 1$
$x, y \geq 0$

Code:

```
syms x y z lambda mu
m = input("Ïnput Number of variables: ");
maxeqn = input("Input Max Objective Function: ");
n = input("Ïnput Number of constraints: ");
consLHS(1) = input("Ïnput constraint LHS: ");
consRHS(1) = input("Ïnput constraint RHS: ");
for i=2:n
    consLHS(i) = input("Ïnput constraint RHS: ");
    consRHS(i) = input("Ïnput constraint RHS: ");
end
if m==2
    var = [x,y];
    z = [0,0];
elseif m==3
    var = [x,y,z];
    z = [0,0,0];
else
    var = x;
    z = 0;
end
J = jacobian(maxeqn,var);
D = (1/2)*jacobian(J,var)
C = transpose(subs(J,var,z))
A = jacobian(consLHS,var)
b = transpose(consRHS)
fprintf("KKT Conditions are:\n");
KKTLHS = [-2*D, transpose(A), -eye(ndims(D(1))), zeros(ndims(D(1)),size(A,1));
A, zeros(size(A,1)), zeros(size(A,1),ndims(D(1))), eye(size(A,1))]
KKTRHS = [C; b]
```

```
%Solving the problem thus generated by using Restricted Entry Simplex Method
D = Hessian(maxeqn,m);
for i = 1:m
    lx(i,1) = subs(diff(maxeqn,var(i)),var(i),0);
    inq(1,i) = -1;
end
minimize = 0;
x = wolf(D,lx,b,A,inq,minimize)
```

Output:

```
>> Exp6
Ïnput Number of variables: 2
Input Max Objective Function: 6*x + 6*y - x^2 - y^2 - 18
Ïnput Number of constraints: 2
Ïnput constraint LHS: x+y
Ïnput constraint RHS: 2
Ïnput constraint LHS: x-y
Ïnput constraint RHS: 1
KKT Conditions are:

KKTLHS =

[ 2,  0, 1,  1, -1,  0, 0, 0]
[ 0,  2, 1, -1,  0, -1, 0, 0]
[ 1,  1, 0,  0,  0,  0, 1, 0]
[ 1, -1, 0,  0,  0,  0, 0, 1]


KKTRHS =

 6
 6
 2
 1


The optimum has achieved!

x =

    1
    1

>>
```

Conclusion:

The optimal value of the given objective function subject to given constraint are $(x, y) = (1,1)$.
Maximum objective function value is $z = -8$

Experiment 7

Aim:

Program to solve linear fractional programming problem using Charnes and Cooper Method.

Theory:

Charnes and Cooper Method:

$Max\ z = \frac{c^T x + \alpha}{d^T x + \beta}$

subject to,

$Ax = b$

$x \geq 0$

Then let us assume $d^T x + \beta = w$ and $y = wx$

Then we get the problem changed as,

$Max\ z = c^T y + \alpha w$

subject to,

$Ay - wb = 0$

$d^T y + \beta w = 1$

$y, w \geq 0$

We will be using linprog inbuilt function for solving simplex tableau.

Documentation of the linprog function is available at,

https://in.mathworks.com/help/optim/ug/linprog.html

Problem:

Solve the following Linear Fractional Programming Problem using Charnes and Cooper Method,

$Max\ z = \frac{y-5}{-x-y+9}$

subject to,

$-2x - 5y \leq -10$

$4x + 3y \leq 20$

$-x + y \leq 2$

$-x, -y \leq 0$

Code:

```
%Charnes and Cooper Method
clear
syms x y w a b

m = input("Input Number of variables: ");
func = input("Input Max Objective Function: ");
n = input("Input Number of constraints: ");
cons(1) = input("Input constraint: ");
for i=2:n
    cons(i) = input("Input constraint: ");
end

%Declaration of Variable arrays
oldvar = [x y];
newvar = [a/w b/w];
vari = [a b w];
```

```
%Checking the condition for Num and Deno Values
[num,deno] = numden(func)
if subs(num,oldvar,[0 0])>0
    num = -num;
    deno = -deno;
end

%Forward substitution for the given problem
nfunc = subs(func,deno,1/w)
ncons = subs(cons,oldvar,newvar);
nfunc = simplify(subs(nfunc,oldvar,newvar))
n = n+1;
ncons(n) = subs(deno*w,oldvar,newvar)==1

%Getting the LHS and RHS coefficient values from the constraints
consLHS(1) = lhs(ncons(1));
consRHS(1) = rhs(ncons(1));
for i=2:n
    consLHS(i) = lhs(ncons(i));
    consRHS(i) = rhs(ncons(i));
end

%Solving the construted problem using simplex method,
A = zeros(n-1,m+1);
b = zeros(1,n-1);
Aeq = zeros(1,m+1);
Beq = 1;
for i=1:n-1
    A(i,1) = subs(diff(consLHS(i),vari(1)),vari(m+1),1);
    A(i,2) = subs(diff(consLHS(i),vari(2)),vari(m+1),1);
    A(i,3) = -consRHS(i);
end
consLHS(n) = simplify(consLHS(n));
for i=1:m+1
    f(1,i) = diff(nfunc,vari(i));
    Aeq(1,i) = diff(consLHS(n),vari(i));
end

%LinProg Inbuilt function
f = -double(f)
A = double(A)
b = double(b)
Aeq = double(Aeq)
Beq = double(Beq)
xsol = linprog(f,A,b,Aeq,Beq)

%Reverse substitution for getting the final value
final_sol(1) = xsol(1)/xsol(m+1);
for i=2:m
    final_sol(i) = xsol(i)/xsol(m+1);
end
final_sol
objective_func_val = vpa(subs(func,oldvar,final_sol),2)
```

Output:

```
>> Exp7
Input Number of variables: 2
Input Max Objective Function: (y - 5)/(-x - y + 9)
Input Number of constraints: 5
Input constraint: -2*x - 5*y <= -10
Input constraint: 4*x + 3*y <= 20
Input constraint: -x + y <= 2
Input constraint: -x <= 0
Input constraint: -y <= 0

num =

5 - y


deno =

x + y - 9


nfunc =

w*(y - 5)


nfunc =

b - 5*w


ncons =

[ - (2*a)/w - (5*b)/w <= -10, (4*a)/w + (3*b)/w <= 20, b/w - a/w <= 2, -a/w <= 0, -b/w <= 0, -w*(a/w + b/w - 9) == 1]


f =

     0    -1     5


A =

    -2    -5    10
     4     3   -20
    -1     1    -2
    -1     0     0
     0    -1     0


b =

     0     0     0     0     0
```

```
Aeq =

    -1    -1     9


Beq =

     1


Optimal solution found.


xsol =

    0.6667
    1.3333
    0.3333


final_sol =

    2.0000    4.0000


objective_func_val =

-0.33

>>
```

Conclusion:
The optimal value of the given objective function subject to given constraint are $(a, b, w) = (0.6667, 1.3333, 0.3333)$
Hence the values of optimal variables are $(x, y) = (2,4)$
Maximum objective function value is $z = -0.33$

Experiment 8

Aim:

Program to write down mixed dual, Lagrange dual and Wolf dual.

Theory:

Let the optimization problem be given by,

$Max\ z$

s.t. $g_i(x) \leq 0$

Lagrange Dual:

The Lagrange Dual is calculated as the optimal solution of the maximum optimization problem of the Lagrangian equation which is of the form,

$L(\lambda, x) = z + \lambda g_i(x)$

Mixed dual:

The Mixed Dual is calculated as the optimal solution of the maximum optimization problem of the equation which is of the form,

$M(\theta, \mu) = z + \theta g_i(x) + \mu g_i^2(x)$

Wolf Dual:

The Wolf Dual is calculated as the optimal solution of the maximum optimization problem of the equation which is of the form,

$W(\gamma, \delta) = z + \gamma g_i(x) + \frac{\delta}{2} {g_i}^2(x)$

Problem:

Find the mixed, Lagrange and Wolf dual for the following problem,

$Max\ z = -x^2 - y^2$

s.t. $x + y - 1 \leq 0$

$x^2 + y^2 - 1 \leq 0$

Code:

```
%Lagrange, Mixed, Wolfe Dual
syms x y lambda1 lambda2 theta mu gamma delta

f = input("Input Objective Function: ");
cons(1) = input("Input constraint 1: ");
cons(2) = input("Input constraint 2: ");
g = lhs(cons);

% Lagrange Dual
lambda = [lambda1; lambda2];
L = f + lambda(1)*g(1) + lambda(2)*g(2);
LD = simplify(-L);
disp(['Lagrange Dual: ', char(LD)]);

% Mixed Dual
M = f + theta*g' + mu*(g.^2)';
MD = simplify(-M);
disp(['Mixed Dual: ', char(MD)]);
```

```
% Wolfe Dual
W = f + gamma'*g' + 0.5*(g.^2)*delta;
WD = simplify(-W);
disp(['Wolfe Dual: ', char(WD)]);
```

Output:

```
>> Exp8
Input Objective Function: -x^2 - y^2
Input constraint 1: x + y - 1 <= 0
Input constraint 2: x^2 + y^2 - 1 <= 0

Lagrange Dual: x^2 - lambda1*(x + y - 1) + y^2 - lambda2*(x^2 + y^2 - 1)

Mixed Dual: [x^2 - mu*(conj(x) + conj(y) - 1)^2 + y^2 - theta*(conj(x) + conj(y) - 1); x^2 - theta*(conj(x)^2 + conj(y)^2
- 1) - mu*(conj(x)^2 + conj(y)^2 - 1)^2 + y^2]

Wolfe Dual: [x^2 - (delta*(x + y - 1)^2)/2 - conj(gamma)*(conj(x) + conj(y) - 1) + y^2, x^2 - conj(gamma)*(conj(x) +
conj(y) - 1) - (delta*(x^2 + y^2 - 1)^2)/2 + y^2; x^2 - (delta*(x + y - 1)^2)/2 - conj(gamma)*(conj(x)^2 + conj(y)^2 - 1) +
y^2, x^2 - (delta*(x^2 + y^2 - 1)^2)/2 - conj(gamma)*(conj(x)^2 + conj(y)^2 - 1) + y^2]

>>
```

Conclusion:

Lagrange Dual, Mixed Dual and Wolfe Method for the given problem is given by the output of the given code.
Here we have used Min and Max function used for the optimization of the problem.

Experiment 9

Aim:

Program to solve the Non Linear Programming Problem by Penalty function method.

Theory:

Penalty Function Method:

$Min\ f(x)$

subject to, $c_i(x) \leq 0$, for every $i \in I$

This problem can be solved as a series of unconstrained minimization problems,

$Min\ \phi_k(x) = f(x) + \sigma_k \sum_{i \in I} g(c_i(x))$

where,

$g(c_i(x)) = max(0, c_i(x))^2$

Problem:

Solve the following Non Linear Programming Problem using Penalty Function Method,

$Min\ x^2$

subject to, $x \geq 1$

Code:

```
%Penalty Function Method
clear
syms x alpha;

f = input("Input Objective Function: ");
cons = input("Input Constraints: ");
cond = input("Less than (1) or, Greater than (-1): ");
r_val = lhs(cons);

if(cond == 1)
    %Case for less than constraint
    p = 0;
    u = f + alpha*p;
    sol = solve(diff(u,x)==0);
    sol = limit(sol,alpha,inf);

    %Condition for less than
    if(sol<r_val)
        x_val = sol
        obj_func_val = subs(f,sol)
    else
        return;
    end

else
    %Case for greater than constraint
    p = subs(f,x,r_val-x);
    u = f + alpha*p;
    sol = solve(diff(u,x) == 0);
    sol = limit(sol,alpha,inf);
    %Condition for greater than
    if(sol<r_val)
        return;
    else
        x_val = sol
        obj_func_val = subs(f,sol)
    end
end
```

Output:

```
>> Exp9
Input Objective Function: x^2
Input Constraints: x>=1
Less than (1) or, Greater than (-1): -1

x_val =

1



obj_func_val =

1

>>
```

Conclusion:

Using Penalty Function Method for given Non Linear Programming Problem,
Value of optimal variables are $x = 1$
Maximum objective function value is $f(x) = 1$

Experiment 10

Aim:
Program to solve the Non Linear Programming Problem by Barrier function method.

Theory:
Barrier Function Method:

Consider the given form of optimization problem,

$Min\ f(x)$

s.t. $c(x) = 0$

$x \geq 0$

The Barrier Function is,

$Min\ f(x) - \mu \sum_{i=1}^{n} ln(x_i) \rightarrow \nabla f(x) + \nabla c(x)\lambda\ -\ \mu \sum_{i=1}^{n} \frac{1}{x_i}$

s.t. $c(x) = 0$

we can now define $z_i = \frac{\mu}{x_i}$ and solve the modified version of the KKT consitions:

$\nabla f(x) + \nabla c(x)\lambda - z = 0$

$c(x) = 0$

$XZe -\ \mu e = 0$

Problem:
Solve the following Non Linear Programming Problem using Barrier Function Method,

$Min\ x^2$

subject to, $-1 \leq x \leq 1$

Code:
```
%Barrier Function Method
clear
syms x alpha;

f = input("Input Objective Function: ");
cons(1) = input("Input constraint 1: ");
cons(2) = input("Input constraint 2: ");

r_val(1) = lhs(cons(1));
r_val(2) = rhs(cons(2));
g(1) = 1/(r_val(1)-x);
g(2) = 1/(x-r_val(2));

b = -g(1)-g(2);
C = f + alpha*b;

sol = limit(solve(diff(C,x) == 0),alpha,0);
for i=1:length(r_val)
    sol = sol(sol~=r_val(i));
end

x_val = sol
fval = subs(f,sol)
```

Output:

```
>> Exp10
Input Objective Function: x^2
Input constraint 1: -1 <= x
Input constraint 2: x <= 1

x_val =

0


fval =

0

>>
```

Conclusion:

Using Barrier Function Method for given Non Linear Programming Problem,
Value of optimal variables are $x = 0$
Maximum objective function value is $f(x) = 0$