Technical University of Crete

School of Electrical and Computer Engineering

Course : **Optimization**

Exercise 2

Instructor : Athanasios P. Liavas

Students : Toganidis Nikos - 2018030085

Tzimpimpaki Evangelia - 2018030165

(A) Let $f : \mathbb{R}^n \to \mathbb{R}$. For fixed $\mathbf{x} \in \mathbb{R}^n$ and $m > 0$, let $g_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}$ be defined as

$$g_{\mathbf{x}}(\mathbf{y}) := f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \tfrac{m}{2}||\mathbf{y} - \mathbf{x}||_2^2$$

(i) Computing $\nabla g_{\mathbf{x}}(\mathbf{y})$

$$\nabla g_{\mathbf{x}}(\mathbf{y}) = \frac{d}{d\mathbf{y}}\left(g_{\mathbf{x}}(\mathbf{y})\right) = \frac{d}{d\mathbf{y}}\left(f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{m}{2}||\mathbf{y} - \mathbf{x}||_2^2\right) =$$

$$= \frac{d}{d\mathbf{y}}\left(f(\mathbf{x})\right) + \frac{d}{d\mathbf{y}}(\nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})) + \frac{d}{d\mathbf{y}}\left(\frac{m}{2}||\mathbf{y} - \mathbf{x}||_2^2\right) =$$

$$= 0 + \frac{d}{d\mathbf{y}}(\nabla f(\mathbf{x})^T\mathbf{y}) + \frac{m}{2} \cdot \frac{d}{d\mathbf{y}}\left(||\mathbf{y} - \mathbf{x}||_2^2\right) \overset{(\star),(\star\star)}{=\!=\!=} \nabla f(\mathbf{x}) + m \cdot (\mathbf{y} - \mathbf{x})$$

Using the identities $\frac{d}{d\mathbf{y}}(\mathbf{y}^T\mathbf{a}) = \mathbf{a}$ and $\frac{d}{d\mathbf{y}}(\mathbf{y}^T\mathbf{A}\mathbf{y}) = (\mathbf{A} + \mathbf{A^T})\mathbf{y}$ we get :

$$(*) : \frac{d}{d\mathbf{y}}(\nabla f(\mathbf{x})^T\mathbf{y}) = \frac{d}{d\mathbf{y}}(\mathbf{y}^T\nabla f(\mathbf{x})) = \nabla f(\mathbf{x})$$

$$(**) : \frac{d}{d\mathbf{y}}\left(||\mathbf{y} - \mathbf{x}||_2^2\right) = \frac{d}{d\mathbf{y}}((\mathbf{y} - \mathbf{x})^T(\mathbf{y} - \mathbf{x})) = \frac{d}{d\mathbf{y}}((\mathbf{y}^T - \mathbf{x}^T)(\mathbf{y} - \mathbf{x})) =$$

$$= \frac{d}{d\mathbf{y}}(\mathbf{y}^T\mathbf{y}) - \frac{d}{d\mathbf{y}}(\mathbf{y}^T\mathbf{x}) - \frac{d}{d\mathbf{y}}(\mathbf{x}^T\mathbf{y}) + \frac{d}{d\mathbf{y}}(\mathbf{x}^T\mathbf{x}) =$$

$$= 2(\mathbf{I} + \mathbf{I^T})\mathbf{y} - \mathbf{x} - \mathbf{x} + 0 = 2\,(\mathbf{y} - \mathbf{x})$$

(ii) Computing the optimal point $\mathbf{y}_* = \underset{\mathbf{y}}{\operatorname{argmin}}\, g_{\mathbf{x}}(\mathbf{y})$ and the optimal value $g_{\mathbf{x}}(\mathbf{y}_*)$

The gradient of g must be zero at the optimal point $\mathbf{y}_*$, so $\mathbf{y}_*$ will be :

$$\nabla g_{\mathbf{x}}(\mathbf{y}_*) = 0 \Leftrightarrow \nabla f(\mathbf{x}) + m \cdot (\mathbf{y}_* - \mathbf{x}) = 0 \Leftrightarrow \mathbf{y}_* = \mathbf{x} - \frac{\nabla f(\mathbf{x})}{m}$$

And the optimal value $g_{\mathbf{x}}(\mathbf{y}_*)$ will be :

$$\begin{aligned} g_{\mathbf{x}}(\mathbf{y}_*) &= g_{\mathbf{x}}\left(\mathbf{x} - \tfrac{\nabla f(\mathbf{x})}{m}\right) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \left(\mathbf{x} - \tfrac{\nabla f(\mathbf{x})}{m} - \mathbf{x}\right) + \frac{m}{2}||\mathbf{x} - \tfrac{\nabla f(\mathbf{x})}{m} - \mathbf{x}||_2^2 \\ &= f(\mathbf{x}) + \nabla f(\mathbf{x})^T \left(-\tfrac{\nabla f(\mathbf{x})}{m}\right) + \frac{m}{2}|| - \tfrac{\nabla f(\mathbf{x})}{m}||_2^2 = \\ &= f(\mathbf{x}) - \frac{1}{m}\nabla f(\mathbf{x})^T \cdot \nabla f(\mathbf{x}) + \frac{m}{2} \cdot \frac{1}{m^2} \cdot ||\nabla f(\mathbf{x})||_2^2 = \\ &= f(\mathbf{x}) - \frac{1}{m}||\nabla f(\mathbf{x})||_2^2 + \frac{1}{2m} \cdot ||\nabla f(\mathbf{x})||_2^2 = \\ &= f(\mathbf{x}) - \frac{1}{2m} \cdot ||\nabla f(\mathbf{x})||_2^2 \end{aligned}$$

(B) THE GRADIENT METHOD (EXACT AND BACKTRACKING LINE SEARCH)

Problem : $\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\, f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x}$, where $\mathbf{P} \in \mathbb{R}^{n\times n}, \mathbf{P} = \mathbf{P}^T$, $\mathbf{q} \in \mathbb{R}^n$

First we need to construct the vectors $\mathbf{x}$, $\mathbf{q}$ and the matrix $\mathbf{P}$. Matrix $\mathbf{P}$ is random positive definite, so it can be expressed as $\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ with $\mathbf{U}, \mathbf{\Lambda} \in \mathbb{R}^{n\times n}, \mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}_n$ (orthonormal matrix) and $\mathbf{\Lambda} = diag(\lambda_1, ..., \lambda_n)$. The columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{P}$ and the elements of the diagonal of $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{P}$.

(i) Constructing a random orthonormal matrix $\mathbf{U}$

In MATLAB this can be done by creating a random $n \times n$ matrix $A = rand(n, n)$ and then computing its singular value decomposition as $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(A)$.

Then, we compute the matrixes $\mathbf{U}\mathbf{U}^T$ and $\mathbf{U}^T\mathbf{U}$. It is confirmed that $\mathbf{U}$ is orthonormal, since the property $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}_n$ is satisfied.

```
matrix_UUT =

    1.0000    0.0000
    0.0000    1.0000


matrix_UTU =

    1.0000    0.0000
    0.0000    1.0000
```

Constructing diagonal matrix $\mathbf{\Lambda}$

To construct the eigenvalues λ_i for $i = 1, ..., n$, we select the smallest and largest λ_{min} and λ_{max} respectively. Let's select $\lambda_{min} = 1$, so λ_{max} will be equal to $\lambda_{max} = K \cdot \lambda_{min} = K$, where K is the condition number. Then, we create (n-2) random numbers, uniformly distributed in the interval $[\lambda_{min}, \lambda_{max}] = [1, K]$. These will be the rest (n-2) eigenvalues λ_i. Matrix $\mathbf{\Lambda}$ is then constructed as a matrix with the eigenvalues on its diagonal elements and zeros on the rest elements.

(ii) Constructing random vector $\mathbf{q}$ and random positive definite matrix $\mathbf{P}$.

The random vector $\mathbf{q}$ is created as $\mathbf{q} = rand(n, 1)$ and the random positive definite matrix $\mathbf{P}$ is created as $\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$.

(iii) Solving the convex quadratic problem using the closed-form solution

CLOSED-FORM SOLUTION : A point $\mathbf{x}_* \in \mathbb{R}^n$ is the solution of $(\mathbf{1})$ if and only if $\nabla f(\mathbf{x}_*) = 0$.

Given that $\mathbf{x} \in \mathbb{R}^n$, we calculate the gradient of f as :

$$\nabla f(\mathbf{x}) = \frac{d}{d\mathbf{x}}\left(f(\mathbf{x})\right) = \frac{d}{d\mathbf{x}}\left(\frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x}\right) = \frac{1}{2}\cdot\frac{d}{d\mathbf{x}}\left(\mathbf{x}^T\mathbf{P}\mathbf{x}\right) + \frac{d}{d\mathbf{x}}\left(\mathbf{q}^T\mathbf{x}\right) =$$

$$= \frac{1}{2}(\mathbf{P} + \mathbf{P}^T)\mathbf{x} + (\mathbf{q}^T)^T \overset{(\star)}{=\!=} \frac{1}{2}\left(2\mathbf{P}\right)\mathbf{x} + \mathbf{q} = \mathbf{P}\mathbf{x} + \mathbf{q}$$

$(*)$: $\mathbf{P}$ is symmetric, which means that $\mathbf{P}^T = \mathbf{P}$.

So the optimal point $\mathbf{x}_*$ will be :

$$\nabla f(\mathbf{x}_*) = 0 \Leftrightarrow \mathbf{P}\mathbf{x}_* = -\mathbf{q} \Leftrightarrow \mathbf{P}^{-1}\mathbf{P}\mathbf{x}_* = -\mathbf{P}^{-1}\mathbf{q} \Leftarrow \boxed{\mathbf{x}_* = -\mathbf{P}^{-1}\mathbf{q}}$$

(The inverse of a symmetric positive definite matrix $\mathbf{P}$ always exists)

And the optimal value $\mathrm{p}_* = f(\mathbf{x}_*)$ will be :

$$\mathrm{p}_* = f(\mathbf{x}_*) = \frac{1}{2}\mathbf{x}_*^T\mathbf{P}\mathbf{x}_* + \mathbf{q}^T\mathbf{x}_* = \frac{1}{2}\left(-\mathbf{P}^{-1}\mathbf{q}\right)^T\mathbf{P}\left(-\mathbf{P}^{-1}\mathbf{q}\right) + \mathbf{q}^T\left(-\mathbf{P}^{-1}\mathbf{q}\right) =$$

$$= \frac{1}{2}\mathbf{q}^T\left(\mathbf{P}^{-1}\right)^T\mathbf{P}\mathbf{P}^{-1}\mathbf{q} - \mathbf{q}^T\mathbf{P}^{-1}\mathbf{q} = \frac{1}{2}\mathbf{q}^T\left(\mathbf{P}^{-1}\right)^T\mathbf{q} - \mathbf{q}^T\mathbf{P}^{-1}\mathbf{q} =$$

$$\underline{\underline{\left(\mathbf{P}^{-1}\right)^T = \left(\mathbf{P}^T\right)^{-1}}} \frac{1}{2}\mathbf{q}^T\left(\mathbf{P}^T\right)^{-1}\mathbf{q} - \mathbf{q}^T\mathbf{P}^{-1}\mathbf{q} = \frac{1}{2}\mathbf{q}^T\mathbf{P}^{-1}\mathbf{q} - \mathbf{q}^T\mathbf{P}^{-1}\mathbf{q} =$$

$$= -\frac{1}{2}\mathbf{q}^T\mathbf{P}^{-1}\mathbf{q}$$

In MATLAB we calculate $\mathbf{x}_*$ and p_* using the results above. Later, these values will be compared to the results of the exact and backtracking line search to make sure the gradient algorithm is applied correctly.

(iv) Solving the convex quadratic problem using the cvx

After downloading the cvx, we run the following piece of code :

```
cvx_begin
    variable x(n)
    minimize( f(x,P,q) )
cvx_end
```

We observe that the variable cvx_optval is equal to p_star. So, the theoretical optimal value is the same with the one provided by the cvx.

(v) Solving the convex quadratic problem using the gradient algorithm

THE GRADIENT ALGORITHM :

$\mathbf{x}_0 \in \mathbb{R}^n,\ k = 0.$

While (stopping criterion is FALSE)

1. $\Delta \mathbf{x}_k := -\nabla f(\mathbf{x}_k)$.
2. Line search and choose t_k.
3. $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \Delta \mathbf{x}_k$.
4. $k := k + 1$.

Stopping criterion is $\|\nabla f(\mathbf{x_k})\|^2 < \epsilon \Leftrightarrow \|\mathbf{P}\,\mathbf{x_k} + \mathbf{q}\|^2 < \epsilon$, for a small $\epsilon > 0$.

- Exact line search :

  Let $g(t) := f(\mathbf{x} - t\nabla f(\mathbf{x}))$.

  We want to calculate $t_* = \underset{t \geq 0}{\operatorname{argmin}} g(t)$.

  Using the equation (5.17) from chapter 5 of the notes, $g(t)$ is written as :

$$g(t) = \frac{1}{2}\nabla f(\mathbf{x})^T \mathbf{P}\, \nabla f(\mathbf{x})\, t^2 - (\|\nabla f(\mathbf{x})\|^2)t + \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x}$$

  And the derivative of g will be : $\dfrac{dg(t)}{dt} = \left(\nabla f(\mathbf{x})^T\, \mathbf{P}\, \nabla f(\mathbf{x})\right) t - \|\nabla f(\mathbf{x})\|^2$

  So, t_* will be the solution of : $\dfrac{d}{dt_*} g(t_*) = 0 \Rightarrow \boxed{t_* = \dfrac{\|\nabla f(\mathbf{x})\|^2}{\nabla f(\mathbf{x})^T \mathrm{P} \nabla f(\mathbf{x})}}$

CODE RESULTS :

- The optimal point of the exact line search is obtained in the last repetition of the while loop. We can check that the optimal point $\mathbf{x_k}$ of the exact line search is equal to the optimal point of the closed-form solution $\mathbf{x}_*$, by viewing these vectors.

- The optimal value using the exact line search is equal to the optimal value using the closed-form solution.

- Backtracking line search :

  For the backtracking line search t is not known. It is first defined as $t = 1$, and then it is decreased by being multiplied to a factor $b\epsilon(0, 1)$. Using the terminal condition $f(\mathbf{x} + t\mathbf{Dx}) > f(\mathbf{x}) + \alpha t \nabla f(\mathbf{x})^T \mathbf{Dx}$ inside a while loop, we get the final value of t, for which the optimal point $\mathbf{x_{kk}}$ is calculated.

  For every iteration of the gradient algorithm, we print the iteration number and the function value at this specific iteration. The optimal value of the backtracking line search is the one printed in the last iteration. We can check that it is equal to the optimal value p_* using the closed-form solution.

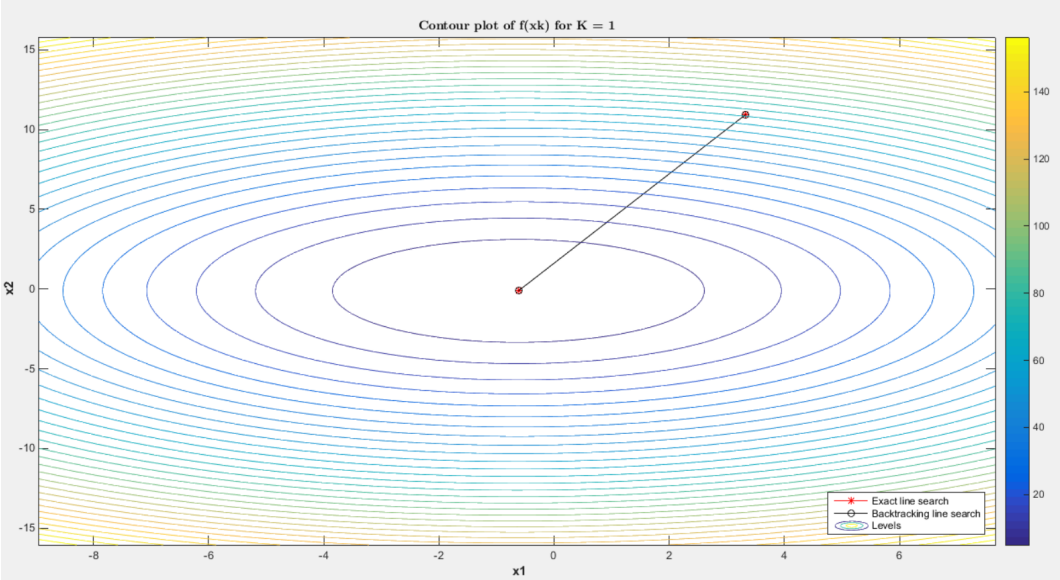
- Running the code with K = 1, we can see that only 1 iteration is necessary for convergence for the exact line search.

(vi) Contour plot of $f(\mathbf{x_k})$

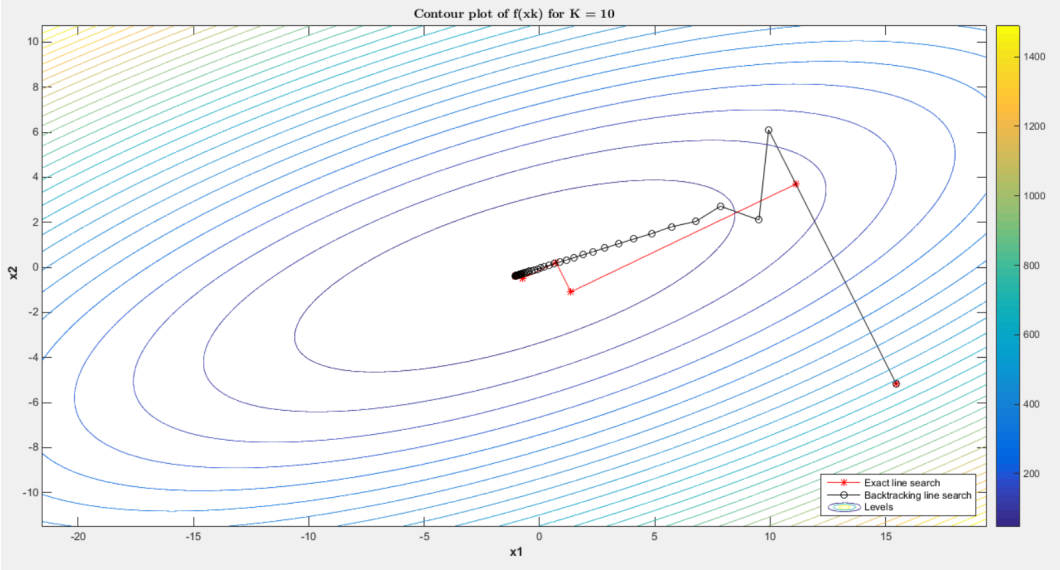
We construct the contour plot of $f(\mathbf{x_k})$ for n = 2 and different values of K. On top of this, we plot the trajectories of $\mathbf{x_k}$ produced by the two algorithms.

We can see that exact line search requires less iterations for convergence, since every iteration is much more effective (less iterations, bigger step).

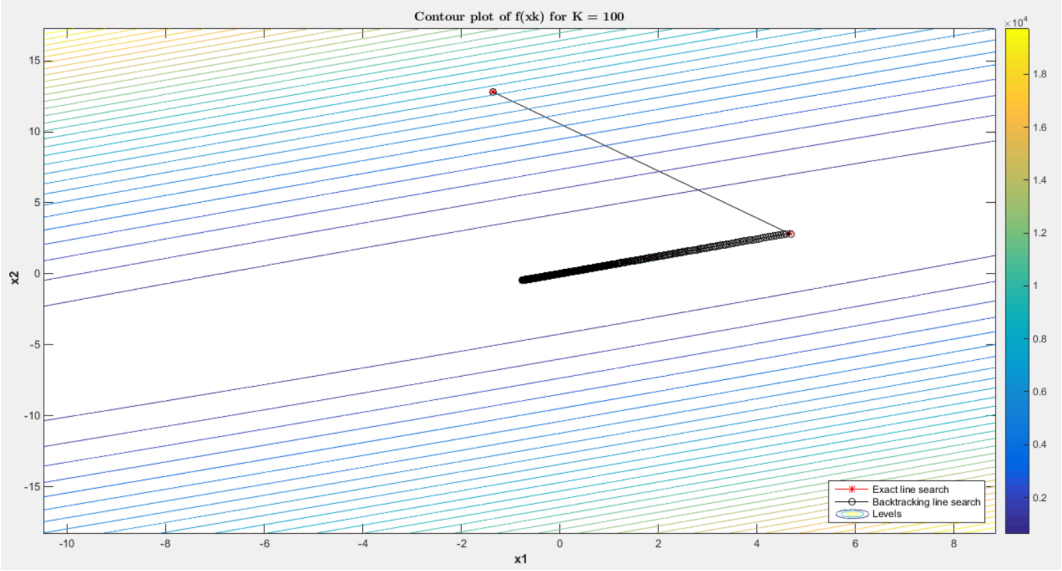
For K = 1 :



For K = 10 :



For K = 100 :

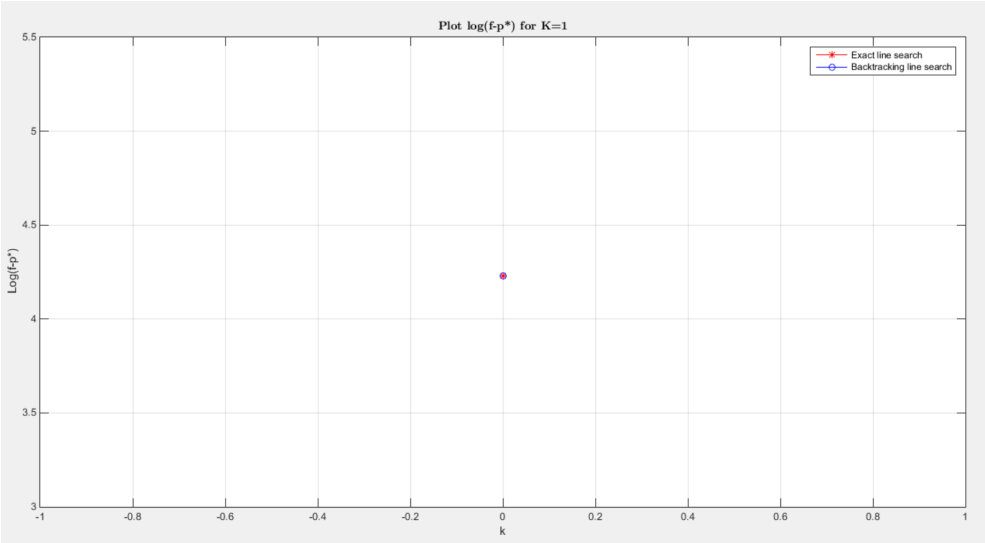


For problems with large condition number, we observe the "zig-zag" effect.

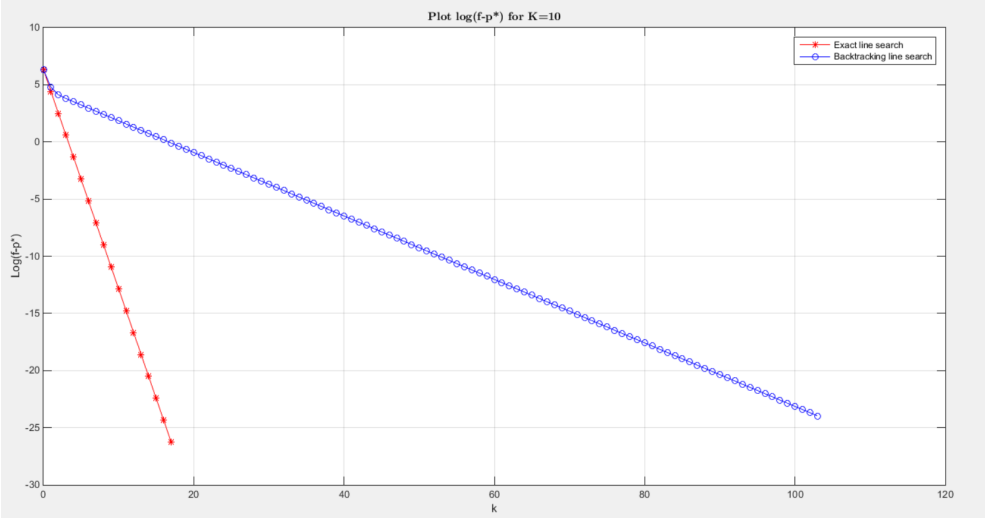
(vii) Plotting quantity $\log\left(f\left(\mathbf{x}_k\right) - p_*\right)$, produced by the two algorithms, versus k.

The difference $f\left(\mathbf{x}_k\right) - p_*$ is calculated and saved as a new vector inside the while loop for the exact and the backtracking line search. So, now we only need to plot the requested signals versus the iteration number k.

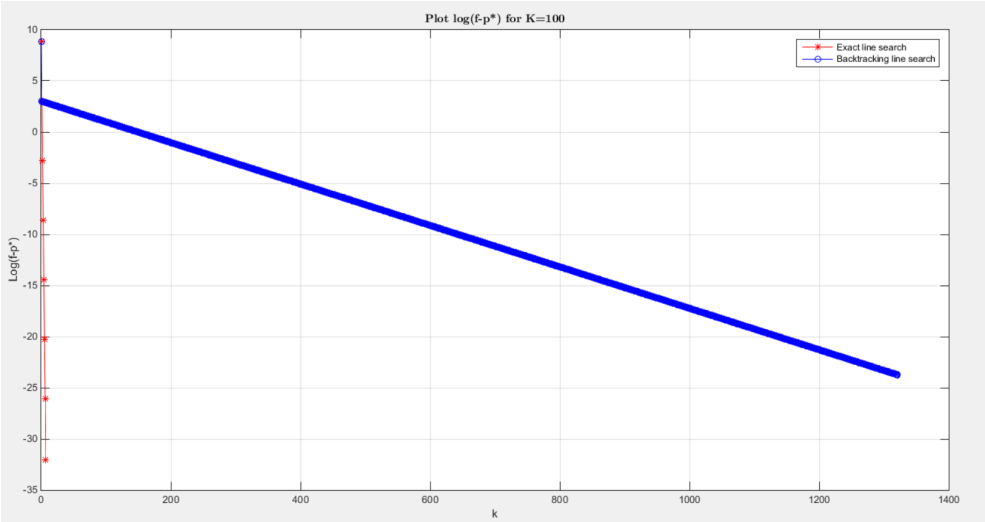
Results are depicted bellow for K = 1 :



For K = 10 :



And for K = 100 :



For small values of condition number K, both algorithms converge without taking too long. However, as K is getting bigger, backtracking line search is not as effective as exact line search, since it requires much more iterations to converge.

For example, for K = 100, backtracking line search requires almost 100 times the iterations required in exact line search.

The slope of this plot shows the rate at which each line search method converges to the optimal value $p_\star$.

$$\text{slope} = \frac{\text{change in y}}{\text{change in x}} = \frac{log(f(x_0) - p_*)}{\text{iterations}}$$

Also, we know that :

$$c = \begin{cases} 1 - \frac{\lambda_{\min}}{\lambda_{\max}} = 1 - \frac{1}{K}, \text{ for exact line search} \\ 1 - \min\left\{2\lambda_{\min} a, \frac{2ab\lambda_{\min}}{\lambda_{\max}}\right\} = 1 - \min\left\{2a, \frac{2ab}{K}\right\}, \text{ for backtracking line search} \end{cases}$$

which means that for big values of K, c is almost 1 and the gradient algorithm generally requires more repetitions to converge.

(viii) Minimum number of iterations that guarantees solution within accuracy

The minimum number of iterations that guarantees solution within accuracy ϵ, satisfies the condition : $f(\mathbf{x}_k) - p \leq \epsilon$. Using relation (5.44) of the notes, we equally get :

$$k \geq k_e := \frac{\log\left(\frac{f(\mathbf{x_0}) - p_*}{\epsilon}\right)}{\log\left(\frac{1}{c}\right)} \Rightarrow k_{min} = \frac{\log\left(\frac{f(\mathbf{x_0}) - p_*}{\epsilon}\right)}{\log\left(\frac{1}{c}\right)}$$

In MATLAB we calculate the minimum value of k for the two kinds of line search, as shown above (**theoretical k**). This value is depicted bellow, along with the real number of iterations required (**practical k**). It is obvious that for bigger values of K the algorithm requires more repetitions. Also, the exact line search is proved to be a much better method and finally, the theoretical value of iterations is always bigger than the practical value.

	K = 1	**K = 10**	**K = 100**	**K = 1000**
EXACT practical	1	6	5	5
EXACT theoretical	1	165	1663	22769
BACKTRACKING practical	1	87	1058	10057
BACKTRACKING theoretical	42	569	5563	75921

(C) BACKTRACKING LINE SEARCH AT GRADIENT AND NEWTON METHODS

Problem: $\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\, f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \sum_{i=1}^{m} \log\left(b_i - \mathbf{a}_i^T\mathbf{x}\right) = \mathbf{c}^T\mathbf{x} - \text{sum}(\log(\mathbf{b} - \mathbf{A}\mathbf{x}))$,

where $\mathbf{x} \in \mathbb{R}^n, \mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m\times n}$, with rows a_i^T

Also, $m \gg n$ (indicative value pairs $(n, m) = (2, 20), (50, 200)$).

- Proving that $\mathbf{dom}f$ is convex

  Since the arguments of a logarithm have to be positive, the domain set of f is :

  $$\mathbf{dom}f = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{b} - \mathbf{A}\mathbf{x} > 0\} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} < \mathbf{b}\}$$

  Set $\mathbf{dom}f$ is convex iff $\forall \mathbf{x_1}, \mathbf{x_2} \in \mathbf{dom}f, 0 \le \theta \le 1$, we have :

  $$\theta\mathbf{x_1} + (1-\theta)\mathbf{x_2} \in \mathbf{dom}f$$

  - $\mathbf{x_1} \in \mathbf{dom} \Rightarrow \mathbf{A}\mathbf{x_1} < \mathbf{b} \Rightarrow \theta\mathbf{A}\mathbf{x_1} < \theta\mathbf{b}$, since $\theta > 0$
  - $\mathbf{x_2} \in \mathbf{dom} \Rightarrow \mathbf{A}\mathbf{x_2} < \mathbf{b} \Rightarrow (1-\theta)\mathbf{A}\mathbf{x_2} < (1-\theta)\mathbf{b}$, since $1 - \theta > 0$

  Adding the relations above we prove that $\mathbf{dom}f$ is convex, as following :

  $\theta\mathbf{A}\mathbf{x_1} + (1-\theta)\mathbf{A}\mathbf{x_2} < \theta\mathbf{b} + (1-\theta)\mathbf{b} \Rightarrow \mathbf{A}(\theta\mathbf{x_1} + (1-\theta)\mathbf{x_2}) < \mathbf{b} \Rightarrow \theta\mathbf{x_1} + (1-\theta)\mathbf{x_2} \in \mathbf{dom}f$

- Proving that function f is convex

  A function f is convex if its domain set is convex and its second derivative is positive semi-definite. We have already proved that $\mathbf{dom}f$ is convex, so it remains to show that $D^2 f(\mathbf{x}) \succeq 0$.

  - $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \sum_{i=1}^{m} \log\left(b_i - \mathbf{a}_i^T\mathbf{x}\right) = \mathbf{c}^T\mathbf{x} - \sum_{i=1}^{m} \log\left(b_i - \sum_{k=1}^{n} a_{i_k} x_k\right)$

– $\nabla f(\mathbf{x}) = \frac{d}{d\mathbf{x}}\left(f(\mathbf{x})\right) = \frac{d}{d\mathbf{x}}\left(\mathbf{c}^T\mathbf{x}\right) - \frac{d}{d\mathbf{x}}\left(\sum_{i=1}^m \log\left(b_i - \mathbf{a}_i^T\mathbf{x}\right)\right) \overset{(\star)}{=\!=}$

$= (\mathbf{c}^T)^T + \sum_{i=1}^m \frac{\mathbf{a}_i}{b_i - \mathbf{a}_i^T\mathbf{x}} = \mathbf{c} + \sum_{i=1}^m \frac{\mathbf{a}_i}{b_i - \mathbf{a}_i^T\mathbf{x}}$

$(*) : \frac{d}{dx_j}\left(\sum_{i=1}^m \log\left(b_i - \mathbf{a}_i^T\mathbf{x}\right)\right) = \frac{d}{dx_j}\left(\sum_{i=1}^m \log\left(b_i - \sum_{k=1}^n a_{i_k}x_k\right)\right) =$

$= \sum_{i=1}^m \frac{d}{dx_j}\left(\log\left(b_i - \sum_{k=1}^n a_{i_k}x_k\right)\right) = \sum_{i=1}^m \frac{-a_{i_j}}{b_i - \sum_{k=1}^n a_{i_k}x_k} =$

$= -\sum_{i=1}^m \frac{a_{i_j}}{b_i - \mathbf{a}_i^T\mathbf{x}}$

– $D^2 f(\mathbf{x}) = \frac{d}{d\mathbf{x}}\left(\nabla f(\mathbf{x})\right) = \frac{d}{d\mathbf{x}}\left(\mathbf{c} + \sum_{i=1}^m \frac{\mathbf{a}_i}{b_i - \mathbf{a}_i^T\mathbf{x}}\right) = \frac{d}{d\mathbf{x}}\left(\sum_{i=1}^m \frac{\mathbf{a}_i}{b_i - \mathbf{a}_i^T\mathbf{x}}\right) \overset{(\star)}{=\!=}$

$= \sum_{i=1}^m \frac{\mathbf{a}_i\mathbf{a}_i^T}{(b_i - \mathbf{a}_i^T\mathbf{x})^2}$

$(*) : \frac{d}{dx_j}\left(\sum_{i=1}^m \frac{a_{i_j}}{b_i - \mathbf{a}_i^T\mathbf{x}}\right) = \sum_{i=1}^m \frac{d}{dx_j}\left(\frac{a_{i_j}}{b_i - \sum_{k=1}^n a_{i_k}x_k}\right) =$

$= \sum_{i=1}^m (-a_{i_j})\left(b_i - \sum_{k=1}^n a_{i_k}x_k\right)^{-2}(-a_{i_i}) = \sum_{i=1}^m \frac{a_{i_j}a_{i_i}}{\left(b_i - \sum_{k=1}^n a_{i_k}x_k\right)^2} =$

$= \sum_{i=1}^m \frac{a_{i_j}a_{i_i}}{\left(b_i - \mathbf{a}_i^T\mathbf{x}\right)^2}$

We need to show that $D^2 f(\mathbf{x}) \succeq 0 \Leftrightarrow \sum_{i=1}^m \frac{\mathbf{a}_i\mathbf{a}_i^T}{(b_i - \mathbf{a}_i^T\mathbf{x})^2} \succeq 0 \Leftrightarrow \frac{\mathbf{a}_i\mathbf{a}_i^T}{(b_i - \mathbf{a}_i^T\mathbf{x})^2} \succeq 0$

Since $(b_i - \mathbf{a}_i^T\mathbf{x})^2 \geq \forall i = 1, ..., m$, the equation above is true iff $\mathbf{a}_i\mathbf{a}_i^T, \forall i = 1, ..., m$.

The matrix $\mathbf{a}_i\mathbf{a}_i^T$ is positive semi-definite if the following condition holds :

$$\mathbf{v}^T\mathbf{a}_i\mathbf{a}_i^T\mathbf{v} \geq 0, \forall \mathbf{v} \in \mathbb{R}^n$$

So, $\mathbf{v}^T\mathbf{a}_i\mathbf{a}_i^T\mathbf{v} = (\mathbf{a}_i^T\mathbf{v})^T\mathbf{a}_i^T\mathbf{v} = ||\mathbf{a}_i^T\mathbf{v}||_2^2$, which is indeed non-negative $\forall \mathbf{v} \in \mathbb{R}^n$.

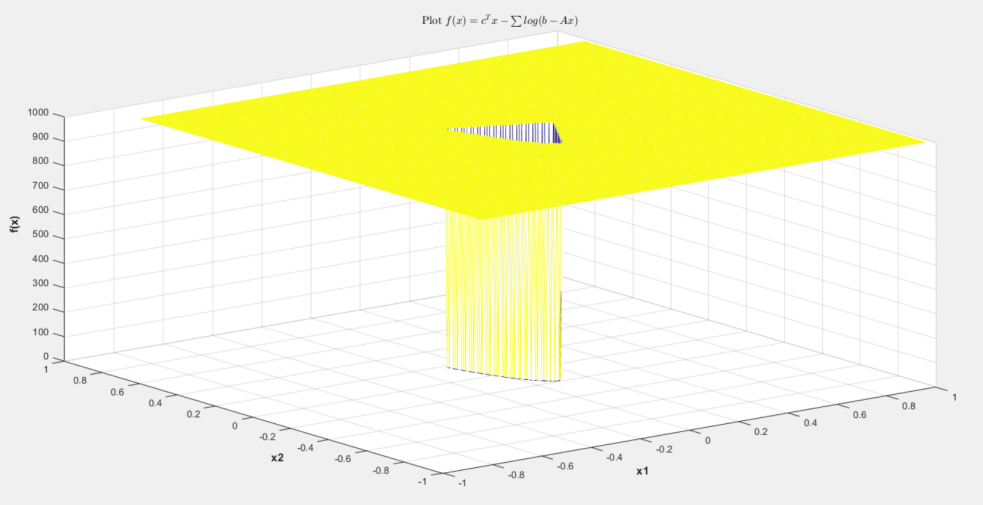
So, we proved that function f is convex.

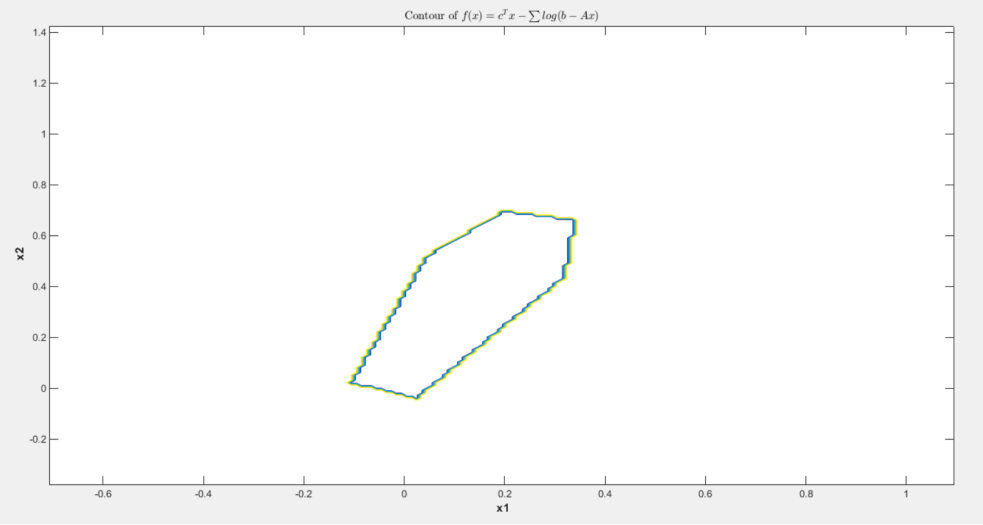
(a) Minimize f using the cvx.

First, we solve the quadratic problem, using the cvx. If the problem has a solution, then cvx will compute it, otherwise it will display a message saying that the problem has no solution.

(b) Plotting f and its level sets in the neighborhood of the optimum point

For $n = 2$ and $m = 20$, we plot f and its level sets in the neighborhood of the optimum point, which was calculated using the cvx. We check for each possible point $\mathbf{x}$, if it belongs to $\mathbf{dom}f$. If it belongs, then we calculate the value of $\mathbf{f}$ at this point, otherwise we assign to $\mathbf{f}$ the value 10^3. This way, it can be shown that in the boundary of $\mathbf{dom}f$, function $\mathbf{f}$ is infinite for this certain point.





(c) Minimize f, using the gradient algorithm with backtracking line search

We consider $\mathbf{x_0} = \mathbf{0}$ as a feasible point. The main difference of the algorithm compared with the baseline implementation is that before starting backtracking, we now check whether the point $\mathbf{x_{k+1}} = \mathbf{x_k} + t\Delta_{x_k}$ belongs to $\mathbf{dom}f$ or not. For the points that don't belong to $\mathbf{dom}f$, we refresh the value of $t = \beta t$ repeatedly until they belong to $\mathbf{domf}$. When a certain point belongs to $\mathbf{domf}$, the typical backtracking line search is then executed.

Comparing the optimal point and optimal value produced by the gradient algorithm with the ones produced by the cvx, we make sure that the algorithm works fine.

(d) Minimize $\mathbf{f}$, using the Newton algorithm with backtracking line search

We consider $\mathbf{x_0} = \mathbf{0}$ as a feasible point. The main difference of the algorithm compared with the baseline implementation is the same as the gradient algorithm.

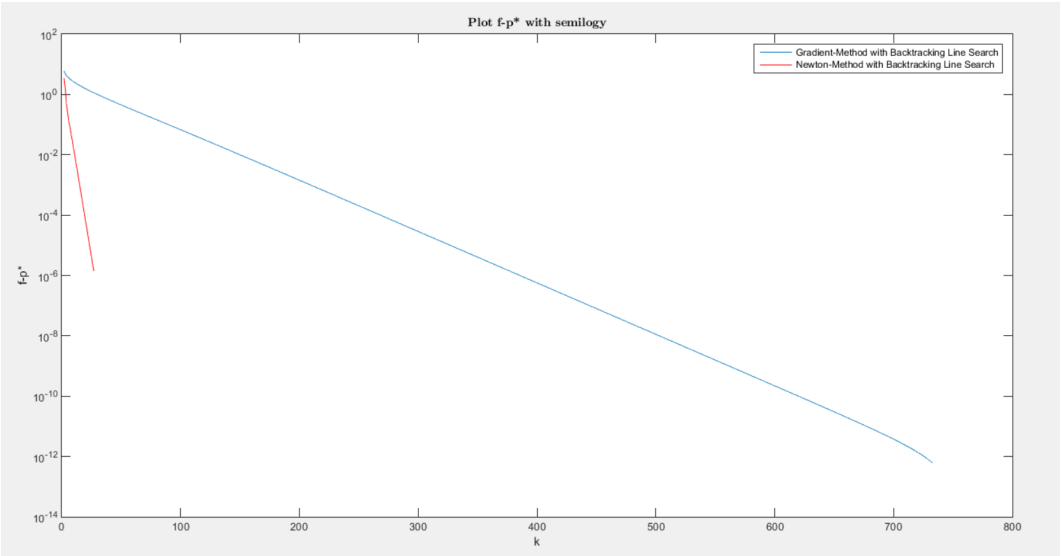
Comparing the optimal point and optimal value produced by the Newton algorithm with the ones produced by the cvx, we make sure that the algorithm works fine.

For example, this is the result of a random code implementation :

```
C.a    Optimal value (CVX) = 18.557464

C.b    Plotting function f and its level sets

C.c    Optimal value (GRADIENT METHOD) = 18.557464
       Number of iterations in GRADIENT METHOD = 160

C.d    Optimal value (NEWTON METHOD) = 18.557464
       Number of iterations in NEWTON METHOD =  10
```

(e) Plotting the quantities $f_{gradient}(\mathbf{x_k}) - p_*$ and $f_{Newton}(\mathbf{x_k}) - p_*$ with semilogy

For $n = 2, m = 20$ we get the following results :



It is obvious that the Newton method converges much faster than the Gradient method, since it requires much less repetitions until it reaches convergence to the optimal value.

We observe that for larger values of the pair (n, m), the Newton method is even more effective, since the difference between the two methods in the number of repetitions for convergence is even bigger now. For example, just by increasing value n from $n = 2$ to $n = 5$ the iterations for the gradient method are doubled while for the Newton method they remain.