Optimization

2nd Problem Set

Kalamarakis Theodoros: 2018030022

19/11/2022

Problem A

(a) (10) Let $f : \mathbb{R}^n \to \mathbb{R}$. For fixed $x \in \mathbb{R}^n$ and $m > 0$, let $g_x : \mathbb{R}^n \to \mathbb{R}$ be defined as

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

i. (5) Compute $\nabla g_x(\mathbf{y})$.

Solution:

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

On point $(*)$ we used that $(*)\frac{\partial(\|\mathbf{x}\|_2^2)}{\partial \mathbf{x}} = \nabla(\|\mathbf{x}\|_2^2)^T = 2\mathbf{x}^T$ as proven in the previous set in Problem 5 (d)

ii. (5) Compute the optimal point, $\mathbf{y}_* = \arg\min_{\mathbf{x}} g_{\mathbf{x}}(\mathbf{y})$, and the optimal value, $g_{\mathbf{x}}(\mathbf{y}_*)$.

Solution:

We can find the optimal point by solving the equation $\nabla g_{\mathbf{x}}(\mathbf{y}) = 0$

$$\nabla g_{\mathbf{x}}(\mathbf{y}) = 0 \Leftrightarrow \nabla f(\mathbf{x}) + m(\mathbf{y} - \mathbf{x}) = 0 \Leftrightarrow \mathbf{y} = \mathbf{y}_* = -\frac{1}{m}\nabla f(\mathbf{x}) + \mathbf{x}$$

Hence the optimal value $g_{\mathbf{x}}(\mathbf{y}_*)$ is:

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

Problem B

(a) (40) Consider the problem

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

where $\mathbf{P} \in \mathbb{R}^{n\times n}, \mathbf{P} = \mathbf{P}^T \succ \mathbf{O}$ and $\mathbf{q} \in \mathbb{R}^n$ (indicative values of n are $n = 2, 50, 102, 103$.)

ii. (5) Construct random vector q and random positive definite matrix P, with condition number $\mathcal{K}$ (indicative values, $\mathcal{K} = 10, 102, 103$).

Solution.

The matrices $\mathbf{P}$ and $\mathbf{q}$ that were generated in matlab are

$$\mathbf{P} = \begin{bmatrix} 1.8018 & 2.5638 \\ 2.5638 & 9.1982 \end{bmatrix} \text{ and } \mathbf{q} = \begin{bmatrix} 0.5886 \\ 0.3519 \end{bmatrix}$$

.

iii. (5) Solve problem (2) using the closed-form solution, and compute the optimal point, $\mathbf{x}_*$, and the optimal value, $\mathbf{p}_* = f(\mathbf{x}_*)$.

Solution:

In order to find the optimal point we have to solve the equation $\nabla f(\mathbf{x}) = 0$

$$\nabla f(\mathbf{x}) = 0 \Leftrightarrow \nabla(\frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x}) = 0 \Leftrightarrow \mathbf{P}\mathbf{x} + \mathbf{q} = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{x}_* = -\mathbf{P}^{-1}\mathbf{q}$$

The result in matlab for $n = 2$ is $\mathbf{p}_* = f(\mathbf{x}_*) = -0.1174$

.

iv. Solve problem (2) using the cvx.

Solution:

The cvx result is $cvx\_optval = -0.117368$ which is very close to the result above calculated by the closed form solution

v. (15) Solve problem (2) using the gradient algorithm (with exact and backtracking line search). How many iterations are necessary for convergence, if $K = 1$ and you use the exact line search?

Solution:

For $\mathcal{K} = 10$ the results of the two algorithms are

- $\mathbf{p}^*_{backtracking} = -0.117344$ in 16 iterations for the backtracking line search algorithm whereas

- $\mathbf{p}^*_{exact} = -0.117366$ in 7 iterations for the exact line search

We observe that both of them are very close to the results above, yet the exact line searching achieved in fewer iterations (this is not the case though for $n > 2$)

For $\mathcal{K} = 1$ the exact line search uses only a single iteration to reach the necessary convergence

vi. (5) For $n = 2$, construct a contour plot, with levels the values $f(\mathbf{x_k})$. Then, on top of this plot, plot the trajectories of xk produced by the two algo- rithms. (For problems with large condition number, you must observe the "zig-zag" effect.)

Solution:

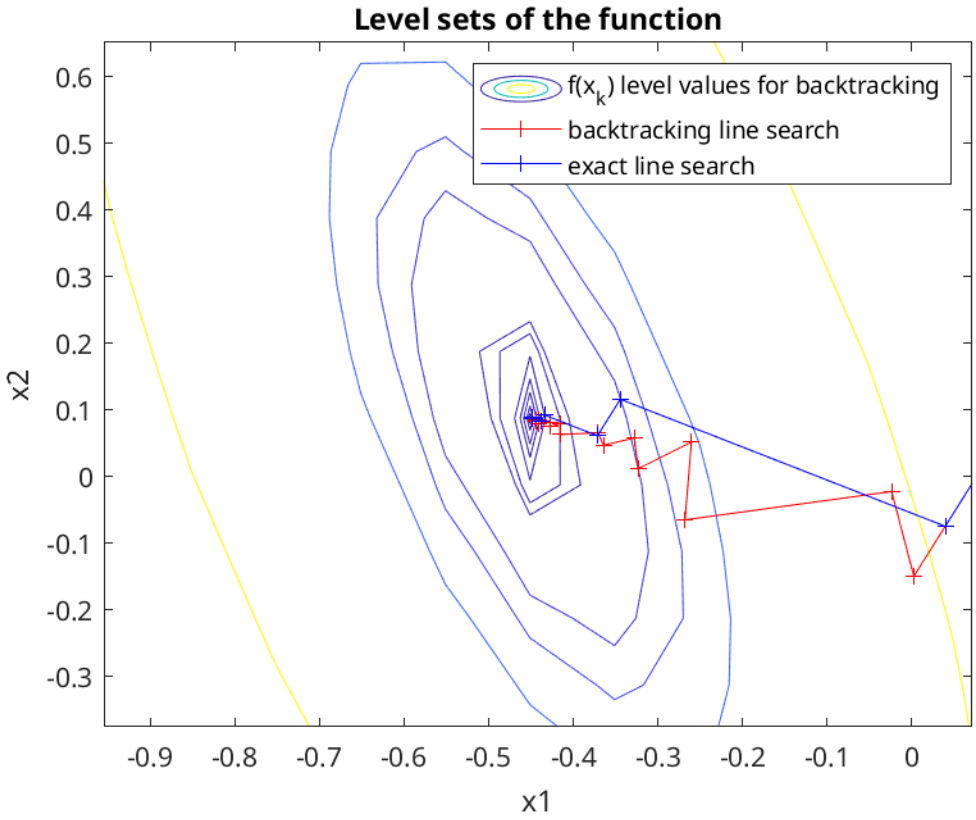


Figure 1: Level set of values $f(\mathbf{x_k})$ (as they were calculated in backtracking line search) alongoside with the trajectories $\{\mathbf{x_k}\}$ by the backtracking line search algorithms and the exact line search

.

.

vii. (5) Plot quantity $log(f(\mathbf{x_k}) - \mathbf{p}_*)$, produced by the two algorithms, versus k. What is the slope of this plot? What do you observe for different condition numbers?.

Solution:

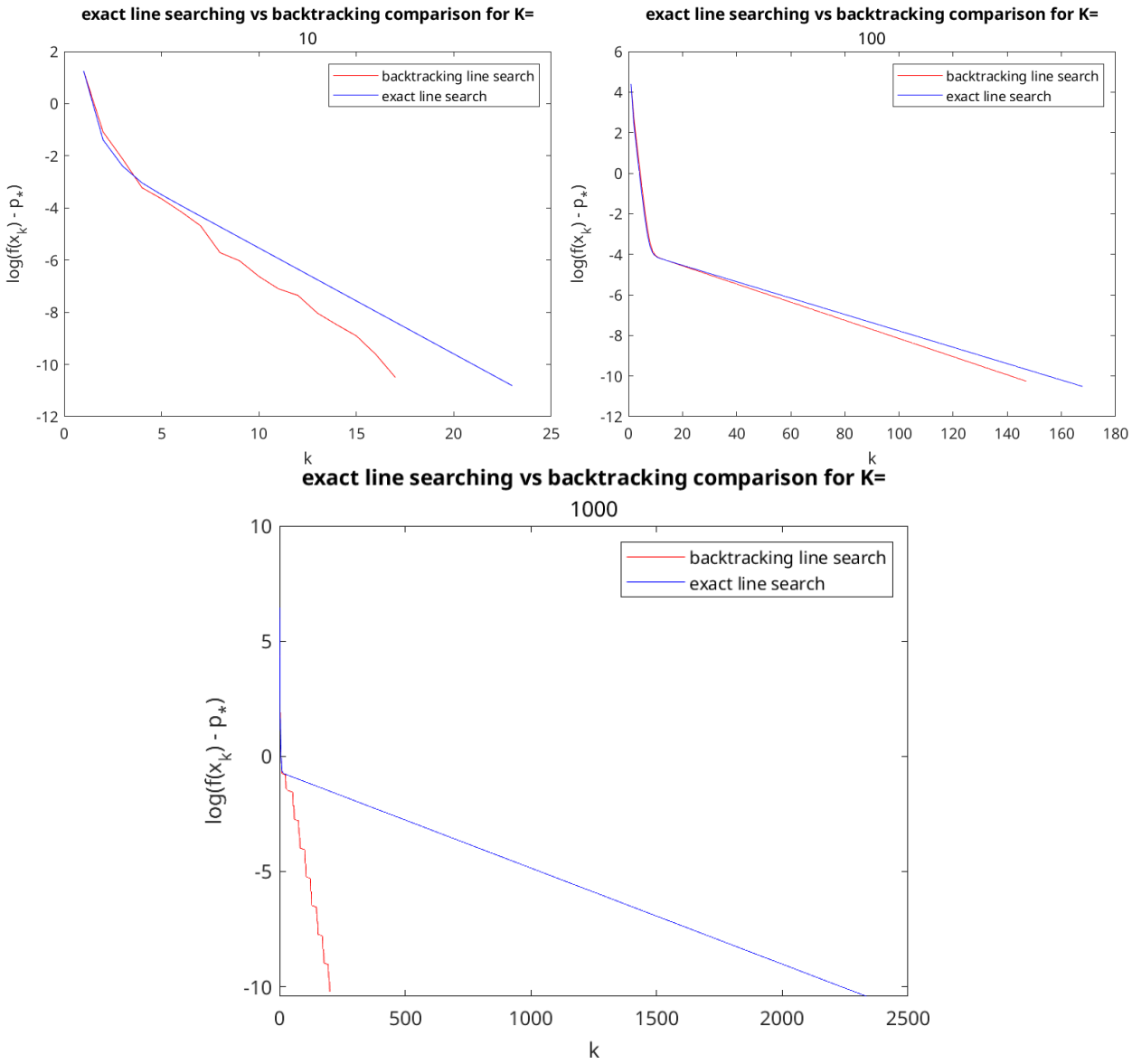


Figure 2: Comparison of the $log(f(\mathbf{x_k}) - \mathbf{p}_*)$ function for the backtracking line search algorithm and the exact line search fro $\mathcal{K} = 10, 100, 1000$

.

We observe that for low $\mathcal{K}$ the two plots have pretty similar descending slopes , yet for $\mathcal{K} = 1000$ the slope of the backtracking line search is much steeper than the exact line search which means that it requires less iterations.

viii. (5) Using the convergence analysis results for strongly convex functions, and the values of $f(\mathbf{x_0})$, $\mathbf{p}_*$ and ϵ, compute the minimum number of iterations

that guarantees solution within accuracy ϵ, i.e., $f(\mathbf{x_k}) - \mathbf{p}_* \leq \epsilon$, and compare it with the number of iterations performed by the algorithms. What do you observe?.

Solution:

The minimum number of iteration is calculated by the equation

$$k_\epsilon = \mathcal{K} \log\left(\frac{f(\mathbf{x_0}) - \mathbf{p}_*}{\epsilon}\right)$$

However if we insert this formula in matlab, the result we get back is $k_\epsilon = 44.1256$ which is obviously wrong, since this number is supposed to be the minimum number of iterations. Yet the iterations used by both of the algorithms is significantly lower.

.

Problem C

We proceed to the solution of a convex unconstrained problem.

(90) Let $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$, with rows $\mathbf{a}_i^T$, for $i = 1, ..., m$. Consider the function $f : \mathbf{dom} f \subseteq \mathbb{R}^n \to \mathbb{R}$, defined as

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \sum_{i=1}^{m} \log(b_i - \mathbf{a_i}^T\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \mathrm{sum}(\log(\mathbf{b_i} - \mathbf{a}_i^T\mathbf{x})),$$

with $\mathbf{x} \in \mathbb{R}^n$ and $m > n$ ($m \gg n$) (indicative value pairs $(n, m) = (2, 20), (50, 200)$. If you have patience and enough memory in the computer, you can set (n, m) = $(300, 800)$ or larger values).

(a) (5) Prove that $\mathbf{dom} f$ is convex.

Solution:

We know that the argument of the logarithm must be positive so:

$$\mathbf{dom} f = \{\mathbf{x} \in \mathbb{R}^n | b_i - \mathbf{a_i}^T\mathbf{x} > 0\ ,\ i = 1, ...m\}$$

Let $\mathbb{S}_i = \{\mathbf{x} \in \mathbb{R}^n | b_i - \mathbf{a_i}^T\mathbf{x} > 0\}$

Then $\mathbf{dom}f$ must be the intersection of all the $\mathbb{S}_i$ for $i = 1, ...m$

$$\mathbf{dom}f = \mathbb{S}_1 \cap \mathbb{S}_2 \cap .... \cap \mathbb{S}_m$$

Which means that if we prove that every one of the $\mathbb{S}_i$ is convex then $\mathbf{dom}f$ will be convex as well

To prove that $\mathbb{S}_i$ is convex we work as follows:

Let $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{S_i}$

Then

$$\left\{ \begin{array}{l} \mathbf{a_i}^T\mathbf{x_1} < b_i \\ \\ \mathbf{a_i}^T\mathbf{x_2} < b_i \end{array} \right\} \overset{\substack{0<\theta_1<1 \\ 0<\theta_2<1}}{\Longleftrightarrow} \left\{ \begin{array}{l} \mathbf{a_i}^T\theta_1\mathbf{x_1} < \theta_1 b_i \\ \\ \mathbf{a_i}^T\theta_2\mathbf{x_2} < \theta_2 b_i \end{array} \right\} \Leftrightarrow \mathbf{a_i}^T(\theta_1\mathbf{x_1} + \theta_2\mathbf{x_2}) < (\theta_1 + \theta_2)b_i \overset{\theta_1+\theta_2=1}{\Longleftrightarrow}$$

$$\overset{\theta_1+\theta_2=1}{\Longleftrightarrow} \mathbf{a_i}^T(\theta_1\mathbf{x_1} + \theta_2\mathbf{x_2}) < b_i$$

Hence $\theta_1\mathbf{x_1} + \theta_2\mathbf{x_2} \in \mathbb{S_i}$ for $\theta_1 + \theta_2 = 1$ which proves that $\mathbb{S}_i$ is convex

(b) (5) Prove that Function f is convex.

Solution:

Initialy we have to calculate the gradient of f

$$\nabla f(\mathbf{x}) = \left(\frac{\partial \mathbf{c}^T\mathbf{x}}{\partial \mathbf{x}}\right)^T - \sum_{i=1}^{m}\left(\frac{\partial \log(b_i - \mathbf{a_i}^T\mathbf{x})}{\partial \mathbf{x}}\right)^T \overset{(*)}{=} \mathbf{c} - \sum_{i=1}^{m}\frac{-1}{b_i - \mathbf{a_i}^T\mathbf{x}}\mathbf{a_i} =$$

$$= \mathbf{c} + \sum_{i=1}^{m}\frac{1}{b_i - \mathbf{a_i}^T\mathbf{x}}\mathbf{a_i}$$

Now we have to calculate the Hessian of f

$$\nabla^2 f(\mathbf{x}) = \left(\frac{\partial \nabla f(\mathbf{x})}{\partial \mathbf{x}}\right)^T = \left(\frac{\partial \sum_{i=1}^{m} \frac{1}{b_i - \mathbf{a_i}^T \mathbf{x}} \mathbf{a_i}}{\partial \mathbf{x}}\right)^T = \sum_{i=1}^{m} \left(\frac{\partial \frac{1}{b_i - \mathbf{a_i}^T \mathbf{x}} \mathbf{a_i}}{\partial \mathbf{x}}\right)^T \overset{(**)}{=}$$

$$\overset{(**)}{=} -\sum_{i=1}^{m} \frac{-1}{(b_i - \mathbf{a_i}^T \mathbf{x})^2} \mathbf{a_i} \mathbf{a_i}^T = \sum_{i=1}^{m} \frac{1}{(b_i - \mathbf{a_i}^T \mathbf{x})^2} \mathbf{a_i} \mathbf{a_i}^T$$

On point $(*)$ we did

$$\left(\frac{\partial \log(b_i - \mathbf{a_i}^T \mathbf{x})}{\partial \mathbf{x}}\right)^T = \left(\frac{\partial \log(b_i - \sum_{j=1}^{n} a_{ij} x_j)}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial \log(b_i - \sum_{j=1}^{n} a_{ij} x_j)}{\partial x_1} \\ \frac{\partial \log(b_i - \sum_{j=1}^{n} a_{ij} x_j)}{\partial x_2} \\ \vdots \\ \frac{\partial \log(b_i - \sum_{j=1}^{n} a_{ij} x_j)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{-a_{i1}}{b_i - \sum_{j=1}^{n} a_{ij} x_j} \\ \frac{-a_{i2}}{b_i - \sum_{j=1}^{n} a_{ij} x_j} \\ \vdots \\ \frac{-a_{in}}{b_i - \sum_{j=1}^{n} a_{ij} x_j} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{-a_{i1}}{b_i - \mathbf{a_i}^T \mathbf{x}} \\ \frac{-a_{i2}}{b_i - \mathbf{a_i}^T \mathbf{x}} \\ \vdots \\ \frac{-a_{in}}{b_i - \mathbf{a_i}^T \mathbf{x}} \end{bmatrix} = -\frac{1}{b_i - \mathbf{a_i}^T \mathbf{x}} \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix} = -\frac{1}{b_i - \mathbf{a_i}^T \mathbf{x}} \mathbf{a}_i$$

And on point $(**)$ we did

$$\left(\frac{\partial\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}\mathbf{a_i}\right)}{\partial\mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{i1}\right)}{\partial^2 x_1} & \dots & \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{i1}\right)}{\partial x_1\partial x_n} \\ \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{i2}\right)}{\partial x_2\partial x_1} & \dots & \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{i2}\right)}{\partial x_2\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{in}\right)}{\partial x_n\partial x_1} & \dots & \frac{\partial^2\left(\frac{1}{b_i-\mathbf{a_i}^T\mathbf{x}}a_{in}\right)}{\partial^2 x_n} \end{bmatrix} = \begin{bmatrix} -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{i1}a_{i1} & \dots & -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{i1}a_{in} \\ -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{i2}a_{i1} & \dots & -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{i2}a_{in} \\ \vdots & \ddots & \vdots \\ -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{in}a_{i1} & \dots & -\frac{-1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}a_{in}a_{in} \end{bmatrix}$$

$$= \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}\begin{bmatrix} a_{i1}a_{i1} & \dots & a_{i1}a_{in} \\ a_{i2}a_{i1} & \dots & a_{i2}a_{in} \\ \vdots & \ddots & \vdots \\ a_{in}a_{i1} & \dots & a_{in}a_{in} \end{bmatrix} = \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}\mathbf{a_i}\mathbf{a_i}^T$$

Now that we know the value of the Hessian we have to prove that $\nabla^2 f(\mathbf{x}) \succeq \mathbf{0}$

let $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{v} \neq \mathbf{0}$. Then:

$$\mathbf{v}^T\nabla^2 f(\mathbf{x})\mathbf{v} = \mathbf{v}^T\left(\sum_{i=1}^m \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}\mathbf{a_i}\mathbf{a_i}^T\right)\mathbf{v} = \sum_{i=1}^m \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}\mathbf{v}^T\mathbf{a_i}\mathbf{a_i}^T\mathbf{v} =$$

$$= \sum_{i=1}^m \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}(\mathbf{v}^T\mathbf{a_i})(\mathbf{v}^T\mathbf{a_i})^T = \sum_{i=1}^m \frac{1}{(b_i-\mathbf{a_i}^T\mathbf{x})^2}\|\mathbf{v}^T\mathbf{a_i}\| \geq 0$$

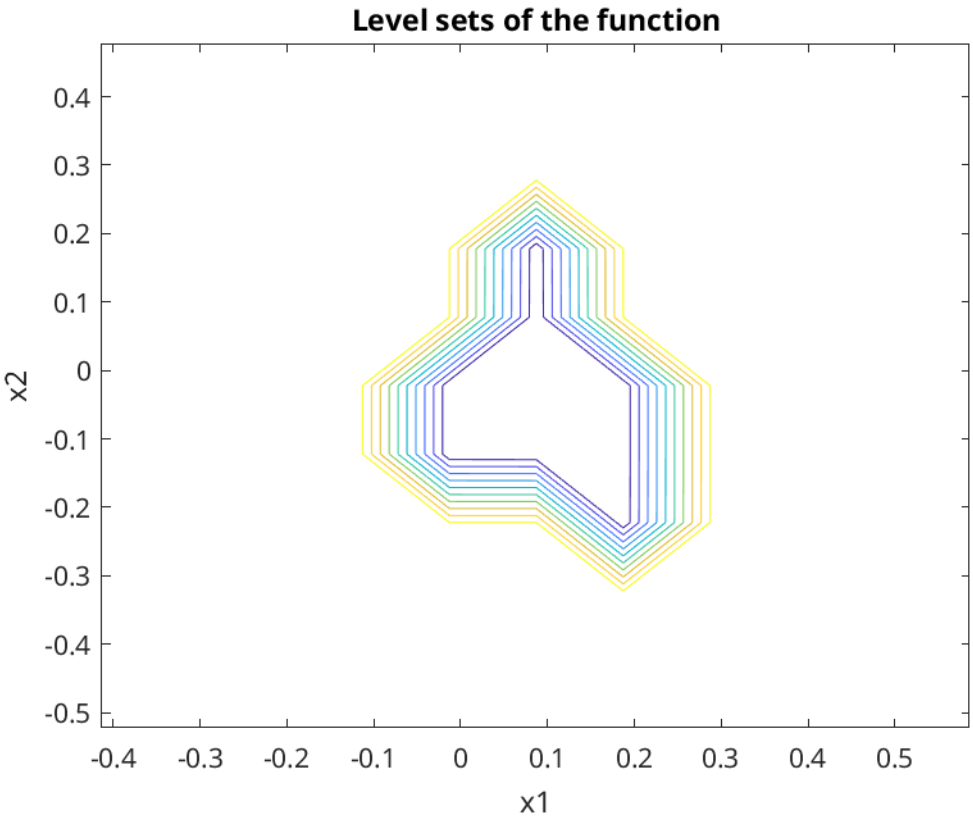
Hence f is convex.

(a) Minimize f 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.

Solution:

he cvx result is cvx_optval = +19.8737

(b) (10) If $n = 2$, then plot f and its level sets in the neighborhood of the optimum point. You can check whether a point x belongs to the domain of f by checking the argument of the logarithm at this point. If a point x belongs to dom$f$, then you can compute the value of f at this point. Otherwise, you can give an arbitrarily large value to f at this point (for example, $f(x) = 10^3$).



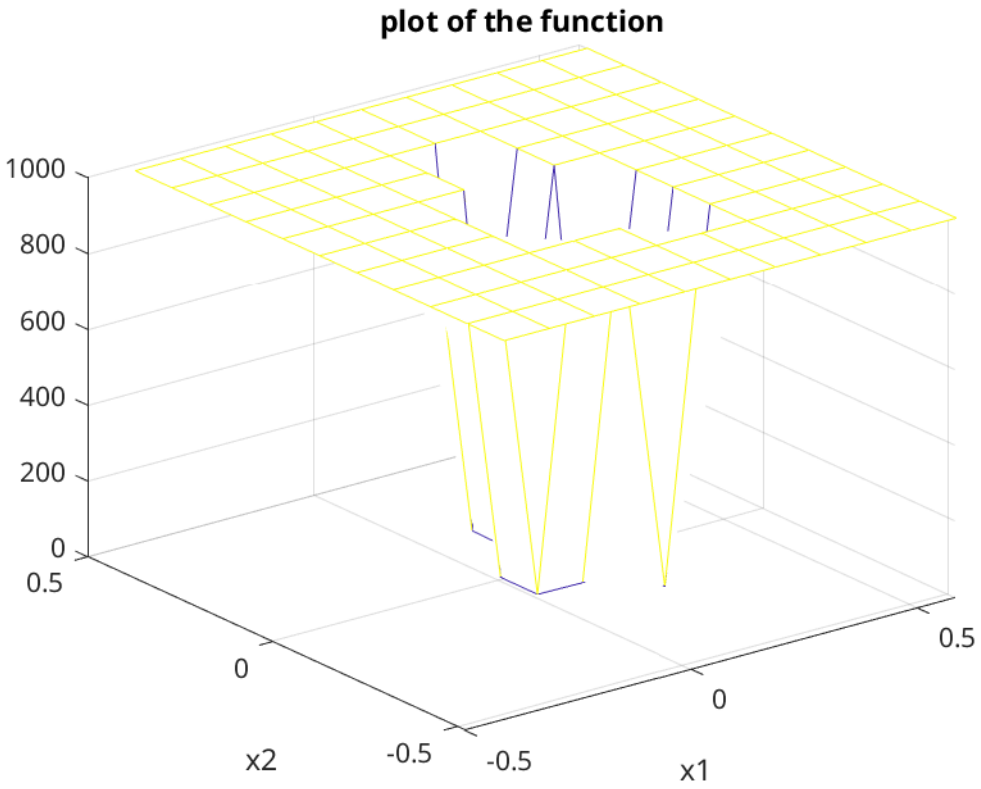


Figure 3: contour and mesh plots for function f

.

.

(c) (30) Assuming that $\mathbf{x} = \mathbf{0}$ is a feasible point, minimize f using the gradient algo- rithm with backtracking line search, starting from $\mathbf{x_0} = \mathbf{0}$. The implementation will have the following main difference from the baseline implementation.

- **In step $(k+1)$, given the vectors $\mathbf{x_k}$ and $\mathbf{\Delta x_k}$, and having set $t = 1$, before starting the backtracking, you should check whether the point $\mathbf{x_k} + \mathbf{tbf\Delta x_k}$ belongs to dom$f$ or not. If it does not belong to dom , then you must put $t := \beta t$ (β is the backtracking parameter) and repeat this process until you find a point that belongs to dom$f$. When you arrive at a point that belongs to dom$f$, then you can proceed to the typical backtracking line search.**

Solution:

For $(n, m) = (2, 20)$ the result of the gradient descent method is

$$\mathbf{p}^*_{gradient} = 19.873663 \text{ in } 5 \text{ iterations}$$

Which is similar to cvx result

(d) (30) Following the analogous procedure, minimize function f using the Newton algorithm.

Solution:

the result of the Newton algorithm is

$$\mathbf{p}^*_{newton} = 19.873673 \text{ in } 2 \text{ iterations}$$

Which is similar to both gradient descent result but it requires less iterations

(e) (10) Plot, using semilogy, quantities $(f_{gradient}(\mathbf{x}_k) - \mathbf{p}_*)$ and $(f_{newton}(\mathbf{x}_k) - \mathbf{p}_*)$, as a function of step k. What do you observe for small and large values of the pair (n, m)?

Solution:

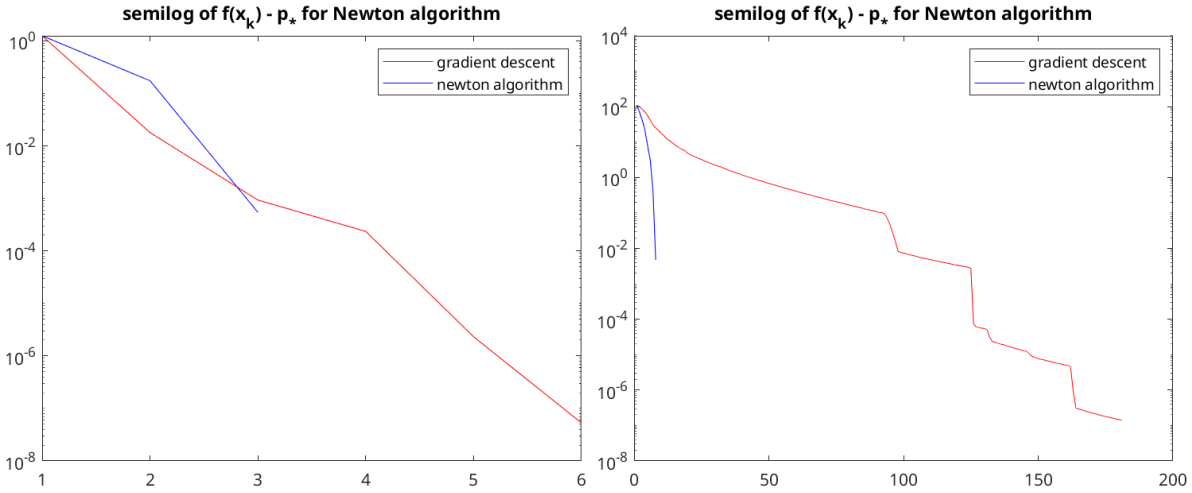


Figure 4: Semilogy for quantities $(f_{gradient}(\mathbf{x}_k) - p_*)$ and $(f_{newton}(\mathbf{x}_k) - p_*)$ for $(n, m) = (2, 20)$ (left) and $(n, m) = (50, 200)$ (right)

We observe that newton algorithm uses less iterations to approach the optimal value compared with the gradient descent , and as the pair $(n.m)$ increases, so does the difference between the performance of the two methods