Case Study

The objective of this case study is to investigate the performance of the first-order Lagrangian algorithm that minimizes a quadratic function subject to linear inequality constraints.

The optimization problem considered has the form:

$$\begin{aligned} &\text{minimize } x_1^2 + 4x_2^2 \\ &\text{subject to } x_1 + x_2 \geq 4. \end{aligned}$$

We first solve this problem analytically. To begin with solving the problem, we represent the constraint as

$$g = g(x_1, x_2) = 4 - x_1 - x_2 \leq 0.$$

We then form the Lagrangian function,

$$l(x, \mu) = x_1^2 + 4x_2^2 + \mu(4 - x_1 - x_2).$$

The KKT conditions take the form,

$$\begin{aligned} &D_x l(x, \mu) = [2x_1 - \mu \quad 8x_2 - \mu] = 0^\top \\ &\mu(4 - x_1 - x_2) = 0 \\ &\mu \geq 0 \\ &4 - x_1 - x_2 \leq 0. \end{aligned}$$

We have the following set of relations:

$$\begin{aligned} &2x_1 - \mu = 0 \\ &8x_2 - \mu = 0 \\ &g(x_1, x_2) = x_1 + x_2 - 4 \geq 0 \\ &\mu \geq 0. \end{aligned}$$

From the first two equations above, we obtain

$$x_1 = 4x_2.$$

We consider two cases:

1. $g = 0$
2. $g < 0.$

In the first case, when $g = 0$, we have, $g = x_1 + x_2 - 4 = 0$ and $x_1 = 4x_2$. Hence

$$x^* = \begin{bmatrix} 3.2 \\ 0.8 \end{bmatrix}$$

and

$$\mu^* = 2x_1^* = 6.4 > 0.$$

Using the second-order sufficient condition, we conclude that x^* is a strict minimizer.

In the second case, when $g < 0$, we have to have $\mu = 0$. This gives

$$x^* = 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

However, $x^* = 0$ does not satisfy the constraint $g \leq 0$. Thus we have only one point that satisfies the KKT conditions.

We verify our computations using MATLAB's `fmincon` function. The script has the form:

```
function[]=module41()
% Minimization subject to linear inequality constraint

fun = @(x)x(1)^2 + 4*x(2)^2;
x0 = [-1,2];
A = [-1,-1];
b = -4;
x = fmincon(fun,x0,A,b,[],[])
function_value = x(1)^2 + 4*x(2)^2
```

The solution obtained using the above MATLAB script is the same as the one above arrived at analytically.

Finally, we solve the above problem using the first-order Lagrangian algorithm implemented in MATAB with the following script.

```
function[]=Lagrangian_algo_inequality()
%First-order Lagrangian algo for inequality constraints.

clear all
close all
fs=12;
xx1 = linspace(-4, 4, 201);
xx2 = linspace(-4, 4, 201);
[X1, X2] = meshgrid(xx1, xx2);
 fR = X1.^2 + 4*X2.^2;
 gR=X1+X2-4;

x = [3.2; 0.8];
button=1;
fun = x(1)^2 + 4*x(2)^2;
g=x(1)+x(2)-4;
contour(X1, X2, fR, [fun fun],'color','b','ShowText','on');
 hold on
contour(X1, X2, gR, [g g],'color','r');
axis(0.4*[-10 10 -10 10])
axis square
grid

set(gca,'Fontsize',fs,'FontName','Times')
xlabel('$x_1$','Interpreter','latex')
ylabel('$x_2$','Interpreter','latex')

x=[3; 2];
while (button==1)
alpha = 0.005;
mu=0.05;

   dfx1 = 2*x(1);
   dfx2 = 8*x(2);
   gf = [dfx1; dfx2];

   g_con=[-1; -1];
   g=-x(1)-x(2)+4;

while (norm(gf + mu*g_con)>0.01)|(abs(mu*(x(1)+x(2)-4))>0.01)|(g<0)

   dfx1 = 2*x(1);
   dfx2 = 8*x(2);
   gf = [dfx1; dfx2];
   g_con=[-1;- 1];
   px = x;
   % First-order Lagarangian algorithm
   x = x - alpha*(gf + mu*g_con);
   mu = max(mu + alpha*g,0);

   fun = x(1)^2 + 4*x(2)^2;
   g=4-x(1)-x(2);
   X = [px(1), x(1)]; Y = [px(2), x(2)];
   plot(X, Y);
   arrow(X,Y);
fprintf('x_1(%2.0f) = %7.5f,  x_2(%2.0f) = %7.5f, mu(%2.0f)= %7.5f, fun = %7.5f,g = %7.5f,gra
d l norm =%7.5f\n',...
i,x(1),i,x(2),i,mu,fun,g, norm(gf + mu*g_con))

end
[x1,x2,button]=ginput(1);
```

```
x=[x1 x2]';
 end
%-----------------------------------
function h = arrow(x, y, s, style)
%ARROW Use arrows to plot curves.
%      LINE_HANDLE = ARROW(X, Y, S, STYLE)
%      S (0.2 by default) is the scale of the arrow head;
%      STYLE ('-' by default) is the line style of the arrow;
%      LINE_HANDLE is the handle of the arrow.

% J.-S. Roger Jang, 1993

if nargin <= 2, s = 0.2; end
if nargin <= 3, style = '-'; end

xx = [0 1 1-s 1 1-s].';
yy = [0 0 s/2 0 -s/2].';
arrow = xx + yy.*sqrt(-1);

x=x(:);
y=y(:);
z = x + y*sqrt(-1);
a = arrow*diff(z).'+ones(5,1)*z(1:length(z)-1).';
h = plot(real(a), imag(a), style);
```

In the the figure below, we show trajectories generated by the algorithm for different initial points using the above script. Note all trajectories converge to the solution point obtained previously.