MAE 5930 - Optimization

Fall 2019

Homework 6

Jared Hansen

Due: 11:59 PM, Thursday November 7, 2019

A-number: `A01439768`

e-mail: `jrdhansen@gmail.com`

Purpose: the problems assigned help develop your ability to

- understand and recognize convexity.
- implement numerical algorithms for linearly constrained optimization.
- solve convex programs using MATLAB's `quadprog`.
- solve convex programs using Newton's Method.

NOTE: please write or type your formulations clearly so that a reader can understand what you are doing. You are welcome to use the equivalent functions in Python.

1. Let Q be an n-by-n symmetrix matrix. Use the definition of a convex function to show under what conditions the quadratic function $f(x) = x^TQx$ is convex.

- Definition of strict convexity: a function $f : D \to \mathbb{R}$ is strictly convex if $\left[\theta f(x) + (1-\theta)f(y)\right] > \left[f(\theta x + 1 - \theta y)\right]$ for all $x, y \in D$ and all $\theta \in (0,1)$
We'll use strict convexity because below we have to divide by $(\theta - \theta^2)$ at one point, which is only > 0 if $\theta \in (0,1)$, whereas the regular definition of convexity has $\theta \in [0,1]$ which let's $(\theta - \theta^2) = 0$ if either $\theta = 0$ or $\theta = 1$.
Also, strict convexity is a stronger condition to have hold for f, so it is desirable if we can show conditions for Q under which $f(x)$ is strictly convex (since it is also "normally" convex under those conditions).

  Also, at the bottom I will show that the edge cases of $\theta = 0$ and $\theta = 1$ still hold for "normal" convexity. This will make a slight difference for conditions on Q under which $f(x)$ is normally convex (f is strictly convex when Q is PD, f is normally convex when Q is PSD.)
- Strict convexity definition for given $f(x)$: $\left[\theta(x^TQx) + (1-\theta)(y^TQy)\right] >? \left[(\theta x + (1-\theta)y)^T(Q)(\theta x + (1-\theta)y)\right]$
- RHS manipulation: $\left[(\theta x + (1-\theta)y)^T(Q)(\theta x + (1-\theta)y)\right] = \left[\left(\theta x + (1-\theta)y\right)^T\left(\theta Qx + (1-\theta)Qy\right)\right]$
$= \left[\left(\theta x^T + (1-\theta)y^T\right)\left(\theta Qx + (1-\theta)Qy\right)\right] = \left[\theta^2 x^TQx + \theta(1-\theta)x^TQy + \theta(1-\theta)y^TQx + (1-\theta)^2 y^TQy\right]$
Now let's expand the last term, as well as use the fact that $x^TQy = y^TQx$ to rewrite as:
RHS $= \left[\theta^2 x^TQx + 2(\theta - \theta^2)x^TQy + y^TQy - 2\theta y^TQy + \theta^2 y^TQy\right]$
- Now, using our current expression for RHS and the LHS from the definition in the first bullet:
$\left[LHS\right] >? \left[RHS\right] \longrightarrow \left[LHS - RHS\right] >? \left[RHS - RHS\right] \longrightarrow \left[LHS - RHS\right] >? \left[0\right]$ which looks like:
$\left[(\theta - \theta^2)(x^TQx) - 2(\theta - \theta^2)(x^TQy) + (y^TQy - y^TQy) + (-\theta y^TQy + 2\theta y^TQy) - \theta^2 y^TQy\right] >? \left[0\right]$
$= \left[(\theta - \theta^2)(x^TQx) - 2(\theta - \theta^2)(x^TQy) + (\theta - \theta^2)(y^TQy)\right] >? \left[0\right]$ now divide both sides by $(\theta - \theta^2)$ which is > 0

  (so long as $\theta \in (0,1)$ as we've specified above in the first bullet.)
- Now we have: $= \left[(x^TQx) - 2(x^TQy) + (y^TQy)\right] >? \left[0\right]$ and since $x^TQy = y^TQx$ can be written as
$= \left[(x^TQx) - (x^TQy) - (y^TQx) + (y^TQy)\right] >? \left[0\right]$ which can be "un-FOILed" (factored) to get
$= \left[(x^T - y^T)(Qx - Qy)\right] >? \left[0\right]$
- Now let's manipulate our factored expression just a bit further to arrive at an answer:
$\left[(x^T - y^T)(Qx - Qy)\right] >? \left[0\right] \longrightarrow \left[(x-y)^T(Q)(x-y)\right] >? \left[0\right]$
- We know that $x, y \in \mathbb{R}^n$, and since the vector space $\mathbb{R}^n$ is closed under addition, the vector $(x - y) \in R^n$ also. Let's say $z = x - y$, (where $z \in \mathbb{R}^n$ due to closure under addition of $\mathbb{R}^n$).
- Now our expression becomes:
$\left[(x-y)^T(Q)(x-y)\right] >? \left[0\right] \longrightarrow \left[(z^T)(Q)(z)\right] >? \left[0\right]$ which looks just like the mathematical definition of a PD

  (positive definite) matrix!
- Definition of a $PD_{n\times n}$ matrix, call it $A_{n\times n}$: $\left[A \text{ is PD iff } (w^T)(A)(w) > 0, \forall w \in \mathbb{R}^n\right]$
- Therefore, the inequality $\left[(z^T)(Q)(z)\right] >? \left[0\right]$ **ONLY HOLDS when Q is a PD matrix**.
Since we arrived at this conclusion through algebraic manipulation of the definition of strict convexity for the function $f(x) = x^TQx$, we can say that
$f(x) = x^TQx$ **is strictly convex (and thus also "normally convex") only when Q is a PD matrix**.

- As promised in the first bullet point above, let's show what happens for the edge cases of $\theta = 0$ and $\theta = 1$ so that we can give conditions on Q under which f is normally convex.
  - CASE: ($\theta = 0$)

    Generically, we can see that $\left[\theta f(x) + (1-\theta)f(y)\right] \geq \left[f(\theta x + 1 - \theta y)\right]$ with $\theta = 0$

    $\longrightarrow \left[(0) + (1)(f(y))\right] \geq \left[f(0 + y)\right] \longrightarrow \left[f(y) \geq f(y)\right]$ will always hold since $\left[f(y) = f(y)\right], \forall y$

    Therefore, we know that the case where $\theta = 0$ will be satisfied for our function $f(y) = y^T Q y, \forall y$.
  - CASE: ($\theta = 1$)

    Generically, we can see that $\left[\theta f(x) + (1-\theta)f(y)\right] \geq \left[f(\theta x + 1 - \theta y)\right]$ with $\theta = 1$

    $\longrightarrow \left[(1)(f(x)) + (0)(f(y))\right] \geq \left[f(x + (0)(y))\right] \longrightarrow \left[f(x) \geq f(x)\right]$ will always hold since $\left[f(x) = f(x)\right], \forall x$

    Therefore, we know that the case where $\theta = 1$ will be satisfied for our function $f(x) = x^T Q x, \forall x$.
- Now that we've shown that both of our edge cases are satisfied, we can confidently re-impose the equality constraint, allowing $\geq$ instead of just $>$ in the definition of convexity (normal VS strict convexity).
- Going to the inequality boxed-in above, let's change it now that we have equality (altering $>$ to be $\geq$):

  Therefore, the inequality $\left[(z^T)(Q)(z)\right] \geq ? \left[0\right]$ **ONLY HOLDS when Q is a PSD matrix**.

  Since we arrived at this conclusion through algebraic manipulation of the definition of strict convexity for the function $f(x) = x^T Q x$ and then examined edge cases of $\theta = 0, 1$ for normal convexity we can say that $f(x) = x^T Q x$ **is normally convex (but not strictly convex) when Q is a PSD matrix**.

- **SUMMARY:**
  - We have shown that when Q is PSD the function $f(x) = x^T Q x$ is "normally" convex.
  - We have shown that when Q is PD the function $f(x) = x^T Q x$ is strictly convex (and thus also "normally" convex since strict convexity is a more restrictive condition which is also met under "normal" convexity).

2. Let S be the set of points contained in the unit square, i.e. $S = \{(x_1, x_2) : -1 \leq x_1 \leq 1 \text{ and } -1 \leq x_2 \leq 1\}$ which reads "S is the set of all points x_1 and x_2 such that $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$."
Use the definition of a convex set to show that S is convex. (You know that the filled-in square is convex because it does not have any holes or indentations. I am asking you to formalize this mathematically.)

- Let's say that we have 2-dimensional vectors $p = [p_1, p_2]^T$ and $w = [w_1, w_2]^T$ where $p, w \in S$.
- This means that: $\begin{bmatrix}-1\\-1\end{bmatrix} \leq \begin{bmatrix}p_1\\p_2\end{bmatrix} \leq \begin{bmatrix}1\\1\end{bmatrix}$ and $\begin{bmatrix}-1\\-1\end{bmatrix} \leq \begin{bmatrix}w_1\\w_2\end{bmatrix} \leq \begin{bmatrix}1\\1\end{bmatrix}$
- Now let's take some steps to make the two above expressions (the inequalities that bound the values of elements in p and w) look like the definition for a convex set in order to show that S is convex.
- First, multiply each term in $\big[-1 \leq p \leq 1\big]$ by θ. Note that doing this won't change the direction of the inequality since θ is non-negative by definition. We'll have: $\big[(-1)(\theta) \leq (p)(\theta) \leq (1)(\theta)\big] = \big[-\theta \leq \theta p \leq \theta\big]$
- Next, multiply each term in $\big[-1 \leq w \leq 1\big]$ by $(1-\theta)$. Note that doing this won't change the direction of the inequality since $(1-\theta)$ is non-negative by definition (since $\theta \in [0,1]$).
We'll have: $\big[(-1)(1-\theta) \leq (w)(1-\theta) \leq (1)(1-\theta)\big] = \big[(\theta - 1) \leq (1-\theta)w \leq (1-\theta)\big]$
- Now let's combine these two inequalities into one by element-wise-adding them together, giving:
$\Bigg[\Big((-\theta) + (\theta - 1)\Big) \leq \Big((\theta p) + ((1-\theta)w)\Big) \leq \Big((\theta) + (1-\theta)\Big)\Bigg] = \Bigg[\Big(-1\Big) \leq \Big(\theta p + (1-\theta)w\Big) \leq \Big(1\Big)\Bigg]$
where -1 and 1 are vectors (2-dim vectors to match dimensions of p and w.)
- Next let's examine what the definition of a convex set is: $\Big[\theta x + (1-\theta)y \in C, \forall x, y \in C \text{ and } \forall \theta \in [0,1]\Big]$
Translating this generic definition into a specific definition of what makes S a convex set:
  (a) First, let's switch C for S, x for p, and y for w. We will leave the θ symbol as it is.
  (b) Substituting these new symbols into the generic definition, we have the condition that S is a convex set if $\Bigg[\Big(\theta p + (1-\theta) \in S\Big)\forall p, w \in S \text{ and } \forall \theta \in [0,1]\Bigg]$
  (c) Next, what does it mean for p or w to be $\in S$? This falls back on the definition given in the prompt: $p \in S$ if $\begin{bmatrix}-1\\-1\end{bmatrix} \leq \begin{bmatrix}p_1\\p_2\end{bmatrix} \leq \begin{bmatrix}1\\1\end{bmatrix}$, and $w \in S$ if $\begin{bmatrix}-1\\-1\end{bmatrix} \leq \begin{bmatrix}w_1\\w_2\end{bmatrix} \leq \begin{bmatrix}1\\1\end{bmatrix}$
- Finally, let's look at the combined inequality $\begin{bmatrix}-1\\-1\end{bmatrix} \leq \Big(\theta p + (1-\theta)w\Big) \leq \begin{bmatrix}1\\1\end{bmatrix}$ we have and see if it satisfies the definition of S being a convex set:
  - First, we can see that this has the necessary form of $\Big(\theta p + (1-\theta)\Big)$, and since this quantity (a 2-dim vector) is bounded by the vectors $[-1,-1]^T$ and $[1,1]^T$ this confirms that this quantity (a 2-dim vector) is indeed $\in S$ as it must be in order for the set S to be convex.
  - Next, does the above condition hold $\forall p, w \in S$? **Yes!** We know this because we have constructed p and w such that $p, w \in S$ always. See near the top (second bullet): we specify the bounds on the elements of p and w such that they are both vectors $\in S$. Therefore, we can confirm that $\Big(\theta p + (1-\theta)w\Big) \in S$, $\forall p, w \in S$.
  - Finally, does $\Big(\theta p + (1-\theta)\Big) \in S$ hold $\forall \theta \in [0,1]$? **Again, the answer is yes!** We know this because when manipulating inequalities above we simply treated θ as a non-negative constant, e.g. $\theta \in [0, \infty)$. When we did this, all of the inequalities still held and were valid. This was even more general than specifying that we must have $\theta \in [0,1]$. Therefore, all of the above also hold $\forall \theta \in [0,1]$ since this is a subset of holding $\forall \theta \in [0,\infty)$.
- **In summary:**
Using the definition of a convex set we have shown that $\Big(\theta p + (1-\theta)w\Big) \in S$, $\forall p, w \in S$ and $\forall \theta \in [0,1]$.
Therefore, we have shown that S is indeed a convex set.

3. Let D be the set of points contained in an annulus (or donut) with inner radius R_i and outer radius R_o, i.e. $D = \{(x_1, x_2) : (R_i)^2 \leq (x_1)^2 + (x_2)^2 \leq (R_o)^2\}$.
Use the definition of a convex set to show that D is non-convex. (Again, you can draw a picture and see that the set is non-convex because it has a hole. I am asking you to formalize this mathematically.)

- We just need to find a single counterexample, showing that for two vectors (let's call them p and w) and some value of θ that we can have $p, w \in D$ and $\theta \in [0, 1]$ but still get that $\theta p + (1-\theta)w \notin D$ (since the "for all" constraints won't hold with a single counterexample).
- Therefore, if we can find one combination of "legal" values for p, w, θ such that $\theta p + (1-\theta)w \notin D$ we will have shown that D is not a convex set.
- Let $p = (0, R_i)$. Showing that $p \in D$: $\left[(R_i)^2 \leq ?(0)^2 + (R_i)^2 \leq ?(R_o)^2\right] \longrightarrow \left[(R_i)^2 \leq (R_i)^2 \leq (R_o)^2\right]$ holds, therefore $p \in D$.
- Let $w = (0, -R_i)$. Showing that $w \in D$: $\left[(R_i)^2 \leq ?(0)^2 + (-R_i)^2 \leq ?(R_o)^2\right] \longrightarrow \left[(R_i)^2 \leq (R_i)^2 \leq (R_o)^2\right]$ holds, therefore $w \in D$.
- Now find $\theta \longrightarrow \left[\theta p + (1-\theta)w\right] = \theta \begin{bmatrix} 0 \\ R_i \end{bmatrix} + (1-\theta) \begin{bmatrix} 0 \\ -R_i \end{bmatrix} = \begin{bmatrix} 0 \\ \theta R_i + (1-\theta)(-R_i) \end{bmatrix} = \begin{bmatrix} 0 \\ \theta R_i - R_i + \theta R_i \end{bmatrix}$
$= \begin{bmatrix} 0 \\ 2\theta R_i - R_i \end{bmatrix}$
- Is this new vector $\in D$? $\longrightarrow \left[(0)^2 + (2\theta R_i - R_i)^2\right] = \left[4\theta^2(R_i)^2 - 4\theta(R_i)^2 + (R_i)^2\right]$
Is $\left[4\theta^2(R_i)^2 - 4\theta(R_i)^2 + (R_i)^2 \geq (R_i)^2\right]$ as it must be in order to be $\in D$?
- $\left[4\theta^2(R_i)^2 - 4\theta(R_i)^2 + (R_i)^2 \geq ?(R_i)^2\right] \longrightarrow \left[4\theta^2(R_i)^2 - 4\theta(R_i)^2 \geq ?0\right] \longrightarrow \left[(4\theta(R_i)^2)(\theta - 1) \geq ?0\right]$
- Let $\theta = \frac{1}{2} \longrightarrow \left[\left(\frac{4}{1} \cdot \frac{1}{2}\right)\left(R_i\right)^2\left(\frac{1}{2} - \frac{2}{2}\right) \geq ?0\right] \longrightarrow \left[(-1)(R_i)^2 \geq ?0\right]$
Since $R_i > 0$ must hold (otherwise we don't have a donut, we would just have a regular circle) it must also be that $(R_i)^2 > 0 \implies \left[-(R_i)^2 \ngeq 0\right]$ since a positive value $\big((R_i)^2\big)$ multipiled by -1 is a negative value.
- We have shown that for $\left(\theta = \frac{1}{2}\right) \in [0, 1]$, and for $p, w \in D$ where $p = (0, R_i)$ and $w = (0, -R_i)$
that $\left[\theta p + (1-\theta)w \notin D\right] \implies$
$\left[\theta p + (1-\theta)w \in D\right]$ **does not hold for all** $p, w \in D$ and **does not hold for all** $\theta \in [0, 1]$.
Therefore, D is not a convex set.

4. For each of the following functions, use the Hessian test to determine if the function is convex.

(a) $\left[f : \mathbb{R} \to \mathbb{R}, f(x) = -8x^2\right]$

- The question we want to answer: is $\left[\nabla^2 f(x) \text{ PSD } \forall x \in D \text{ (where } D = \mathbb{R} \text{ in this problem)}\right]$?
- $\left[\nabla f(x) = -(8)(2)x\right] \longrightarrow \left[\nabla^2 f(x) = -16\right]$
- Is $\nabla^2 f(x) = -16$ PSD ? $\longrightarrow \left[-16 \geq ?0, \forall x \in \mathbb{R}\right] \longrightarrow$ No! $\forall x \in \mathbb{R}$ we know that -16 is always -16 which is $\not\geq 0 \implies \left[\nabla^2 f(x) \text{ is not PSD } \forall x \in \mathbb{R}\right] \implies$ $\underline{f(x) = -8x^2 \textbf{ is not a convex function.}}$

(b) $\left[f : \mathbb{R}^3 \to \mathbb{R}, f(x) = 4{x_1}^2 + 3{x_2}^2 + 5{x_3}^2 + 6x_1x_2 + x_1x_3 - 3x_1 - 2x_2 + 15\right]$

- The question we want to answer: is $\left[\nabla^2 f(x) \text{ PSD } \forall x \in D \text{ (where } D = \mathbb{R}^3 \text{ in this problem)}\right]$?
- $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 8x_1 + 6x_2 + x_3 - 3 \\ 6x_1 + 6x_2 + 0x_3 - 2 \\ x_1 + 0x_2 + 10x_3 + 0 \end{bmatrix} \longrightarrow \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} = \begin{bmatrix} 8 & 6 & 1 \\ 6 & 6 & 0 \\ 1 & 0 & 10 \end{bmatrix}$
- Is the matrix $\nabla^2 f(x)$ PSD? We will check this by computing the eigenvalues of the matrix. If all of the eigenvalues are $\geq 0 \implies \nabla^2 f(x)$ is a PSD matrix $\implies f(x)$ is convex.
Using WolframAlpha for computation, the eigenvalues for the Hessian are $13.2631, 9.8657, 0.8712$, all of which are $\geq 0 \implies$ that **the function** $f(x) = 4{x_1}^2 + 3{x_2}^2 + 5{x_3}^2 + 6x_1x_2 + x_1x_3 - 3x_1 - 2x_2 + 15$ **is a convex function** (since the evals are all > 0 we can be more precise and say that this function f is strictly convex.)

(c) $\left[f : \mathbb{R}^2 \to \mathbb{R}, f(x) = -2x_1x_2 + {x_1}^2 + {x_2}^2\right]$

- The question we want to answer: is $\left[\nabla^2 f(x) \text{ PSD } \forall x \in D \text{ (where } D = \mathbb{R}^2 \text{ in this problem)}\right]$?
- $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 2x_2 \\ -2x_1 + 2x_2 \end{bmatrix} \longrightarrow \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$
- Is the matrix $\nabla^2 f(x)$ PSD? We will check this by computing the eigenvalues of the matrix. If all of the eigenvalues are $\geq 0 \implies \nabla^2 f(x)$ is a PSD matrix $\implies f(x)$ is convex.
Using WolframAlpha for computation, the eigenvalues for the Hessian are 4 and 0, both of which are $\geq 0 \implies$ that **the function** $f(x) = -2x_1x_2 + {x_1}^2 + {x_2}^2$ **is a convex function**.

(d) $\left[f:\mathbb{R}^n \to \mathbb{R}, f(x) = \alpha_1 x_1 + \ldots + \alpha_n x_n\right]$

- The question we want to answer: is $\left[\nabla^2 f(x) \text{ PSD } \forall x \in D \text{ (where } D = \mathbb{R}^n \text{ in this problem)}\right]$?

- $$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \longrightarrow \nabla^2 f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} = \mathbf{0}_{n\times n}$$

- Is the matrix $\nabla^2 f(x)$ PSD? Normally we would check the eigenvalues (which we could still do if desired), and if all eigenvalues are ≥ 0 then the Hessian is PSD.
  In this case it's even simpler than that: we know that $\left[(x^T)(\mathbf{0}_{n\times n}) = \mathbf{0}_{1\times n}, \forall x \in \mathbb{R}^n\right]$ and $\left[(\mathbf{0}_{n\times n})(x) = \mathbf{0}_{n\times 1}, \forall x \in \mathbb{R}^n\right]$. Therefore, $\left[(x^T)(\nabla^2 f)(x) = (x^T)(\mathbf{0}_{n\times n})(x) = 0, \forall x \in \mathbb{R}^n\right] \implies \left[(x^T)(\nabla^2 f)(x) = 0 \geq 0, \forall x \in \mathbb{R}^n\right] \implies \left[\nabla^2 f \text{ is PSD}\right] \implies$ [**the function f is a convex function**]

(e) $\left[f:\left(0, \frac{\pi}{2}\right) \to \mathbb{R}, f(x) = sin(x)\right]$

- The question we want to answer: is $\left[\nabla^2 f(x) \text{ PSD } \forall x \in D \text{ (where } D = \left(0, \frac{\pi}{2}\right) \text{ in this problem)}\right]$?

- $\left[\nabla f = cos(x)\right] \longrightarrow \left[\nabla^2 f = -sin(x)\right]$

- Is $\nabla^2 f(x) = -sin(x)$ PSD ? $\to \left[(-sin(x)) \geq ?0, \forall x \in (0, \pi/2)\right]$
  No! Let $x = \frac{\pi}{4}$ giving: $\left[\left(-sin\left(\frac{\pi}{4}\right)\right) = -\left(\frac{\sqrt{(2)}}{2}\right) \approx -0.7071 \ngeq 0\right] \implies$
  $\left[\nabla^2 f(x) \text{ is not PSD } \forall x \in (0, \pi/2)\right] \implies$ **$f(x) = sin(x)$ on $D = (0, \pi/2)$ is not a convex function.**

5. Program Newton's Method with Equality Constraints as described by Boyd on page 528. (Note that Boyd considers gradient vectors to be column vectors instead of row vectors.)

   (a) Inputs to the function are the objective, gradient, Hessian, A matrix, and feasible starting point.

   - See my code at the bottom of the assignment (directly below this problem).

   (b) Within the function set $\epsilon = 1e-6$.

   - See my code at the bottom of the assignment (directly below this problem).

   (c) Solve the same problem using MATLAB's `quadprog` and your code. You should get the same answer.

$$min||x|| \text{ subject to } Ax = b \text{ where } A = \begin{bmatrix} 4 & 4 & 4 & 9 & 10 & 10 & 3 & 6 & 0 & 1 \\ 7 & 5 & 6 & 9 & 5 & 7 & 1 & 1 & 5 & 8 \\ 0 & 4 & 1 & 1 & 7 & 3 & 0 & 6 & 7 & 4 \\ 3 & 7 & 2 & 0 & 3 & 8 & 7 & 7 & 5 & 2 \\ 1 & 2 & 8 & 2 & 7 & 1 & 2 & 1 & 9 & 9 \\ 1 & 9 & 10 & 9 & 8 & 4 & 3 & 4 & 6 & 3 \\ 2 & 0 & 3 & 1 & 0 & 9 & 5 & 7 & 9 & 8 \\ 3 & 7 & 7 & 4 & 8 & 3 & 1 & 4 & 1 & 7 \end{bmatrix} \text{ and } b = \begin{bmatrix} 9 \\ 6 \\ 8 \\ 3 \\ 3 \\ 9 \\ 4 \\ 10 \end{bmatrix}$$

```
%==========================================================================
%== PROBLEM 5
%==========================================================================
% Solve the problem using MATLAB's quadprog.

% Define the objective function (here we'll do |x|^2 since it's easier
% to define, and is the same as minimizing |x|)
% multiply by 2 due to Matlab's formulation with the 1/2 in front
H = 2 * eye(10);
f = zeros(10,1)';
% Matrix Aeq and vector beq defining equality constraints
Aeq =[ 4 4 4  9 10 10 3 6 0 1;
       7 5 6  9 5  7  1 1 5 8;
       0 4 1  1 7  3  0 6 7 4;
       3 7 2  0 3  8  7 7 5 2;
       1 2 8  2 7  1  2 1 9 9;
       1 9 10 9 8  4  3 4 6 3;
       2 0 3  1 0  9  5 7 9 8;
       3 7 7  4 8  3  1 4 1 7];
beq = [9 6 8 3 3 9 4 10]';

soln = quadprog(H,f,[],[],Aeq,beq);
disp("Matlab solution using QUADPROG")
soln
```

Figure 1: Here is my MATLAB code that uses `quadprog` to solve the problem.

```
Matlab solution using QUADPROG

soln =

   -0.3700
    0.3496
    0.0643
    0.3944
    0.0915
    0.0401
   -0.7331
    0.9552
   -0.2939
    0.4270
```

Figure 2: Here is my MATLAB solution – uses `quadprog` to solve the problem.

```
---------------------------------
---- Newton's Method solution ---
---------------------------------
x* Newton's mtd    :
[[-0.37004167]
 [ 0.34958219]
 [ 0.06427502]
 [ 0.3943607 ]
 [ 0.09154028]
 [ 0.04014191]
 [-0.73314387]
 [ 0.95517679]
 [-0.29391361]
 [ 0.42704059]]
f(x*)              :  2.147392691583059
iters to converge  :  1
```

Figure 3: Here is my Python solution. Implements Newton's Method with equality constraints "by hand."

```python
'''
Assignment:     hw6, MAE 5930 (Optimization)
File name:      optzn_hw6_JHansen.py
Author:         Jared Hansen
Date created:   11/06/2019
Python version: 3.7.3

DESCRIPTION:
    This Python script is used for answering Problem 5 from hw6 in
    MAE 5930 (Optimization).
'''



#-----------------------------------------------------------------------------
#--- IMPORT STATEMENTS -------------------------------------------------------
#-----------------------------------------------------------------------------
import numpy as np
import numpy.linalg as npla
import random
# Set a seed to make results reproducible
random.seed(1776)


#-----------------------------------------------------------------------------
#--- FUNCTION IMPLEMENTATIONS ------------------------------------------------
#-----------------------------------------------------------------------------
def obj_fctn(x):
    """ Takes the point x (R^10 vec) and returns the value of the squared
    (objective) function (this is easier/nicer to minimize).
    """
    return(np.asscalar(np.dot(x.T,x)))

def obj_grad(x):
    """ Takes the point x (R^10 vec) and returns the gradient of the objective
    function. Here this is just the vector scaled by 2.
    """
    return(2*x)

def obj_hess():
    """ Doesn't need to take any arguments: for the objective function ||x||^2
    the Hessian will always be the identity matrix scaled by 2
    with dimensions numRows=numCols=dim(x)
    """
    return( 2 * np.identity(len(x0)) )

def backtrack(f, grd, x, v):
    """
    This function implements the backtracking algorithm.

    Parameters
    ----------
    f     : a mathematical function (the objective function)
    grd   : a mathematical function (the gradient of f)
    x     : a point in the domain of f (either a scalar or a vector/list/array)
    v     : descent direction (a floating point number)
    t     : step size parameter
    alpha : adjustadble floating point number (between 0 and 0.5)
    beta  : float, btwn 0-1, granularity of search (large=fine, small=coarse)

    Returns
    -------
    t : descent direction (floating point number between 0 and 1)
    """
    # Define the starting value of t, and values of alpha and beta
    t = 1.0
    alpha = 0.2
```

```python
    beta = 0.5
    # Implement the condition found in the algorithm defintion
    while(f(x + t*v) >
          f(x) + alpha*t*np.asscalar(np.dot(grd(x).T, v)) ):
        t *= beta
        print("t = ", t)
    return(t)

def newtons_method(f, grd, hessian, x0):
    """
    This function implements Newton's method. Set epsiolon = 1e-6.

    Parameters
    ----------
    f       : a mathematical function (the objective function)
    grd     : a mathematical function (the gradient of f)
    hessian : a matrix (numpy matrix; the hessian of f)
    x0      : a feasible starting pt (scalar or an array/list).

    Returns
    -------
    newtons_output: a tuple containing (final_x, f_final, iters)
    final_x : the point in the domain of f that minimizes f (to eta tolerance)
    f_final : the function f evaluated at final_x
    iters   : the number of iterations that it took to achieve the minimum
    """
    # Initialize a variable to count the number of iterations
    iters = 1
    # Declare a variable for the max number of iterations
    MAX_ITERS = 20000
    # Define our tolerance, the constant EPSILON
    EPSILON = 1e-6
    # Initialize the point we're going to update, x, as the starting point x0
    x = x0
    # Calculate Newton's decrement value
    lmb_sq = np.asscalar(np.dot(np.dot(grd(x).T , npla.pinv(hessian())) ,
                         grd(x)))
    # Stay in this while loop until the have sufficiently small lmb_sq
    while((lmb_sq/2.0) > EPSILON):
        # Compute delta_x and Newton's decrement
        dlt_x = np.dot(-npla.pinv(A) , (np.dot(A,x) - b))
        # Line search for t (descent direction)
        t = backtrack(f, grd, x, dlt_x)
        # Update our "more minimizing" x
        x = np.array(x + (t * dlt_x))
        # Print statements to see how the algorithm is doing
        print("f(x)  : ", f(x))
        print("iters : ", iters)
        print()
        # Update lmb_sq
        lmb_sq = np.asscalar(np.dot(np.dot(grd(x).T , npla.pinv(hessian())) ,
                         grd(x)))
        # Update number of iterations
        iters += 1
        # Set up the function to quit after reaching MAX_ITERS
        if(iters > MAX_ITERS):
            print("Reached MAX_ITERS (", MAX_ITERS, ") without converging.")
            break
    # After sufficiently approach EPSILON or reaching MAX_ITERS return a tuple
    # containing the final_x, f_final, and iters
    newt_output = (x, f(x), iters)
    return(newt_output)


#------------------------------------------------------------------------------
#--- PROCEDURAL CODE -----------------------------------------------------------
#------------------------------------------------------------------------------
```

```python
# Define matrix A, vector b, and starting point x0
A = np.matrix([[ 4,  4,  4,  9, 10, 10,  3,  6,  0,  1],
               [ 7,  5,  6,  9,  5,  7,  1,  1,  5,  8],
               [ 0,  4,  1,  1,  7,  3,  0,  6,  7,  4],
               [ 3,  7,  2,  0,  3,  8,  7,  7,  5,  2],
               [ 1,  2,  8,  2,  7,  1,  2,  1,  9,  9],
               [ 1,  9, 10,  9,  8,  4,  3,  4,  6,  3],
               [ 2,  0,  3,  1,  0,  9,  5,  7,  9,  8],
               [ 3,  7,  7,  4,  8,  3,  1,  4,  1,  7]])
b = np.array([9,6,8,3,3,9,4,10]).reshape(8,1)
# A point x is feasible (per the equality constraint) if [Ax=b].
# Therefore we can simply do [x = A^(-1)*b] to find x, where A^(-1) will be
# the pseudo-inverse of A since A may not be invertible.
x0 = np.dot(npla.pinv(A), b)
# Find the solution using Newton's Method with equality constraint
newt_output = newtons_method(obj_fctn, obj_grad, obj_hess, x0)
# Output the results to the console
print("\n---------------------------------")
print("---- Newton's Method solution ---")
print("---------------------------------")
print("x* Newton's mtd   : ")
print( newt_output[0])
print("f(x*)             : ", newt_output[1])
print("iters to converge : ", newt_output[2])
```