**Technical University of Crete**
**School of Electrical and Computer Engineering**
Course: **Convex Optimization**
Exercise 4 (100/1000)
Report Delivery Date: 16 December 2021
Instructor: Athanasios P. Liavas

**Student:** Alevrakis Dimitrios 2017030001

---

Consider the problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(x) = -\sum_{i=1}^{n} log(x_i) \qquad\qquad \text{subject to } \mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}\mathbb{R}^p,\ rank(\mathbf{A}) = p,\ \mathbf{b} \in \mathbb{R}^p$.

1. Data Generation. The function rand of MATLAB return a matrix with i.i.d elements with $\mathbb{U}[0,1]$. So for given $n,\ p$ we generate, $\mathbf{A}\ \mathbb{U}[0,1]$ and a $\mathbf{x}\ \mathbb{U}[0,1]$ in order to derive $\mathbf{b} = \mathbf{Ax}$.

2. We use cvx to solve the problem. The MATLAB code will be added in the end for possible review.

3. We solved the problem using the Newton algorithm starting from a feasible point.

   i. First we find a feasible point using cvx and setting the problem equal to 0 and constraints $\mathbf{Ax} = \mathbf{b}$ and $x > 0$.

   ii. We implement the newton algorithm with backtracking line search. In each iteration we need to check if the step keeps $\mathbf{x}$ in the domain of f. Also since the problem has affine equality constraints, the KKT for the second order approximation of f can be written as (by taking into account that $\mathbf{Ax} = b$ since x is feasible):

$$\nabla^2 f(x)\mathbf{\Delta x} + \mathbf{A}^T\mathbf{w} + \nabla f(x) = 0 \tag{1}$$

$$\mathbf{A\Delta x} = 0 \tag{2}$$

We solve (1) for $\mathbf{\Delta x}$:

$$\nabla^2 f(x)\mathbf{\Delta x} + \mathbf{A}^T\mathbf{w} + \nabla f(x) = 0 \Rightarrow \mathbf{\Delta x} = -(\nabla^2 f(x))^{-1}(\mathbf{A}^T\mathbf{w} + \nabla f(x)) \quad (3)$$

Substitute (3) to (2):

$$\begin{aligned}
&\mathbf{A\Delta x} = 0 \Rightarrow -\mathbf{A}(\nabla^2 f(x))^{-1}(\mathbf{A}^T\mathbf{w} + \nabla f(x)) = 0 \Rightarrow \\
&-\mathbf{A}(\nabla^2 f(x))^{-1}\mathbf{A}^T\mathbf{w} - \mathbf{A}(\nabla^2 f(x))^{-1}\nabla f(x) = 0 \Rightarrow \\
&\mathbf{w} = -(\mathbf{A}(\nabla^2 f(x))^{-1}\mathbf{A}^T)^{-1}\mathbf{A}(\nabla^2 f(x))^{-1}\nabla f(x) \qquad (4)
\end{aligned}$$

Also the newton descend which we use for the terminating condition:
$\lambda^2 = \mathbf{\Delta x}^T\nabla^2 f(x)\mathbf{\Delta x}$

iii. We plot the norm of $x_k$ in each iteration against the solution produced by cvx. We observe how the $x_k$ gets closer to the solution in each iteration.

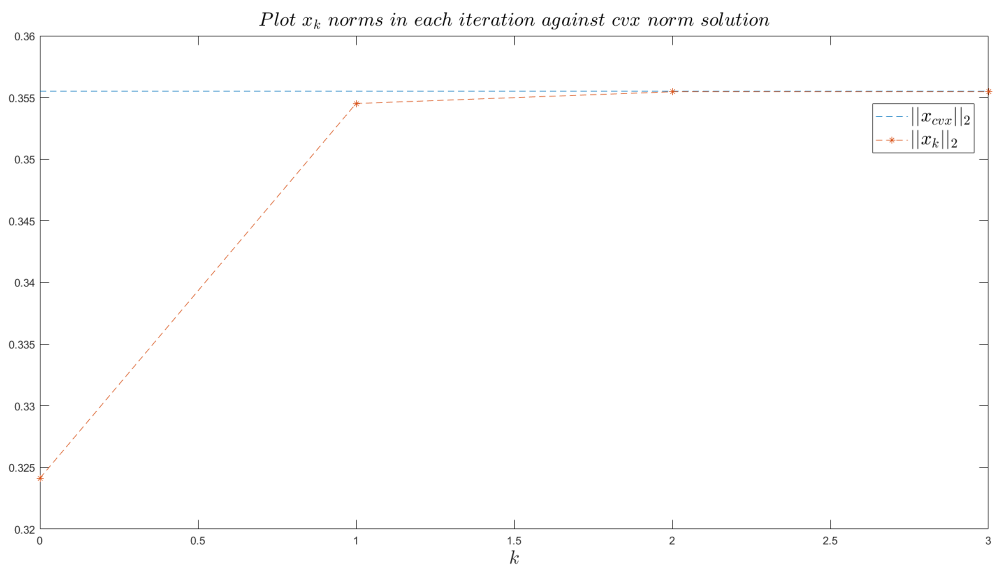



Figure 1: Algorithm progression for $n = 2, p = 1$

4 Again the KKT for the second order approximation of f can be written as:

$$\nabla^2 f(x)\mathbf{\Delta x} + \mathbf{A}^T\mathbf{v} = -\nabla f(x) \qquad (5)$$

$$\mathbf{Ax} - \mathbf{b} = 0 \qquad (6)$$

Let the residual function $r : \mathbb{R}^n x \mathbb{R}^p \rightarrow \mathbb{R}^n x \mathbb{R}^p$:

$$r(\mathbf{x}, \mathbf{v}) = (r_{dual}(\mathbf{x}, \mathbf{v}), r_{primal}(\mathbf{x}, \mathbf{v}))$$

Where the dual residual: $r_{dual}(\mathbf{x}, \mathbf{v}) = \nabla f(x) + \mathbf{A}^T\mathbf{v}$,
and the primal residual: $r_{primal}(\mathbf{x}, \mathbf{v}) = \mathbf{Ax} - \mathbf{b}$

By calculating when the first order approximation of $r$ at $(\mathbf{x}, \mathbf{v})$ is equal to 0 we get:

$$\begin{bmatrix} \nabla^2 f(x) & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} dx_{pd} \\ dv_{pd} \end{bmatrix} = - \begin{bmatrix} r_{dual}(\mathbf{x}, \mathbf{v}) \\ r_{primal}(\mathbf{x}, \mathbf{v}) \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \mathbf{A}^T\mathbf{v} \\ \mathbf{Ax} - \mathbf{b} \end{bmatrix} \tag{7}$$

From (7) we get:

$$\nabla^2 f(x)\mathbf{dx_{pd}} + \mathbf{A}^T\mathbf{dv_{pd}} = -(\nabla f(x) + \mathbf{A}^T\mathbf{v}) \tag{8}$$

$$\mathbf{Adx_{pd}} = -(\mathbf{Ax} - b) \tag{9}$$

Using (8):

$$\begin{aligned}
&(8) \Rightarrow \mathbf{dx_{pd}} = -(\nabla^2 f(x))^{-1}(\nabla f(x) + \mathbf{A}^T\mathbf{v} + \mathbf{A}^T\mathbf{dv_{pd}}) \Rightarrow \\
&\mathbf{Adx_{pd}} = -\mathbf{A}(\nabla^2 f(x))^{-1}(\nabla f(x) + \mathbf{A}^T\mathbf{v} + \mathbf{A}^T\mathbf{dv_{pd}}) \overset{(9)}{\Rightarrow} \\
&-(\mathbf{Ax} - b) = -\mathbf{A}(\nabla^2 f(x))^{-1}(\nabla f(x) + \mathbf{A}^T\mathbf{v} + \mathbf{A}^T\mathbf{dv_{pd}}) \\
&\mathbf{dv_{pd}} = -(\mathbf{A}(\nabla^2 f(x))^{-1}\mathbf{A}^T)^{-1}\Big[\mathbf{A}(\nabla^2 f(x))^{-1}(\nabla f(x) + \mathbf{A}^T\mathbf{v}) - (\mathbf{Ax} - \mathbf{b})\Big]
\end{aligned}$$

Therefore we have calculated the primal-dual steps and can apply the newton algorithm with backtracking line search on $r$.
Since $(\mathbf{dv_{pd}}, \mathbf{dx_{pd}})$ is a descend direction for $||r||_2^2$ we can use $||r(\mathbf{x}, \mathbf{v})|| < \epsilon$ as stop criterion for the newton algorithm.

5. Let the Langragian $L : domf x \mathbb{R}^p \rightarrow \mathbb{R}$ where:

$$L(\mathbf{x}, \mathbf{v}) = -\sum_{i=1}^{n} log(x_i) + \mathbf{v}^T(\mathbf{Ax} - \mathbf{b})$$

Therefore the dual problem is:

$$\underset{\mathbf{v} \in \mathbb{R}^p}{max} \underset{\mathbf{x} \in \mathbf{domf}}{min} L(\mathbf{x}, \mathbf{v})$$

and the dual function is defined as:

$$g(\mathbf{v}) = \underset{\mathbf{x} \in \mathbf{domf}}{inf} \; L(\mathbf{x}, \mathbf{v})$$

.

We will prove that L for a given $\mathbf{v}$ is convex:
The gradient of f:

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \mathbf{v})_i = -\frac{1}{x_i} + (\mathbf{A}^T \mathbf{v})_i, \;\; i = 1, ..., n$$

and the hessian:

$$\nabla^2_{\mathbf{x}} L(\mathbf{x}, \mathbf{v}) = diag(\frac{1}{x_i^2}), \;\; i = 1, ..., n$$

If the hessian is positive definite then L is convex:
Let $\mathbf{z} \in \mathbb{R}^n - \mathbf{0}$

$$\mathbf{z}^T \nabla^2_{\mathbf{x}} L(\mathbf{x}, \mathbf{v}) \mathbf{z} = (z_i)^2 \frac{1}{x_i^2} > 0 \forall \mathbf{z} \in \mathbb{R}^n - \mathbf{0}$$

Since L is convex for a given $\mathbf{v}$ then we can minimize it by:

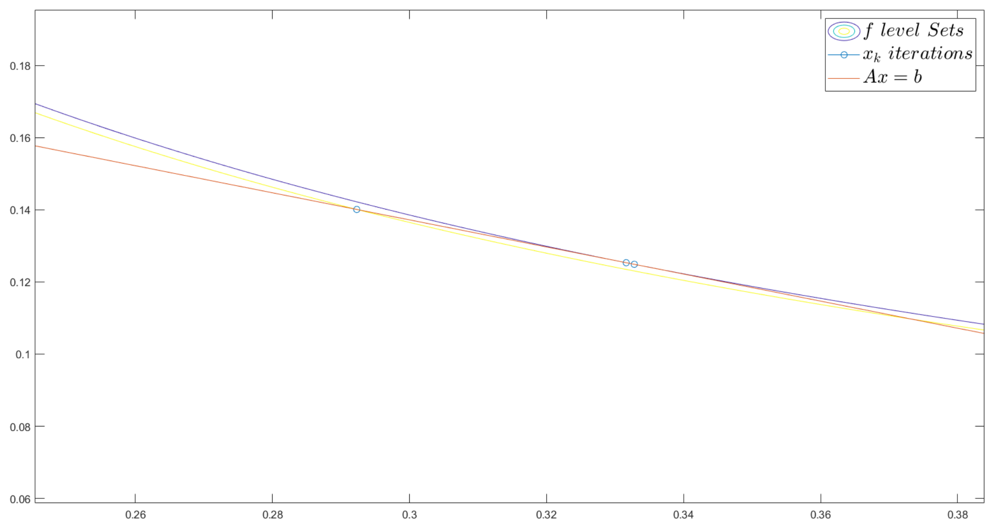
$$\nabla_{\mathbf{x}} L(\mathbf{x}, \mathbf{v}) = 0 \iff \mathbf{x_i} = \frac{1}{(\mathbf{A}^T \mathbf{v})_i} \tag{10}$$

Therefore the dual problem is:

$$maximize g(v) = \underset{\mathbf{x}}{inf} L(\mathbf{x}, \mathbf{v}) = \sum_{i=1}^{n} log(\mathbf{A}^T \mathbf{v})_i + n - \mathbf{b}^T \mathbf{v}$$

We now use cvx to solve the dual and via (10) we calculate the primal.

We observe how in figure 2 the iterations always stay feasible and on $\mathbf{Ax} = \mathbf{b}$, while in figure 3 we start from from an infeasible point and approach $\mathbf{Ax} = \mathbf{b}$.




6.

Figure 2: $\mathbf{Ax} = \mathbf{b}$, f level sets and newton algorithm progress starting from feasible point

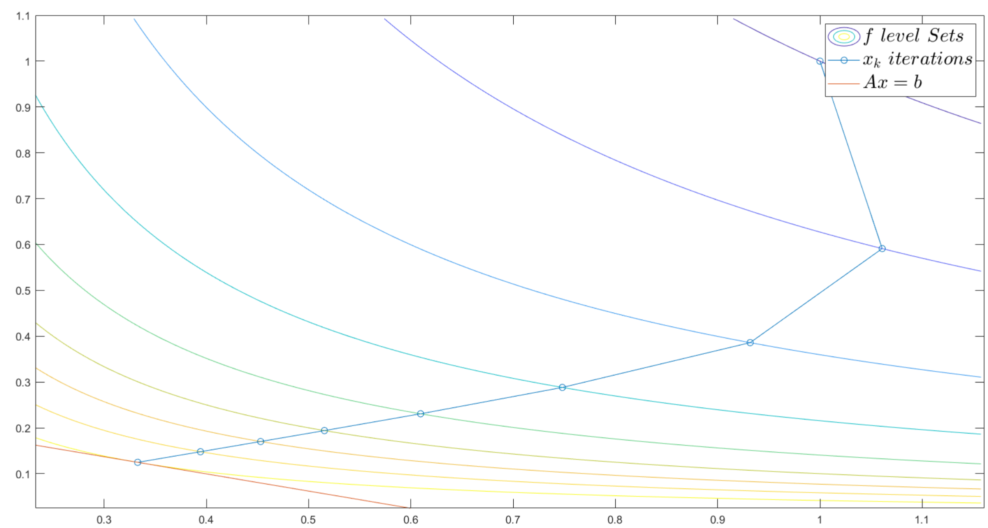



Figure 3: $\mathbf{Ax} = \mathbf{b}$, f level sets and primal-dual algorithm progress with starting from infeasible point

**Exercise4.m**: This script contains the code for the execution of the above.

```
clear; clc; close all;
```

```
% 1. Data Generation
p = 1;
n = 2;

A = rand(p,n);
x_s = rand(n,1);

b = A*x_s;

f=@(x) -sum(log(x));
f2=@(x1,x2) -log(x1)-log(x2);
% 2. CVX solution
cvx_begin
    variable x(n,1)
    minimize( - sum(log(x)) );
    subject to
        A*x == b;
cvx_end
fprintf('Search Time: %e secs\n\n',cvx_cputime);

x_cvx = cvx_optpnt.x;

% 3. Newton solution

% 3.i. Compute a feasible point via cvx
cvx_begin
    variable x(n,1)
    minimize( 0 );
    subject to
        A*x==b;
        x > 0;
cvx_end
```

```
x_0 = cvx_optpnt.x;

% 3.ii. Compute solution with Newton algorithm starting from x0
grad_f = @(x) -1./x;
hessian_f = @(x) diag(1./(x.^2));

alpha = 0.25;
beta = 0.7;
x_k = x_0;
k_newt = 0;
x_vals_newt = x_0;
f_vals_newt = f(x_0);
epsilon=10^-10;
tic;
while(1)
    g_f = grad_f(x_k);
    h_f = hessian_f(x_k);

    w = -inv( A/h_f*A' )*A/h_f*g_f;
    dx = -inv(h_f)*(g_f+A'*w);

    t_k = 1;
    while min(x_k+t_k*dx) < 0
        t_k = beta*t_k;
    end

    while f(x_k+t_k*dx) > f(x_k) + alpha*t_k*g_f'*dx
        t_k = beta*t_k;
    end
    x_k = x_k + t_k*dx;

    x_vals_newt = [x_vals_newt x_k];
    f_vals_newt = [f_vals_newt f(x_k)];
```

```
    k_newt = k_newt+1;

    lambda2=dx'*h_f*dx;
    if(lambda2/2 <= epsilon)
        break;
    end
end
tEnd=toc;
fprintf("Newton descend from feasible point run time:%d\n",tEnd);
min_newt_x = min(x_vals_newt(1,:));
max_newt_x = max(x_vals_newt(1,:));
min_newt_y = min(x_vals_newt(2,:));
max_newt_y = max(x_vals_newt(2,:));
[x_1, x_2] = meshgrid(min_newt_x-0.1 : (max_newt_x-min_newt_x)/100 : max

figure;
contour(x_1,x_2,f2(x_1,x_2),f_vals_newt);
hold on;
plot(x_vals_newt(1,:),x_vals_newt(2,:),'-o');
plot(x_1(1,:),(b-A(1)*x_1(1,:))/A(2));
hold off;
legend({'$f\ level\ Sets$','$x_k\ iterations$','$Ax=b$'},'Interpreter','l
xlim([min_newt_x-0.1 max_newt_x+0.1]);
ylim([min_newt_y-0.1 max_newt_y+0.1]);

% 3.iii Plot the norms of x_k in each iteration against the solution from
% cvx
norm_xk = zeros(1,size(x_vals_newt,2));
norm_x_cvx = zeros(1,size(x_vals_newt,2));
for i=1:k_newt+1
    norm_xk(i)=norm(x_vals_newt(:,i));
    norm_x_cvx(i)=norm(x_cvx);
end
```

```
figure;
plot(0:k_newt,norm_x_cvx,'--');
hold on;
plot(0:k_newt,norm_xk,'--*');
hold off;
title('$Plot\ x_k\ norms\ in\ each\ iteration\ against\ cvx\ norm\ soluti
xlabel('$k$','Interpreter','latex','fontSize',18);
legend({'$||x_{cvx}||_2$','$||x_k||_2$'},'Interpreter','latex','fontSize'
pause(0.1);

% 4.
x_0 = ones(n,1);
x_k = x_0;
v_k = ones(p,1);
k_newtpd = 0;
x_vals_newtpd = x_0;
f_vals_newtpd = f(x_0);
r = @(x,v) [grad_f(x)+A'*v ; A*x-b];
tic;
while(1)
    g_f = grad_f(x_k);
    h_f = hessian_f(x_k);

    dv = -inv( A/h_f*A')*(-(A*x_k-b)+A/h_f*(g_f+A'*v_k));
    dx = -inv(h_f)*(g_f+A'*v_k+A'*dv);

    t_k = 1;
    while min(x_k+t_k*dx) < 0
        t_k = beta*t_k;
    end

    while norm(r(x_k+t_k*dx,v_k+t_k*dv)) > (1-alpha*t_k)*norm(r(x_k, v_k)
```

```
        t_k = beta*t_k;
    end
    x_k = x_k + t_k*dx;
    v_k = v_k + t_k*dv;

    x_vals_newtpd = [x_vals_newtpd x_k];
    f_vals_newtpd = [f_vals_newtpd f(x_k)];
    k_newtpd = k_newtpd+1;

    if norm(r(x_k,v_k))<= epsilon
        break;
    end
end
tend=toc;
fprintf("Dual primal newton descend from any point run time:%d\n",tEnd);

min_newtpd_x = min(x_vals_newtpd(1,:));
max_newtpd_x = max(x_vals_newtpd(1,:));
min_newtpd_y = min(x_vals_newtpd(2,:));
max_newtpd_y = max(x_vals_newtpd(2,:));
[x_1, x_2] = meshgrid(min_newtpd_x-0.1 : (max_newtpd_x-min_newtpd_x)/100

figure;
contour(x_1,x_2,f2(x_1,x_2),f_vals_newtpd);
hold on;
plot(x_vals_newtpd(1,:),x_vals_newtpd(2,:),'-o');
plot(x_1(1,:),(b-A(1)*x_1(1,:))/A(2));
hold off;
legend({'$f\ level\ Sets$','$x_k\ iterations$','$Ax=b$'},'Interpreter','l
xlim([min_newtpd_x-0.1 max_newtpd_x+0.1]);
ylim([min_newtpd_y-0.1 max_newtpd_y+0.1]);

norm_xkpd = zeros(1,size(x_vals_newtpd,2));
```

```
norm_x_cvxpd = zeros(1,size(x_vals_newtpd,2));

for i=1:k_newtpd+1
    norm_xkpd(i)=norm(x_vals_newtpd(:,i));
    norm_x_cvxpd(i)=norm(x_cvx);
end

figure;
plot(0:k_newtpd,norm_x_cvxpd,'--');
hold on;
plot(0:k_newtpd,norm_xkpd,'--*');
hold off;
title('$Plot\ x_k\ norms\ in\ each\ iteration\ against\ cvx\ norm\ soluti
xlabel('$k$','Interpreter','latex','fontSize',18);
legend({'$||x_{cvx}||_2$','$||x_k||_2$'},'Interpreter','latex','fontSize'

% 5.
cvx_begin
    variable v(p,1)
    maximize( -b'*v + n + sum(log(A'*v)) );
cvx_end

v_dual = cvx_optpnt.v;

x_primal = 1./(A'*v_dual);
```