Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Step 0: Initialize data

Make data for matrix A and vector b

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
H2mat = [1 -1; 1 1];
H3mat = [H2mat -1*H2mat; H2mat H2mat];
H4mat = [H3mat -1*H3mat; H3mat H3mat];
H5mat = [H4mat -1*H4mat; H4mat H4mat];
H6mat = [H5mat -1*H5mat; H5mat H5mat];
H7mat = [H6mat -1*H6mat; H6mat H6mat];
H8mat = [H7mat -1*H7mat; H7mat H7mat];

rows = [3, 4, 5, 6 ,7,  8, 11, 12, 13, 19, 20 , 26,  30,  33, 34,  36,
    37,  40,    41, 43];
rows = [rows,47, 49,53, 54,  55,    56,    57,  58,  60,  61,  62,
 64,  65,  69, 72, 75, 80, 81, 82, 84];
rows = [rows,87, 88, 91, 92, 93, 94, 96, 97, 100, 106, 107,109];
A0mat = H8mat(rows,:);        %%% A0mat has 52 rows and 128 columns
tic
x_true = zeros(128,1);
locations = [109,   107,    55,    30,    43];
x_true(locations) = [3.2 -4.3 3.2 -4.3 3.2];
bvect = A0mat*x_true;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```

initial estimate of the vector x

Note: we use an estimator of xvect by computing right inverse of A0mat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
x_initial = A0mat'*inv(A0mat*A0mat')*bvect;
```

verify that Amat*x_initial is equal to bvect

```
if (norm(A0mat*x_initial - bvect) > 0.01)
 disp('WARNING: Amat times initial vect is not equal to bvect');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Step 1

enlarge the matrix to have twice as many columns

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Amat = [A0mat -1*A0mat];
A_size = size(Amat);
mcol = A_size(1);
nrow = A_size(2);
cvect = ones(nrow,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

any vector x can be written as the diference of two vectors,

x_initial =x_plus subtract x_minus

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_plus = zeros(nrow/2,1);
x_minus = zeros(nrow/2,1);
for j = 1:(nrow/2)
 if x_initial(j) > 0
  x_plus(j) = x_initial(j);
 end
 if x_initial(j) < 0
  x_minus(j) = abs( x_initial(j) );
 end
end
```

```
x0_vect = [x_plus ; x_minus];
```

verify that Amat*x0_vect is equal to bvect

```
if (norm(Amat*x0_vect - bvect) > 0.01)
    disp('WARNING: Amat times x0_vect is not equal to bvect');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Main Steps

Given cvect, Amat, and bvect,

use the initial estimate x0_vect to solve for x_vect

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%   Students implement the Interior Point Method in the following.

LINEAR_PROBLEM = 'Minimize';
TOLERANCE = 0.005;
MAX_ITERATIONS = 100;
INFINITY = 1.e15 * max([normest(Amat), normest(bvect),
 normest( cvect)]);

trials = 0;
xvect = x0_vect;
evect = ones(nrow, 1);
yvect = (Amat * Amat') \ (Amat *  cvect);
svect =  cvect - Amat' * yvect;

dx = max(-1.5 * min(xvect), 0);
ds = max(-1.5 * min(svect), 0);

dx_ds = 0.5 * (xvect + dx * evect)' * (svect + ds * evect);
dx_dc = dx + dx_ds / (sum(svect) + nrow * ds);
ds_dc = ds + dx_ds /(sum(xvect) + nrow * dx);
xvect = xvect + dx_dc * evect;
svect = svect + ds_dc * evect;
% Loop until LP reach optimum or solution does not
% exist or MaxIterations reached
for i = 1:MAX_ITERATIONS
    trials = trials + 1;
% calculate the residuals and the duality measure
    dual_residual = Amat' * yvect + svect -  cvect;
    primal_residual = Amat * xvect - bvect;
    complimentary_residual = xvect .* svect;
    % duality measure
```

```
   mu = xvect' * svect / nrow;
   residual = normest( complimentary_residual, 1) / (1 + abs(bvect' *
yvect));
   % Exit if  LP no solution
   if isnan(residual) || normest(xvect) + normest(svect) >= INFINITY
       disp('The LP problem has no solution!');
       break;
   end
   % the termination criterion
   if max(mu, max(normest(primal_residual), normest(dual_residual)))
<= TOLERANCE
       disp(' Termination The LP problem is  minimization')
       break;
   end
   % Create the coefficient matrix of the linear systems
   Coefficient_Matrix = Amat * diag(xvect ./ svect) * Amat';
   % Using Cholesky factorization to find upper triangulat Matrix
   % UpperTriangular of the
   % coefficient matrix and find out if coefficient matrix is
   % positive definite = { p > 0}
   % if Matrix is positive definite we assume that Linear Problem
   % has optimal solution
   [UpperTriangular, p] = chol(Coefficient_Matrix);
   if p > 0
       disp('If positive definite The LP problem is minimization');
       break;
   end
   % predictor step
   % Solve Linear System to find dx dy ds
   rhs = primal_residual - Amat * (( complimentary_residual -
xvect .* dual_residual) ./ svect);
   dy = UpperTriangular \ (UpperTriangular' \ rhs);
   ds = dual_residual - Amat' * dy;
   dx = ( complimentary_residual - xvect .* ds) ./ svect;
   alpha_primal = 1 / max([1; dx ./ xvect]);
   alpha_dual = 1 / max([1; ds ./ svect]);
   % centering parameter step
   mun = ((xvect - alpha_primal * dx)' * (svect - alpha_dual * ds)) /
nrow;
   rho = (mun / mu) ^ 3;
   % corrector step
   complimentary_residual =  complimentary_residual - rho * mu +
dx .* ds;
   rhs = primal_residual - Amat * (( complimentary_residual -
xvect .* dual_residual) ./ svect);
   dy = UpperTriangular \ (UpperTriangular' \ rhs);
   ds = dual_residual - Amat' * dy;
   dx = ( complimentary_residual - xvect .* ds) ./ svect;

   alpha_primal = (1 - mu) / max([(1 - mu); dx ./ xvect]);
   alpha_dual = (1 - mu) / max([(1 - mu); ds ./ svect]);
   % update step
   xvect = xvect - alpha_primal * dx;
   yvect = yvect - alpha_dual * dy;
```

```
    svect = svect - alpha_dual * ds;
end
%%%%%%%%%%%%      End of Main Steps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rememer that the second half of the vector xvect are

the negative entries of the optimal vector that we seek

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

trim anything in the solution that is less than threshold number

```
threshold = 0.01;
for j = 1:nrow
 if abs(xvect(j)) < threshold
  xvect(j) = 0;
 end
end

optxvect = xvect(1:floor(nrow/2)) -xvect(floor(nrow/2)+1:nrow);
```

compare the true solution and the optimal solution

```
[x_true optxvect]

trials

toc
```

```
 Termination The LP problem is  minimization

ans =

         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
```

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
-4.3000	-4.3000
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
3.2000	3.2000
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
3.2000	3.2000
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

```
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
   -4.3000   -4.3000
         0         0
    3.2000    3.2000
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
```

```
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0
         0          0


trials =

     5

Elapsed time is 0.047792 seconds.
```

Published with MATLAB® R2015b