. . .

CME307 HW4

Jacob Perricone
Stanford University
jacobp2@stanford.edu

Problem 1

Recall the (local) second-order (SO) and scaled second-order (SSO) Lipschitz conditions (LC):

$$\text{SOLC} : ||\nabla f(\mathbf{x}+\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\mathbf{d}|| \leq \beta||\mathbf{d}||^2, \text{where } ||\mathbf{d}|| \leq .5$$

and

$$\text{SSOLC} : ||X(\nabla f(\mathbf{x}+\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\mathbf{d})|| \leq \beta|\mathbf{d}^2\nabla^2 f(\mathbf{x})d|, \text{where } ||X^{-1}\mathbf{d}|| \leq .5$$

Find parameter β values (or upper bounds) of (SOLC) and (SSOLC) for each of the following scalar functions:

(a) $f(x) = \frac{1}{3}x^3 + x, x > 0$

We have

$$\nabla f(x) = x^2 + 1$$
$$\nabla^2 f(x) = 2x$$

Then the SOLC condition is

$$||(x+d)^2 - x^2 - 2xd|| \leq \beta||d||^2$$
$$||d^2|| \leq \beta||d||^2$$
$$1 \leq \beta$$

Noticing that in the scalar case, the norm can be replaced by an absolute value, so $\beta = 1$ holds.
For the SSOLC condition we have

$$||xd^2|| \leq 2\beta||2xd^2||$$
$$\frac{1}{2} \leq \beta$$

(b) $f(x) = \log(x), x > 0.$

We have

$$\nabla f(x) = \frac{1}{x}$$
$$\nabla^2 f(x) = -\frac{1}{x^2}$$

SOLC condition

$$||\frac{1}{x+d} - \frac{1}{x} + \frac{1}{x^2}|| \leq \beta||d||$$
$$||\frac{-d}{x(x+d)} + \frac{1}{x^2}|| \leq \beta||d||$$
$$||\frac{d^2}{x^2(x+d)}|| \leq \beta||d||$$
$$||\frac{1}{x^2(x+d)}|| \leq \beta$$

Thus there is no upperbound since we can set x arbitrarily close to zero, causing β to be aribtrarily large
In the SSOLC condition we have

$$||\frac{xd^2}{x^2(x+d)}|| \leq \beta||\frac{d^2}{x^2}||$$
$$||\frac{x}{(x+d)}|| \leq \beta$$
$$||\frac{1}{1+x^{-1}d}|| \leq \beta$$
$$\leq \frac{1}{1-||x^{-1}d||} \leq \beta$$
$$\lim_{||x^{-1}d|| \to 0} \frac{1}{1-||x^{-1}d||} = 1$$
$$\lim_{||x^{-1}d|| \to \frac{1}{2}} \frac{1}{1-||x^{-1}d||} = 2$$

Thus:

$$2 \leq \beta$$

(c) $f(x) = \log(1 + e^{-x}), x > 0$

We have

$$\nabla f(x) = \frac{-e^{-x}}{1+e^{-x}}$$
$$\nabla^2 f(x) = \frac{e^{-x}}{1+e^{-x}} + \frac{e^{-2x}}{(1+e^{-x})^2}$$
$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

Let $\sigma(x) = \frac{e^{-x}}{1+e^{-x}}$ Then

$$\nabla f(x) = -\sigma(x)$$
$$\nabla^2 f(x) = \sigma(x)(1-\sigma(x))$$

For the SOLC condition we have

$$||\frac{\nabla f(x+d)}{d^2} - \frac{\nabla f(x)}{d^2} - \frac{\nabla^2 f(x)}{d}|| \leq \beta$$

Now by the mean value theorem we know that $\exists \psi \in [x, x+d]$ such that

$$\nabla^2 f(\psi) = \frac{\nabla f(x+d) - \nabla f(x)}{d}$$

THus we have

$$||\frac{\nabla^2 f(\psi) - \nabla^2 f(x)}{d}|| \leq \beta$$

Applying the MVT again we can write for some $\mu \in [x, \psi]$,

$$\nabla^3 f(\mu) \approx \frac{\nabla^2 f(\psi) - \nabla^2 f(x)}{d}$$

There fore the original inequality reduces too:

$$||\nabla^3 f(\mu)|| \leq \beta$$

We have that the third derivative is equal to

$$\nabla^3 f(x) = \frac{e^x(1-e^x)}{(1+e^x)^3}$$

Plotting the third derivative in matlab, one can see that it is periodic, with a range of $[\frac{-1}{6\sqrt{3}}, \frac{1}{6\sqrt{3}}]$. Thus, we can establish the bound $\frac{1}{6\sqrt{3}} \leq \beta$.

In the SSOLC case we have

$$||\frac{-x}{1+e^{-x-d}} + \frac{x}{1+e^{-x}} - \frac{dx}{(1+e^{-x})^2} \leq \beta|\frac{-d}{x^2}|$$
$$||\frac{x^3 e^{-x}(e^{-d}-1)(1+e^{-x}) - dx^3(1+e^{-x-d})}{d^2(1+e^{-x-d})(1+e^{-x})^2}|| \leq \beta$$
$$\lim_{x\to\infty, d\neq 0} |\frac{x^3 e^{-x}(e^{-d}-1)(1+e^{-x}) - dx^3(1+e^{-x-d})}{d^2(1+e^{-x-d})(1+e^{-x})^2}| = \infty$$
$$\lim_{d\to 0} |\frac{x^3 e^{-x}(e^{-d}-1)(1+e^{-x}) - dx^3(1+e^{-x-d})}{d^2(1+e^{-x-d})(1+e^{-x})^2}| = \infty$$

Thus there exists no bound for β in the SSOLC.

Problem 2

In Logistic Regression, we like to determine x_0 and $\mathbf{x}$ to maximize

$$\left(\prod_{i,c_i=1} \frac{1}{1+\exp(-\mathbf{a}_i^T\mathbf{x} - x_0)}\right)\left(\prod_{i,c_i=-1} \frac{1}{1+\exp(\mathbf{a}_i^T\mathbf{x} + x_0)}\right).$$

which is equivalent to maximize the log-likelihood probability

$$-\sum_{i,c_i=1} \log\left(1+\exp(-\mathbf{a}_i^T\mathbf{x} - x_0)\right) - \sum_{i,c_i=-1} \log\left(1+\exp(\mathbf{a}_i^T\mathbf{x} + x_0)\right).$$

Or to minimize the log-logistic-loss

$$\sum_{i,c_i=1} \log\left(1+\exp(-\mathbf{a}_i^T\mathbf{x}-x_0)\right) + \sum_{i,c_i=-1} \log\left(1+\exp(\mathbf{a}_i^T\mathbf{x}+x_0)\right).$$

(a) Write down the Hessian matrix funtion of $\mathbf{x}, x_0$

Let $f(x, x_0)$ be the log-logistic loss function.

$$\nabla f(\mathbf{x}, x_0)_{x_j} = \sum_{i,c_i=1} \frac{-a_{ij}\exp[-\mathbf{a}_i^T\mathbf{x}-x_0]}{1+\exp[-\mathbf{a}_i^T\mathbf{x}-x_0]} + \sum_{i,c_i=-1} \frac{a_{ij}\exp[\mathbf{a}_i^T\mathbf{x}+x_0]}{1+\exp[\mathbf{a}_i^T\mathbf{x}+x_0]}\ \forall j$$

$$\nabla f(\mathbf{x}, x_0)_{x_0} = \sum_{i,c_i=1} \frac{-\exp[-\mathbf{a}_i^T\mathbf{x}-x_0]}{1+\exp[-\mathbf{a}_i^T\mathbf{x}-x_0]} + \sum_{i,c_i=-1} \frac{\exp[\mathbf{a}_i^T\mathbf{x}+x_0]}{1+\exp[\mathbf{a}_i^T\mathbf{x}+x_0]}$$

Let us define $\mathbf{z} = -\mathbf{a}_i^T\mathbf{x} - x_0$ and $\bar{\mathbf{z}} = \mathbf{a}_i^T\mathbf{x} + x_0$. We have that

$$\begin{aligned}
\nabla_{x_j,x_k} &= \sum_{i,c_i=1} a_{ij}a_{ik}[\frac{\exp[\mathbf{z}]}{1+\exp[\mathbf{z}]} - \frac{\exp[2\mathbf{z}]}{(1+\exp[\mathbf{z}])^2}] + \sum_{i,c_i=-1} a_{ij}a_{ik}[\frac{\exp[\bar{\mathbf{z}}]}{1+\exp[\bar{\mathbf{z}}]} - \frac{\exp[2\bar{\mathbf{z}}]}{(1+\exp[\bar{\mathbf{z}}])^2}] \\
&= \sum_{i,c_i=1} a_{ij}a_{ik}\frac{\exp[\mathbf{z}]}{(1+\exp[\mathbf{z}])^2} + \sum_{i,c_i=-1} a_{ij}a_{ik}\frac{\exp[\bar{\mathbf{z}}]}{(1+\exp[\bar{\mathbf{z}}])^2} \\
\nabla_{x_j,x_0} &= \sum_{i,c_i=1} a_{ij}\frac{\exp[\mathbf{z}]}{(1+\exp[\mathbf{z}])^2} + \sum_{i,c_i=-1} a_{ij}\frac{\exp[\bar{\mathbf{z}}]}{(1+\exp[\bar{\mathbf{z}}])^2} \\
\nabla_{x_0,x_0} &= \sum_{i,c_i=1} \frac{\exp[\mathbf{z}]}{(1+\exp[\mathbf{z}])^2} + \sum_{i,c_i=-1} \frac{\exp[\bar{\mathbf{z}}]}{(1+\exp[\bar{\mathbf{z}}])^2}
\end{aligned}$$

Thus the $i, j, (i, j) \in \{0, \ldots, n\}$ element of the hessian matrix is given by the equations above

- (b) (Computation Team Work) Apply any Quasi-Newton (e.g., slide 18 of Lecture 13 or L &Y Chapter 10) and Newton methods to solve the problem using the data in HW2 for SVM (may or may not with regulation), randomly generate data sets, and/or benchmark data sets you can find. Compare the two methods with each other and with the previous methods used in HW3.

Problem Three

Consider the LP problem

$$\begin{aligned}
&\min_x f(x) = x_1 + x_2 \\
&\text{Such that } : x_1 + x_2 + x_3 = 1 \\
&\quad (x_1, x_2, x_3) \geq 0
\end{aligned}$$

(a) What is the analytic center of the feasible region with the logarithmic barrier function

The analytic center is found by minimizing

$$\min_{x_i} -\log(x_1) - \log(x_2) - \log(1 - x_1 - x_2)$$

Taking the derivative with respect to x_1, x_2 we have

$$2x_1 = 1 - x_2$$
$$2x_2 = 1 - x_1$$

where the substitution $x_3 = 1 - x_1 - x_2$ was made. Solving the system of equations:

$$x_1 = \frac{1}{3}$$
$$x_2 = \frac{1}{3}$$
$$x_3 = \frac{1}{3}$$

(b) Find the central path $\mathbf{x}(\mu) = (x_1(\mu), x_2(\mu), x_3(\mu))$.

The minimization problem is of the form

$$\min_{x_1,x_2} \quad x_1 + x_2 - \mu\log[x_1] - \mu\log[x_2] - \mu\log[1 - x_1 - x_2]$$

Differentiating with respect to x_1, x_2 we have

$$\nabla_{x_1} \to 1 - \frac{\mu}{x_1} + \frac{\mu}{1 - x_1 - x_2} = 0$$
$$\nabla_{x_2} \to 1 - \frac{\mu}{x_2} + \frac{\mu}{1 - x_1 - x_2} = 0$$

Adding these two equations together, we have that $x_1 = x_2$. Thus:

$$\nabla_x \to 1 - \frac{\mu}{x} + \frac{\mu}{1 - 2x} = 0$$
$$x(1 - 2x) - \mu(1 - 2x) + \mu x = 0$$
$$2x^2 - x(3\mu + 1) + \mu = 0$$
$$x = \frac{3\mu + 1 \pm \sqrt{9\mu^2 - 2\mu + 1}}{4}$$

Now noting that as $\lim_{\mu\to\infty}$ must converge to the analytic center we can eliminate the plus, so that

$$x = \frac{3\mu + 1 - \sqrt{9\mu^2 - 2\mu + 1}}{4}$$

To check the accuracy, lets take the limit as $\mu \to \infty$

$$
\begin{aligned}
\lim_{\mu\to\infty} \frac{3\mu+1-\sqrt{9\mu^2-2\mu+1}}{4} &= \frac{1}{4}\lim_{\mu\to\infty} 3\mu+1-\sqrt{9\mu^2-2\mu+1} \\
&= \frac{1}{4}[\lim_{\mu\to\infty}(3x-\sqrt{9\mu^2-2\mu+1})+1] \\
&= \frac{1}{4}[\lim_{\mu\to\infty}\frac{2\mu-1}{3\mu+\sqrt{9\mu^2-2\mu+1}}+1] \\
&= \frac{1}{4}[2\lim_{\mu\to\infty}\frac{\mu}{3\mu+\sqrt{9\mu^2-2\mu+1}}+1] \\
&= \frac{1}{4}[2\lim_{\mu\to\infty}\frac{1}{3+\frac{\sqrt{9\mu^2-2\mu+1}}{\mu}}+1] \\
&= \frac{1}{4}[2\lim_{\mu\to\infty}\frac{1}{3+\frac{\sqrt{9\mu^2-2\mu+1}}{\mu}}+1] \\
&= \frac{1}{4}[2\frac{1}{\lim_{\mu\to\infty}(3+\frac{\sqrt{9\mu^2-2\mu+1}}{\mu}}) + 1] \\
&= \frac{1}{4}[\frac{2}{\lim_{\mu\to\infty}(\sqrt{\frac{9\mu^2-2\mu}{\mu^2}})+3}+1] \\
&= \frac{1}{4}[\frac{2}{\sqrt{\lim_{\mu\to\infty}\frac{9\mu^2-2\mu}{\mu^2}}+3}+1] \\
&= \frac{1}{4}[\frac{2}{\sqrt{\lim_{\mu\to\infty}(9-\frac{2}{\mu})}+3}+1] \\
&= \frac{1}{4}[\frac{2}{\sqrt{\lim_{\mu\to\infty}(9-\frac{2}{\mu})}+3}+1] \\
&= \frac{1}{4}[\frac{4}{3}] = \frac{1}{3}
\end{aligned}
$$

Thus

$$
\begin{aligned}
x_1(\mu) = x_2(\mu) &= \frac{3\mu+1-\sqrt{9\mu^2-2\mu+1}}{4} \\
x_3(\mu) &= 1 - \frac{3\mu+1-\sqrt{9\mu^2-2\mu+1}}{4}
\end{aligned}
$$

(c) Whos that as μ decreases to 0, $\mathbf{x}(\mu)$ converges to the unique optimal solution.

We see that the

$$\lim_{\mu\to 0} x_{1,2}(\mu) = 0$$

Thus, the optimal solution corresponds to $x_1, x_2 = 0, x_3 = 1$, which is the smallest value the objective function can take while still satisfying the constraint set.

(d) (Computational Team Work) Draw $\mathbf{x}$ part of the the primal-dual potential function level sets

$$\phi_6(\mathbf{x},\mathbf{s}) \le 0 \quad \text{and } \phi_6(\mathbf{x},\mathbf{s}) \le -10$$

and

$$\phi_{12}(\mathbf{x}, \mathbf{s}) \leq 0 \quad \text{and } \phi_{12}(\mathbf{x}, \mathbf{s}) \leq -10$$

respectively in the primal feasible region (on a plane).

(e) Do everything with $f(x) = x_1$

For part a the analytic center remains the same. Now we have

$$\min_{x_1} \quad x_1 - \mu \log[x_1] - \mu \log[x_2] - \mu \log[1 - x_1 - x_2]$$

Differentiating with respect to x_1, x_2 we have

$$\nabla_{x_1} \to 1 - \frac{\mu}{x_1} + \frac{\mu}{1 - x_1 - x_2} = 0$$
$$\nabla_{x_2} \to \frac{\mu}{x_2} + \frac{\mu}{1 - x_1 - x_2} = 0$$

Solving this system of equations we have

$$x_2(\mu) = x_3(\mu) = \frac{1 - x_1(\mu)}{2}$$
$$x_1(\mu) = \frac{1 + 3\mu - \sqrt{9\mu^2 + 2\mu + 1}}{2}$$

Thus as $\mu \to 0$, the optimal solution becomes $x_1 = 0, x_2 = x_3 = \frac{1}{2}$

Problem 4

Questions (a) and (b) of Problem 7, Section 5.9 in textbook

Hint: Use the fact that for any feasible pair $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ of LP,

$$(\mathbf{x} - \mathbf{x}(\mu))^T (\mathbf{s} - \mathbf{s}(\mu)) = 0$$

the optimality of the central path solutions.

Let $(\mathbf{x}(\mu), \mathbf{y}(\mu), \mathbf{s}(\mu))$ be the central path of 5.9. Then prove

(a) The central path point $(\mathbf{x}(\mu), \mathbf{y}(\mu), \mathbf{s}(\mu))$ is bounded for $0 < \mu \leq \mu^0$ and any given $0 < \mu^0 < \infty$.

We have that $(\mathbf{x}(\mu^0) - \mathbf{x}(\mu))^\top (\mathbf{s}(\mu^0) - \mathbf{s}(\mu))^\top = 0$.
Thus

$$\sum_j^n (\mathbf{s}(\mu^0)_j \mathbf{x}(\mu)_j + \mathbf{x}(\mu^0)_j \mathbf{s}(\mu)_j) = n(\mu^0 + \mu) \leq 2n\mu^0$$

Thus

$$\sum_j^n \left(\frac{\mathbf{x}(\mu)_j}{\mathbf{x}(\mu^0)_j} + \frac{\mathbf{s}(\mu)_j}{\mathbf{s}(\mu^0)_j}\right) \leq 2n$$

Thus $\mathbf{x}(\mu), \mathbf{s}(\mu)$ are bounded. Since the KKT condition of the barrier porblems require that $\mathbf{s} = -A^T\mathbf{y} + \nabla f(\mathbf{x})^T$, it follows that since $\mathbf{s}(\mu)$ is bounded, $\mathbf{y}(\mu)$ must be bounded as well.

(b) For $0 < \mu' < \mu$

$$\mathbf{c}^\top \mathbf{x}(\mu') \le \mathbf{c}^\top \mathbf{x}(\mu) \text{ and } \mathbf{b}^\top \mathbf{y}(\mu') \ge \mathbf{b}^\top \mathbf{y}(\mu)$$

Furthermore if $\mathbf{x}(\mu') \neq \mathbf{x}(\mu)$ and $\mathbf{y}(\mu') \neq y(\mu)$,

$$\mathbf{c}^\top \mathbf{x}(\mu') < \mathbf{c}^\top \mathbf{x}(\mu) \text{ and } \mathbf{b}^\top \mathbf{y}(\mu') > \mathbf{b}^\top \mathbf{y}(\mu)$$

Now we know that for a given (μ', μ) that

$$\mathbf{c}^\top \mathbf{x}(\mu) - \mu \sum_j \log[\mathbf{x}(\mu)_j] \le \mathbf{c}^\top \mathbf{x}(\mu') - \mu \sum_j \log[\mathbf{x}(\mu')_j]$$

Since $\mathbf{x}(\mu)$ minimizes $\mathbf{c}^\top \mathbf{x}(\mu) - \mu \sum_j \log[\mathbf{x}(\mu)_j]$ for a given μ. Similarly we know that

$$\mathbf{c}^\top \mathbf{x}(\mu') - \mu' \sum_j \log[\mathbf{x}(\mu')_j] \le \mathbf{c}^\top \mathbf{x}(\mu) - \mu' \sum_j \log[\mathbf{x}(\mu)_j]$$

Adding together these two equations

$$(\mu - \mu') \sum_j \log[\mathbf{x}(\mu')_j] \le (\mu - \mu') \sum_j \log[\mathbf{x}(\mu)]$$

Thus

$$\sum_j \log[\mathbf{x}(\mu')_j] \le \sum_j \log[\mathbf{x}(\mu)_j]$$

Therefore we have that

$$\mathbf{c}^\top \mathbf{x}(\mu') - \mathbf{c}^\top \mathbf{x}(\mu) \le u'[\sum_j \log[\mathbf{x}(\mu')_j] - \sum_j \log[\mathbf{x}(\mu)_j]]$$

Plugging in the inequality that $\sum_j \log[\mathbf{x}(\mu')_j] - \sum_j \log[\mathbf{x}(\mu)_j] \le 0$ we have that

$$\mathbf{c}^\top \mathbf{x}(\mu') - \mathbf{c}^\top \mathbf{x}(\mu) \le u'[\sum_j \log[\mathbf{x}(\mu')_j] - \sum_j \log[\mathbf{x}(\mu)_j]] \le 0$$

And thus that:

$$\mathbf{c}^\top \mathbf{x}(\mu') \le \mathbf{c}^\top \mathbf{x}(\mu)$$

Now if $\mathbf{x}(\mu') \neq \mathbf{x}(\mu)$ the inequalities become strict so that

$$\sum_j \log[\mathbf{x}(\mu')_j] < \sum_j \log[\mathbf{x}(\mu)_j]$$

Thus that

$$\mathbf{c}^\top \mathbf{x}(\mu') < \mathbf{c}^\top \mathbf{x}(\mu)$$

In the dual case we have that for a given μ, $\mathbf{y}(\mu)$ maximizes

$$\mathbf{b}^\top \mathbf{y}(\mu) + \mu \sum_{j=1}^n \log[\mathbf{s}(\mu)_j]$$

Thus for any μ' it must hold that

$$\mathbf{b}^\top \mathbf{y}(\mu) + \mu \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j] \geq \mathbf{b}^\top \mathbf{y}(\mu') + \mu \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j]$$

$$\mathbf{b}^\top \mathbf{y}(\mu') + \mu' \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j] \geq \mathbf{b}^\top \mathbf{y}(\mu) + \mu' \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j]$$

Adding the two equation, we have

$$(\mu - \mu') \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j] \geq (\mu - \mu') \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j]$$

$$\sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j] \geq \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j]$$

Now in the case $\mathbf{y}(\mu) \neq \mathbf{y}(\mu')$ then the inequalities are strict, so that

$$\mathbf{b}^\top \mathbf{y}(\mu) + \mu \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j] > \mathbf{b}^\top \mathbf{y}(\mu') + \mu \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j]$$

$$\mathbf{b}^\top \mathbf{y}(\mu') + \mu' \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j] > \mathbf{b}^\top \mathbf{y}(\mu) + \mu' \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j]$$

$$\sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j] > \sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j]$$

Continuing we have that

$$\mathbf{b}^\top \mathbf{y}(\mu') - \mathbf{b}^\top \mathbf{y}(\mu) \geq \mu' [\sum_{j=1}^{n} \log[\mathbf{s}(\mu')_j] - \sum_{j=1}^{n} \log[\mathbf{s}(\mu)_j]] \geq 0$$

Therefore

$$\mathbf{b}^\top \mathbf{y}(\mu') \geq \mathbf{b}^\top \mathbf{y}(\mu)$$

Or in the strict case that:

$$\mathbf{b}^\top \mathbf{y}(\mu') > \mathbf{b}^\top \mathbf{y}(\mu')$$

Problem 5

Problem 12, Section 6.8, the text book L&Y, where for any given symmetric matrix $D, |D|^2$, is the sum of all its eigenvalue squares, and $|D|_\infty$ is its largest absolute eigenvalue.

Hint: $det(I + D)$ equals the produt of the eigenvalues of $I + D$ Then the proof follows from Taylor expansion.

Prove that if $\mathbf{D} \in \S^n$ and $|D|_\infty < 1$. Then

$$\text{trace}(\mathbf{D}) \geq \log[\det(I + \mathbf{D}) \geq \text{trace}(\mathbf{D}) - \frac{|D|^2}{2(1 - |\mathbf{D}|_\infty)}]$$

We know that the trace$(\mathbf{D}) = \sum_j \lambda_j$ where λ_j is the jth eigenvalue of the matrix D. Furthermore we know that since $\mathbf{D}$ is PSD, that it can be diagonalized so that $\det(I+\mathbf{D}) = \det(X(I+\Sigma)X^{-1}) = \det(I+\Sigma) = \prod_j (1+\lambda_j)$ where Σ is a diagonal matrix of eigenvalues λ_j. Thus we have that

$$\text{trace}(\mathbf{D}) = \sum_j \lambda_j$$
$$\log[\det(I + \mathbf{D})] = \log[\prod_j (1 + \lambda_j)] = \sum_j \log(1 + \lambda_j)$$

Since $\max \lambda_j \leq 1$, we know that $\log(1 + \lambda_j) \leq \lambda_j \; \forall j$. Therefore

$$\text{trace}(\mathbf{D}) = \sum_j \lambda_j \geq \log[\det(I + \mathbf{D}) = \sum_j \log(1 + \lambda_j)$$

Now for a given j we have the taylor expansion

$$\log(1 + \lambda_j) = \lambda_j - \frac{1}{2} \frac{\lambda_j^2}{(1 + c_j)^2}$$

For some $c \in (0, \lambda_j]$ since $\lambda_j < 1$. Now, we know that each c_j is in the radius of $0, \max_j |\lambda_j|$. Thus we have that

$$\frac{1}{(1 + c_j)^2} \leq \frac{1}{1 - \max_j |\lambda_j|}$$

Thus it follows that

$$\log(1 + \lambda_j) \geq \lambda_j - \frac{\lambda_j^2}{2(1 - \max_j |\lambda_j|)}$$

which in matrix forms indicates that

$$\log[\det(I + \mathbf{D})] = \sum_j \log[1 + \lambda_j] \geq \sum_j \lambda_j - \frac{\lambda_j^2}{2(1 - \max_j |\lambda_j|)}$$
$$= \text{trace}(D) - \frac{|D|^2}{2(1 - |\mathbf{D}|_\infty)}$$

Problem 6

Optimization with log-sum-exponential functions arises from smooth approximation for non-smooth optimization. Consider the non-smooth optimization problem:

$$\min_{\mathbf{x}} \max_{1 \leq i < m} (\mathbf{a}_i^\top \mathbf{x} + b_i)$$

given $\mathbf{a}_i \in \mathbb{R}^n, b_i \in \mathbb{R}$.

(a) Derive an equivalent LP problem and write down its dual.

Let A be a matrix of vectors $\mathbf{a}_i$. and $\mathbf{b}$ be a vector of b_i's. The primal optimization problem becomes

$$\min_z \quad z$$

such that :

$$Ax + b \preceq z\mathbf{1}$$

The dual is then

$$\max_y \quad \mathbf{b}^T y$$

such that :

$$A^T y = 0$$
$$\mathbf{1}^T y = 1$$
$$y \succeq 0$$

(b) Suppose we approximate the objective function $\max_{1 \le i \le m}(\mathbf{a}_i^\top \mathbf{x} + b_i)$ with a smooth function and consider the optimization:

$$\min_x \log \Big[\sum_{i=1}^{m} \exp(\mathbf{a}_i^\top \mathbf{x} + b_i\Big]$$

Then z_1 and z_2 be the optimal values of the two formulations, prove that

$$0 \le z_2 - z_1 \le \log(m)$$

Suppose that z^* is dual optimal for the dual general approximated program

$$\max \quad b^T z - \sum_j z_i \log[z_i]$$

Such that:

$$A^T z = 0$$
$$\mathbf{1}^T z = 1$$
$$z \succeq 0$$

In this case we have that z^* is also feasible for the dual of the piecwise linear formulation, that has objective value:

$$b^T z = z_2 + \sum_j z_j^* \log[z_j^*]$$

Furthermore from the concavity of the log we have that

$$\sum_j z_j \log[\frac{1}{z_j}] \le \log[\sum_j 1] = \log m$$

Thus we have that

$$z_1 \ge z_2 + \sum_j z_j^* \log[z_j^*] \ge z_2 + \log(m)$$

Furthermore it holds that

$$\max_i(a_i^T x + b_i) \le \log[\sum_i e^{a_i^T x + b_i}]$$

To prove this, let $r \in \mathbb{R}^m$ and let $m = \max_i r$

$$\begin{aligned}
\log[\sum_i \exp(r_i)] &= \log[\sum_i \frac{\exp(m)}{\exp(m)} \exp(r_i)] \\
&= \log[\exp(m) \sum_i \frac{1}{\exp(m)} \exp(r_i)] \\
&= m + \log(\sum_i \exp(r_i - m)) \\
\log[\sum_i \exp(r_i)] &\geq m
\end{aligned}$$

Therefore we have that $z_1 \leq z_2$. Combining the inequalities yields

$$z_2 - \log(m) \leq z_1 \leq z_2$$
$$0 \leq z_2 - z_1 \leq \log(m)$$

and the proof is complete.

(c) Suppose we use a different function for approximation:

$$\min_{\mathbf{x}} \frac{1}{\gamma} \log \big(\sum_{i=1}^{m} \exp[\gamma(\mathbf{a}_i^\top \mathbf{x} + b_i)] \big)$$

for some $\gamma > 0$. Suppose the optimal value is z_3 derivat a bound for $z_3 - z_1$ similar as above. What happens as $\gamma \to \infty$.
This problem can be reformulated as:

$$\min_{\mathbf{x}} \quad \frac{1}{\gamma} \log \big(\sum_{i=1}^{m} \exp[\gamma(y_i)] \big)$$

Such that:

$$Ax + b = y$$

Forming the Lagrangian we have

$$L(x, y, \lambda) = \frac{1}{\gamma} \log \big(\sum_{i=1}^{m} \exp[\gamma(y_i)] \big) + \lambda^T (Ax + b - y)$$

Now notice that the lagrangian is unbounded below as a function of x unless $A^T \lambda = 0$. We are aiming to minimize the lagragian, or equivalently to maximize the conjugate function:

$$c(\lambda) = \sup_y \{\lambda^T y - \log \big(\sum_{i=1}^{m} \exp[y_i] \big)\}$$

Thus we see that if $\lambda_k < 0$, setting $y_k = c, y_i = 0 \forall i \neq k$ then $\lim_{c \to -\infty}$ of the expression goes $-\infty$., Thus $\lambda_k > 0$. Similarly if $\lambda \succeq 0$, and $\mathbf{1}^T \lambda \neq 1$, then, we can have $y = c\mathbf{1}$ to find that

$$\lim_{t \to \infty} \lambda^T y - \log[\sum_i \exp(y_i)] = \lim_{t \to \infty} c\mathbf{1}^T \lambda - \log(m) - c$$

which goes to infinity or negative infinity depending on $\mathbf{1}^T z - 1$.

Taking the derivative with respect to y and setting it equal to zero we have

$$\lambda_i = \frac{e^{y_i}}{\sum_j e^{x_j}}$$

Plugging this back into $g(\lambda)$ we have $g^*(\lambda) = \sum_i y_i \log[y_i]$
In summary we have that the conjugate function equals:

$$c(\lambda) = \begin{cases} \sum_i y_i \log[y_i] \text{ if } \lambda \succeq 0 \text{ and } \mathbf{1}^T\lambda = 1 \\ 0 \text{ otherwise} \end{cases}$$

The lagrangian dual function can be given for $\lambda \succeq 0, \mathbf{1}^T\lambda = 1, A^T\lambda = 0$.

$$g(\lambda) = b^T\lambda - \frac{1}{\gamma}\sum_i \lambda_i \log[\lambda_i]$$

So that the dual problem can be formulated as:

$$\begin{aligned} \max_{\lambda} \quad & b^T\lambda - \frac{1}{\gamma}\sum_i \lambda_i \log[\lambda_i] \\ \text{such that: } & A^T\lambda = 0 \\ & \mathbf{1}^T\lambda = 1 \end{aligned}$$

Let z_3 be an optimal solution to the optimization above. We then know that z_3 is also feasible for the dual of the piecewise linear formulation, which has objective value

$$b^T\lambda = z_3 + \frac{1}{\gamma}\sum_i \lambda_i^* \log(z_i^*)$$

Thus we have that

$$z_1 \geq z_3 + \frac{1}{\gamma}\sum_i \lambda_i^* \log(z_i^*) \geq z_3 - \frac{1}{\gamma}\log[m]$$

Furthermore it follows as in the previous formulation that $z_3 \geq z_1$. It thus follows that

$$z_3 - \frac{1}{\gamma}\log[m] \leq z_1 \leq z_3$$

and therefore that

$$0 \leq z_3 - z_1 \leq \frac{1}{\gamma}\log[m]$$

Now, notice, as $\gamma \to \infty$, then $z_3 \to z_1$.

Code

```
function [ ] = ADMM(idx, beta, iter, A)

x1(1) = rand(1);
x2(1) = rand(1);
x3(1) = rand(1);
y(:, 1) = rand(3,1);
syms x_1 x_2 x_3
```

```
for k=1: iter
    x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
            A(:, 2)*x2(k) + A(:, 3)*x3(k)) == 0, x_1);

    x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
            A(:, 2)*x_2 + A(:, 3)*x3(k)) == 0, x_2);

    x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
            A(:, 2)*x2(k+1) + A(:, 3)*x_3) == 0, x_3);

    y(:, k+1)=y(:, k)+beta * (A(:, 1)*x1(k+1) + A(:, 2)*x2(k+1) + A(:, 3)*x3(k+1));




    if mod(k,10) == 0
        disp( sprintf ('----- beta: %d ------', beta));
        disp( sprintf ('------- Iteration: %d ----------', k));
        disp('        x1           x2           x3')
        disp([  x1(k), x2(k), x3(k)])

    end

end

if idx == 0
    add = ' without objective function';
else
    add = ' with objective function';
end

figure()
subplot(3,1,1)
plot(1:k+1, x1, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Estimate of x_1 for beta=', num2str(beta), 'and ', add))

subplot(3,1,2)
plot(1:k+1, x2, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Estimate of x_2 for beta=', num2str(beta), 'and ', add))

subplot(3,1,3)
plot(1:k+1, x3, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Estimate of x_3 for beta=', num2str(beta), add))
hold off

end
```

ADMM.m

```
function [x, x0, iter, fvals, gvals, hvals] = Newton(f, df, hessian, x_initial, x0_intial, a, b,ALPHA, MAX_ITER, TOL,debug)


    x = x_initial;
    x0 = x0_intial;

    x_cat_prev = [x; x0];

    H_prev = inv(hessian(x,x0,a,b));



    iter = 1;
    fvals = [];
    gvals = [];
    hvals = [];
    fvals(iter) = f(x,x0, a,b);
    gvals(iter) = norm(df(x,x0,a,b));
    hvals(iter) = norm(inv(H_prev));

    progress = @(iter,x,x0, fvals, delta_f, delta_x, lambda, alpha) fprintf('\n------------------ASDM iter = %d:------------------\n
        (x = %s, x_0=%f, F(x)=%f delta_F = %f,delta_x = %f, lambda= %f, alpha = %f) \n
        ---------------------------------------- \n', ...
    iter, mat2str(x,6),num2str(x0), fvals, delta_f, delta_x, lambda, alpha);


    if debug
            disp( sprintf ('-------------------------Iteration: %d----------------------------------', iter));
            disp('      x_1         x_2         x_0       f(x)      delta_F      delta_x      lambda      alpha')
            disp([  x(1), x(2), x0, fvals(iter), NaN, NaN, 0, NaN])

    end

    norm_grad = 10000;
    delta_f = 10000;
    delta_x = 10000;
```

```
    alpha = ALPHA;


    while iter < MAX_ITER
        if abs(norm_grad) < TOL
            disp( sprintf ('----GRADIENT NORM IS BELOW TOLERANCE CONVERGENCE OF FUNCTION AFTER %d ITERATIONS-----', iter))
            disp( sprintf ('----------------------------FINAL Iteration: %d------------------------------------', iter));
            disp('        x_1          x_2          x_0        f(x)        delta_F      delta_x         alpha      NORMGRAD  HessianNotm')
            disp([  x(1), x(2), x0, fvals ( iter ),   delta_f ,   delta_x  , alpha, gvals( iter )  hvals( iter ) ])
            break;
        end

        if delta_x < 1e-8
            iter = iter + 1;
            disp( sprintf ('CHANGE IN X IS TINY, CONVERGENCE OF FUNCTION AFTER %d ITERATIONS', iter))
            disp( sprintf ('----------------------------FINAL Iteration: %d------------------------------------', iter));
            disp('        x_1          x_2          x_0        f(x)        delta_F      delta_x         alpha      NORMGRAD  HessianNotm')
            disp([  x(1), x(2), x0, fvals ( iter ),   delta_f ,   delta_x  , alpha, gvals( iter )  hvals( iter ) ])
            break;
        end

        iter = iter + 1;

        g_prev = df(x,x0,a,b);
        d = H_prev*g_prev;
        alpha = choose_alpha(ALPHA, -d, x, x0, a,b, f)
        x_cat_new = x_cat_prev - ALPHA*d;


        x = x_cat_new(1:end-1);
        x0 = x_cat_new(end);
        g_new = df(x,x0,a,b);

        fvals ( iter ) = f(x,x0, a, b);
        gvals( iter ) = norm(g_new);


        q = g_new - g_prev;
        p = - ALPHA*d;

        H_new = H_prev +(p*p')/(p'*q) - (H_prev*(q*q')*H_prev)/(q'*H_prev*q);
        hvals( iter ) = norm(inv(H_new));

        delta_f = fvals ( iter ) - fvals( iter - 1);
        delta_x = norm(x_cat_new - x_cat_prev, 2);
        norm_grad = norm(df(x,x0,a,b) ,2) ;




        if debug
            disp( sprintf ('----------------------Iteration: %d----------------------------------', iter));
            disp('        x_1          x_2          x_0        f(x)        delta_F      delta_x         alpha      NORMGRAD  HessianNotm')
            disp([  x(1), x(2), x0, fvals ( iter ),   delta_f ,   delta_x  , alpha, gvals( iter )  hvals( iter ) ])
        end

        x_cat_prev = x_cat_new ;
        H_prev = H_new;
        g_prev = g_new;
    end

    figure () ;
    subplot (3,1,1)
    plot (1: iter , fvals , 'LineWidth',2); grid on;
    title (' Objective Function Value DFP'); xlabel (' Iteration '); ylabel (' f(x)' );
    subplot (3,1,2)
      plot (1: iter , gvals , 'LineWidth',2); grid on;
    title ('Norm of Gradient of Objective Function DFP'); xlabel (' Iteration '); ylabel ('g(x)' );
        subplot (3,1,3)
      plot (1: iter , hvals , 'LineWidth',2); grid on;
    title ('Norm of Hessian of Objective Function DFP'); xlabel (' Iteration '); ylabel ('F(x)' );






end
```

DFP.m

```
function [x, x0, iter , fvals , gvals , hvals ] = Newton(f, df, hessian , x_initial , x0_intial , a, b,ALPHA, MAX_ITER, TOL,debug)
```

```
x =  x_initial ;
x0 =  x0_intial ;

 x_cat_prev  = [x;  x0];




 iter  = 1;
 fvals  = [];
gvals  = [];
hvals  = [];
 fvals ( iter ) = f(x,x0,  a,b);
gvals( iter ) = norm(df(x,x0,a,b));
hvals( iter ) = norm(hessian(x,x0,a,b));

 progress  = @(iter ,x,x0,  fvals ,   delta_f , delta_x ,  lambda,  alpha)   fprintf ('\n-------------------ASDM iter = %d:-------------------\n
       (x = %s, x_0=%f, F(x)=%f delta_F = %f,delta_x = %f, lambda= %f, alpha = %f) \n
       ---------------------------------------------- \n' , ...
 iter ,  mat2str(x,6) ,num2str(x0),   fvals ,   delta_f ,  delta_x ,  lambda,  alpha);


 if  debug
            disp( sprintf ('-----------------------Iteration: %d-----------------------------------', iter));
            disp('        x_1             x_2            x_0        f(x)         delta_F      delta_x      lambda      alpha' )
            disp([   x(1),  x(2),  x0,   fvals ( iter ),  NaN,  NaN, 0,  NaN])

end

norm_grad = 10000;
 delta_f  = 10000;
 delta_x  = 10000;
alpha  = 10000;



while  iter  < MAX_ITER
      if  abs(norm_grad) < TOL
            disp( sprintf ('----GRADIENT NORM IS BELOW TOLERANCE CONVERGENCE OF FUNCTION AFTER %d ITERATIONS-----', iter))
            disp( sprintf ('------------------------------FINAL Iteration: %d
      -----------------------------------', iter));
            disp('        x_1             x_2            x_0        f(x)         delta_F      delta_x          alpha     NORMGRAD  HessianNotm')
            disp([   x(1),  x(2),  x0,   fvals ( iter ),    delta_f ,    delta_x   ,  alpha,  gvals( iter )  hvals( iter ) ])
            break;
      end

      if  delta_x  < 1e-8
            iter  =  iter  + 1;
            disp( sprintf ('CHANGE IN X IS TINY, CONVERGENCE OF FUNCTION AFTER %d ITERATIONS', iter))
            disp( sprintf ('------------------------------FINAL Iteration: %d
      -----------------------------------', iter));
            disp('        x_1             x_2            x_0        f(x)         delta_F      delta_x          alpha     NORMGRAD  HessianNotm')
            disp([   x(1),  x(2),  x0,   fvals ( iter ),    delta_f ,    delta_x   ,  alpha,  gvals( iter )  hvals( iter ) ])
            break;
      end

       iter  =  iter  + 1;


      x_cat_new  =  x_cat_prev   - (hessian(x,x0,a,b)\df(x,x0,a,b));


      x = x_cat_new(1:end-1);
      x0 = x_cat_new(end);

       fvals ( iter ) = f(x,x0,  a,  b);
      gvals( iter ) = norm(df(x,x0,a,b));
      hvals( iter ) = norm(hessian(x,x0,a,b));

       delta_f  =  fvals ( iter ) - fvals( iter   - 1);
       delta_x  = norm(x_cat_new - x_cat_prev,  2);
      norm_grad = norm(df(x,x0,a,b),2);



      if  debug
            disp( sprintf ('-----------------------Iteration: %d-----------------------------------', iter));
            disp('        x_1             x_2            x_0        f(x)         delta_F      delta_x          alpha     NORMGRAD  HessianNotm')
            disp([   x(1),  x(2),  x0,   fvals ( iter ),    delta_f ,    delta_x   ,  alpha,  gvals( iter )  hvals( iter ) ])
      end

       x_cat_prev  = x_cat_new  ;

end
```

```
    figure();
    subplot(3,1,1)
    plot(1:iter, fvals, 'LineWidth',2); grid on;
     title('Objective Function Value Newton'); xlabel(' Iteration '); ylabel('f(x)');
    subplot(3,1,2)
      plot(1:iter, gvals, 'LineWidth',2); grid on;
     title('Norm of Gradient of Objective Function Newton'); xlabel(' Iteration '); ylabel('g(x)');
        subplot(3,1,3)
      plot(1:iter, hvals, 'LineWidth',2); grid on;
     title('Norm of Hessian of Objective Function Newton'); xlabel(' Iteration '); ylabel('F(x)');




end
```

Newton.m

```
clear all
close all

Hw3_path = '/Users/jacobperricone/Desktop/STANFORD/w16/git_cme307/HW3'
addpath(genpath(Hw3_path))

%% Problem 5
a = [0, 1, 0; 0 0, 1];
b = [0,-1, 0; 0, 0,-1];
mu = [0, 10^-5];

 x_initial = [.5;1.25];
 x0_initial = .124;
ALPHA = .05;
MAX_ITER = 25090;
TOL = .000001;

f = @(x,x0, a, b) sum(log(1 + exp(-a'*x - x0))) + sum(log(1 + exp(b'*x + x0)));
f2 = @(x,x0, a, b, mu) sum(log(1 + exp(-a'*x - x0))) + sum(log(1 + exp(b'*x + x0)));
df = @(x,x0, a, b) ...
    [sum((-a(1,:)'.*exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((b(1,:)'.*exp(b'*x + x0)) ./ (1 + exp(b'*x + x0))); ...
    sum((-a(2,:)'.*exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((b(2,:)'.*exp(b'*x + x0)) ./ (1 + exp(b'*x + x0))); ...
    sum((-exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((exp(b'*x + x0)) ./ (1 + exp(b'*x + x0)))];

df2 = @(x,x0, a, b,mu) ...
    [sum((-a(1,:)'.*exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((b(1,:)'.*exp(b'*x + x0)) ./ (1 + exp(b'*x + x0))); ...
    sum((-a(2,:)'.*exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((b(2,:)'.*exp(b'*x + x0)) ./ (1 + exp(b'*x + x0))); ...
    sum((-exp(-a'*x - x0)) ./ (1 + exp(-a'*x - x0))) ...
    + sum((exp(b'*x + x0)) ./ (1 + exp(b'*x + x0)))];

%%
newa = vertcat(a,ones(1,size(a,2)));
tmpa = zeros(size(a,2), size(a,1)*(size(a,2) + 1));
tmpa(1,1:3) = newa(:,1)';
tmpa(2,4:6) = newa(:,2)';
tmpa(3,7:9) = newa(:,3)';
tmpa = reshape(newa*tmpa,3,3,size(a,2));

newb = vertcat(b,ones(1,size(b,2)));
tmpb = zeros(size(b,2), size(b,1)*(size(b,2) + 1));
tmpb(1,1:3) = newb(:,1)';
tmpb(2,4:6) = newb(:,2)';
tmpb(3,7:9) = newb(:,3)';
tmpb = reshape(newb*tmpb,3,3,size(b,2));


z = @(x,x0,a) (repmat(reshape(exp(-a'*x - x0)./(1 + exp(-a'*x - x0)).^2,1,1,size(newa,2)),size(newa,1),size(newa,1),1));
zbar = @(x,x0,b)(repmat(reshape(exp(b'*x + x0)./(1 + exp(b'*x + x0)).^2,1,1,size(newb,2)),size(newb,1),size(newb,1),1));


 hessian = @(x,x0,a,b)(sum(z(x,x0,a).*tmpa,3) + sum(zbar(x,x0,b).*tmpb,3));
%%



[X_Newton, X0_Newton, iter_N,fvals_N, gvals_N, hvals_N] = Newton(f, df, hessian, x_initial, x0_initial, a, b,ALPHA, MAX_ITER, TOL,0);
[X_DFP, X0_DFP, iter_DFP,fvals_DFP, gvals_DFP,hvals_DFP] = DFP(f, df, hessian, x_initial, x0_initial, a, b,.1, MAX_ITER, TOL,0);
[x_SDM, x0_SDM, iter_SDM,fvals_SDM, gvals_SDM] = SDM(f2, df2, x_initial, x0_initial, a, b,ALPHA, MAX_ITER, TOL,0,0);

[x_ASDM, x0_ASMD, iter_ASDM,fvals_ASDM, gvals_ASDM] = ASDM(f2, df2, x_initial, x0_initial, a, b,1/ALPHA, MAX_ITER, TOL,0,0);
```

```
[x_CGD, x0_CGD, iter_CGD,fvals_CGD, gvals_CGD] = CGD(f2, df2, x_initial ,   x0_initial  , a, b, MAX_ITER, TOL,0,0);

[x_BB, x0_BB, iter_BB ,fvals_BB, gvals_BB] = BB1(f2, df2,   x_initial ,   x0_initial  , a, b, MAX_ITER, TOL,1,1,0);
%% Generate random data

 figure ()
subplot (2,1,1)
loglog (1: iter_N ,  fvals_N )
hold on
loglog (1: iter_DFP, fvals_DFP)
hold on
loglog (1: iter_SDM, fvals_SDM)
hold on
loglog (1: iter_ASDM, fvals_ASDM)
hold on
loglog (1: iter_CGD, fvals_CGD)
hold on
loglog (1: iter_BB , fvals_BB )
legend('Newton', 'DFP', 'SDM', 'ASDM', 'CGD','BB')
 title ('Function Value vs.  Iteration  (LogSpace)')
xlabel ('$\log[\mbox{Iteration}]$',' Interpreter ', 'LaTex')
ylabel ('$\log[f(x)]$', ' Interpreter ', 'LaTex')

subplot (2,1,2)
loglog (1: iter_N , gvals_N)
hold on
loglog (1: iter_DFP, gvals_DFP)
hold on
loglog (1: iter_SDM, gvals_SDM)
hold on
loglog (1: iter_ASDM, gvals_ASDM)
hold on
loglog (1: iter_CGD, gvals_CGD)
hold on
loglog (1: iter_BB , gvals_BB)
legend('Newton', 'DFP', 'SDM', 'ASDM', 'CGD','BB')
 title ('Norm Gradient Value vs.  Iteration  (LogSpace)')
xlabel ('$\log[\mbox{Iteration}]$',' Interpreter ', 'LaTex')
ylabel ('$\log[g(x)]$', ' Interpreter ', 'LaTex')

%%
 figure ()
subplot (3,1,1)
loglog (1: iter_N ,  fvals_N )
hold on
loglog (1: iter_DFP, fvals_DFP)
hold on
legend('Newton', 'DFP')
 title ('Function Value vs.  Iteration  (LogSpace)')
xlabel ('$\log[\mbox{Iteration}]$',' Interpreter ', 'LaTex')
ylabel ('$\log[f(x)]$', ' Interpreter ', 'LaTex')

subplot (3,1,2)
loglog (1: iter_N , gvals_N)
hold on
loglog (1: iter_DFP, gvals_DFP)
hold on
legend('Newton', 'DFP')
 title ('Norm Gradient Value vs.  Iteration  (LogSpace)')
xlabel ('$\log[\mbox{Iteration}]$',' Interpreter ', 'LaTex')
ylabel ('$\log[g(x)]$', ' Interpreter ', 'LaTex')


subplot (3,1,3)
loglog (1: iter_N , hvals_N)
hold on
loglog (1: iter_DFP, hvals_DFP)
hold on
legend('Newton', 'DFP')
 title ('Norm Hessian Value vs.  Iteration  (LogSpace)')
xlabel ('$\log[\mbox{Iteration}]$',' Interpreter ', 'LaTex')
ylabel ('$\log[F(x)]$', ' Interpreter ', 'LaTex')
```

Problem2_Client.m

```
clear all
close all

%% Problem 7
% Part (a) and (b)

A = [1, 1, 1; 1, 1, 2; 1, 2, 2];
beta = [.1, 1, 10];

for i=1:3
    % Without the objective function .
```

```
    ADMM(0, beta(i), 100, A)

    % With the objective function.
    ADMM(1, beta(i), 100, A)
end



%% Random Permutation
% Part (c)

for i=1:3

    % Without the objective function and with permuation.
    PermADMM(0, beta(i), 100, A)

    % With objective function and permuation.
    PermADMM(1, beta(i), 100, A)

end
```

Problem7.m

```
function [out, points] = evaluate_candidacy(x, s, p, cutoff)

    out = p*diag(log(x'*s))' - sum(log(x.*s),1);
    points = x(:, out < cutoff);


end
```

evaluate_candidacy.m

```
function [spoints] = generate_s_points_two(npoints)
    y = rand(1, npoints);
    spoints = vertcat(1 + y, y, y);

end
```

generate_s_points_two.m

```
function [ Z ] = CVXSolZ(a, A, d, idx)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
C = 10*rand(4,4);
cvx_expert true
cvx_begin sdp quiet
            variable Z(4,4) symmetric
            minimize(sum(dot(C, Z)) + idx(1)*trace(Z) + ...
                idx(2)*power(2,norm(Z(3, 1:2)' - a(1:2, 3)) + norm(Z(4, 1:2)' - a(1:2, 1))))
        %idx(2)*power(2,sqrt(power(2,Z(3,1)-a(1,3))+power(2,Z(3,2)-a(2,3))) + ...
                %sqrt(power(2,Z(4,1)-a(1,1))+power(2,Z(4,2)-a(2,1)))))
            subject to

            sum(dot(A(:,1)*A(:,1)', Z)) == 1;
            sum(dot(A(:,2)*A(:,2)', Z)) == 1;
            sum(dot(A(:,3)*A(:,3)', Z)) == 2;

            for i = 1:2

                sum(dot([a(:, i); -1; 0] * [a(:, i); -1; 0]', Z)) ...
                    == d(i)^2;
                sum(dot([a(:, i+1); 0; -1] * [a(:, i+1); 0; -1]', Z)) ...
                    == d(i+2)^2;
            end

            sum(dot([0; 0; 1; -1] * [0; 0; 1; -1]', Z)) == d(5)^2 ;

            Z >= 0;
cvx_end


end
```

CVXSolZ.m

```
clear all
close all
%%
a = [1, -1, 0; 0, 0, 2];
```

```
idx =[1, 0; -1, 0; 0, 1; 0, -1];

DMethods_error_C = [];
DMethods_error_V = [];
for i = 1:3
%
    Methods_error_C = [];
    Methods_error_V = [];
    [X_1, Y_1] = meshgrid( linspace (-1.5, 1.5,20) ' , linspace (-.5,2.5,20) ' ) ;

    Cx1_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;
    Cx2_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;
    Ctotal_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;


    Vx1_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;
    Vx2_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;
    Vtotal_error_SDP = zeros( size (X_1,1), size (Y_1,1)) ;

    A1 = [1; 0; 0; 0];    A2 = [0; 1; 0; 0];    A3 = [1; 1; 0; 0];
    A = [A1, A2, A3];

    for j=1: size (X_1, 1)

        for l=1: size (Y_1,1)

            x1_center = [0,1];
            dist_x1_center = pdist2 ( x1_center , a (:,1:2) ' ) ;

            x1_vertex = [.95,.05];
            dist_x1_vertex = pdist2 ( x1_vertex , a (:,1:2) ' ) ;

            x2 = [X_1(j,1) , Y_1(j,1) ];
            dist_x2 = pdist2 (x2, a (:,2:3) ' ) ;

            dhat_center = pdist2 ( x1_center , x2);
            dhat_vertex = pdist2 ( x1_vertex , x2);


            d = [ dist_x1_center (1) , dist_x1_center (2) , dist_x2 (1) , ...
            dist_x2 (2) , dhat_center ];



            % Minimize the trace of Z
            Z = CVXSolZ(a, A, d, idx( i , :) ) ;

            x_1 = [Z(3,1) , Z(3,2) ]' ;
            x_2 = [Z(4,1) , Z(4,2) ]' ;




            Cx1_error_SDP(j,1) = norm(x_1 - x1_center) ;
            Cx2_error_SDP(j,1) = norm(x_2 - x2);
            Ctotal_error_SDP ( j ,1) = Cx1_error_SDP(j,1) + Cx2_error_SDP(j,1) ;
            Cdistance( j ,1) = dhat_center ;

            Methods_error_C = [Methods_error_C, Cx1_error_SDP(j,1) + Cx2_error_SDP(j,1) ];

            d = [ dist_x1_vertex (1) , dist_x1_vertex (2) , dist_x2 (1) , ...
                dist_x2 (2) , dhat_vertex ];


            % Minimize the trace of Z
            Z = CVXSolZ(a, A, d, idx( i , :) ) ;

            x_1 = [Z(3,1) , Z(3,2) ]' ;
            x_2 = [Z(4,1) , Z(4,2) ]' ;

            if isnan(x_1(1)) || isnan(x_1(2))
                disp('YOOx1')
                x_1;
            end

            if isnan(x_1(1)) || isnan(x_1(2))
                disp('YOOx2')
                x_2;
            end

            Vx1_error_SDP(j,1) = norm(x_1 - x1_vertex);
            Vx2_error_SDP(j,1) = norm(x_2 - x2);
            Vtotal_error_SDP ( j ,1) = Vx1_error_SDP(j,1) + Vx2_error_SDP(j,1) ;
            Vdistance( j ,1) = dhat_vertex ;

            Methods_error_V = [Methods_error_V, Vx1_error_SDP(j,1) + Vx2_error_SDP(j,1) ];

        end
```

```
end

%%
idx =[1, 0; -1, 0; 0, 1; 0, -1];
col = idx(i ,:) ;
if col(1) == 1 || col(2) == 0
     addinfo = ' Objective is to Minimize C\cdot Z + tr(Z).';

elseif col(1) == -1 col(2) == 0
     addinfo = ' Objective is to Minimize C\cdot Z - tr(Z).';

elseif col(1) == 0 || col(2) == 1
     addinfo = ' Objective is to Minimize C\cdotZ + (d_{13} + d_{21})^2';

else
     addinfo = ' Objective is to Minimize C\cdot Z - (d_{13} + d_{21})^2';
end

%%
figure ()
subplot (3,1,1)

mesh(X_1, Y_1, Cx1_error_SDP)
colormap hsv
alpha (.4)
colorbar
view(-30,30); camlight; axis image


title ( strcat ('Magntitude of X_1 Error with X_1 Fixed at (x = 0, y = 1) in Hull.', addinfo), 'FontSize', 10)
xlabel ('X Data')
ylabel ('Y Data')
zlabel ('Error')

[mx,k] = min(Cx1_error_SDP(:));
[ix, jx] = ind2sub( size(Cx1_error_SDP),k);
dim = [.10 .605 .3 .3];
str = strcat ('Minimum Error of (', num2str(Cx1_error_SDP(ix,jx)), ')', ' at X_2: (', num2str(X_1(ix,jx)), ',', num2str(Y_1(ix,jx)), ')');
annotation ('textbox',dim,'String', str, 'FitBoxToText','on', 'FontSize',10);




hold on
tmp = a;
tmp(:,4) = tmp(:,1);
plot3(tmp(1,:), tmp(2,:), zeros( size(tmp(2,:))), '-r')
plot3( x1_center(1), x1_center(2), 0, '-*b','MarkerSize',10)
xlabel ('X Data')
ylabel ('Y Data')
hold off


subplot (3,1,2)
surf(X_1, Y_1, Cx2_error_SDP)
colormap hsv
alpha (.4)
colorbar
view(-30,30); camlight; axis image

title ( strcat ('Magntitude of X_2 Error with X_1 Fixed at (x = 0, y = 1) in Hull', addinfo), 'FontSize', 10)
xlabel ('X Data')
ylabel ('Y Data')
zlabel ('Error')

[mx,k] = min(Cx2_error_SDP(:));
[ix, jx] = ind2sub( size(Cx2_error_SDP),k);
dim = [.10 .295 .3 .3];
str = strcat ('Minimum Error of (', num2str(Cx2_error_SDP(ix,jx)), ')', ' at X_2: (', num2str(X_1(ix,jx)), ',', num2str(Y_1(ix,jx)), ')');
     annotation ('textbox',dim,'String', str, 'FitBoxToText','on', 'FontSize',10);

hold on
tmp = a;
tmp(:,4) = tmp(:,1);
plot3(tmp(1,:), tmp(2,:), zeros( size(tmp(2,:))), '-r')
plot3( x1_center(1), x1_center(2), 0, '-*b','MarkerSize',10)
xlabel ('X Data')
ylabel ('Y Data')
hold off


subplot (3,1,3)
surf(X_1, Y_1, Ctotal_error_SDP )
colormap hsv
alpha (.4)
colorbar
```

```
view(-30,30); camlight;  axis  image


 title ( strcat ('Magntitude of  Total  Error  with  X_1 Fixed  at  (x = 0,  y = 1)  in  Hull' ,  addinfo) ,  'FontSize' ,  10)
xlabel ('X Data')
ylabel ('Y Data')
zlabel ('Error' )

[mx,k] = min(Ctotal_error_SDP (:) ) ;
[ix , jx ]  = ind2sub( size ( Ctotal_error_SDP ) ,k) ;
dim = [.10  .009  .3   .3];
str  =  strcat ('Minimum Error of (' ,  num2str( Ctotal_error_SDP (ix , jx )) , ')' ,  '  at  X_2: (' ,  num2str(X_1(ix , jx )) ,  ',' ,  num2str(Y_1(ix , jx )) , ')' ) ;
annotation ('textbox' ,dim,'String' , str , 'FitBoxToText', 'on',  'FontSize' ,10) ;

hold  on
tmp = a;
tmp (:,4)  = tmp (:,1) ;
plot3 (tmp (1,:) ,  tmp (2,:) ,  zeros ( size (tmp (2,:) ) ) ,  '-r')
plot3 ( x1_center (1) ,  x1_center (2) ,  0,  '-*b','MarkerSize',10)
xlabel ('X Data')
ylabel ('Y Data')
hold  off
%
 figure ()
subplot (3,1,1)
surf (X_1,  Y_1,  Vx1_error_SDP)
colormap hsv
alpha (.4)
colorbar
view(-30,30); camlight;  axis  image


 title ( strcat ('Magntitude of  X_1 Error  with  X_1 Fixed  at  (x =  .95,  y = .05)  in  Hull' ,  addinfo) ,  'FontSize' ,  10)
xlabel ('X Data')
ylabel ('Y Data')
zlabel ('Error' )


[mx,k] = min(Vx1_error_SDP(:));
[ix , jx ]  = ind2sub( size (Vx1_error_SDP),k);
dim = [.10  .595  .3   .3];
str  =  strcat ('Minimum Error of (' ,  num2str(Vx1_error_SDP(ix,jx )) , ')' ,  '  at  X_2: (' ,  num2str(X_1(ix , jx )) ,  ',' ,  num2str(Y_1(ix , jx )) , ')' ) ;
annotation ('textbox' ,dim,'String' , str , 'FitBoxToText', 'on',  'FontSize' ,10) ;

hold  on
tmp = a;
tmp (:,4)  = tmp (:,1) ;
plot3 (tmp (1,:) ,  tmp (2,:) ,  zeros ( size (tmp (2,:) ) ) ,  '-r')
plot3 ( x1_vertex (1) ,  x1_vertex (2) ,  0,  '-*b','MarkerSize',10)
xlabel ('X Data')
ylabel ('Y Data')
hold  off



subplot (3,1,2)
surf (X_1,  Y_1,  Vx2_error_SDP)
colormap hsv
alpha (.4)
colorbar
view(-30,30); camlight;  axis  image


 title ( strcat ('Magntitude of  X_2 Error  with  X_1 Fixed  at  (x =  .95, y = .05  )  in  Hull' ,  addinfo) ,  'FontSize' ,  10)
xlabel ('X Data')
ylabel ('Y Data')
zlabel ('Error' )


[mx,k] = min(Vx2_error_SDP(:));
[ix , jx ]  = ind2sub( size (Vx2_error_SDP),k);
dim = [.10  .295  .3   .3];
str  =  strcat ('Minimum Error of (' ,  num2str(Vx2_error_SDP(ix,jx )) , ')' ,  '  at  X_2: (' ,  num2str(X_1(ix , jx )) ,  ',' ,  num2str(Y_1(ix , jx )) , ')' ) ;
annotation ('textbox' ,dim,'String' , str , 'FitBoxToText', 'on',  'FontSize' ,10) ;

hold  on
tmp = a;
tmp (:,4)  = tmp (:,1) ;
plot3 (tmp (1,:) ,  tmp (2,:) ,  zeros ( size (tmp (2,:) ) ) ,  '-r')
plot3 ( x1_vertex (1) ,  x1_vertex (2) ,  0,  '-*b','MarkerSize',10)
xlabel ('X Data')
ylabel ('Y Data')
hold  off



subplot (3,1,3)
surf (X_1,  Y_1,  Vtotal_error_SDP )
```

```
    colormap hsv
    alpha (.4)
    colorbar
    view(-30,30); camlight; axis image


    title ( strcat ('Magntitude of Total Error with X_1 Fixed at (x = .95,y = .05) in Hull', addinfo), 'FontSize', 10)
    xlabel ('X Data')
    ylabel ('Y Data')
    zlabel ('Error')
    [mx,k] = min(Vtotal_error_SDP (:) );
    [ix, jx] = ind2sub( size (Vtotal_error_SDP), k);
    dim = [.10 .011 .3 .3];
    str = strcat ('Minimum Error of (', num2str(Vtotal_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ',', num2str(Y_1(ix,jx)),')');
    annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);


    hold on
    tmp = a;
    tmp (:,4) = tmp (:,1) ;
    plot3 (tmp (1,:) , tmp (2,:) , zeros( size (tmp (2,:) )), '-r')
    plot3 ( x1_vertex (1), x1_vertex (2), 0, '-*b','MarkerSize',10)
    xlabel ('X Data')
    ylabel ('Y Data')
    hold off


    DMethods_error_C = [DMethods_error_C; Methods_error_C];
    DMethods_error_V = [DMethods_error_V; Methods_error_V];

end
figure ()
plot (1: length (DMethods_error_C(1, :)), DMethods_error_C(1, :))
hold on
plot (1: length (DMethods_error_C(2, :)), DMethods_error_C(2, :))
hold on
plot (1: length (DMethods_error_C(3, :)), DMethods_error_C(3, :))
title ('Differences Across Methods in Estimating Fixed X = (0, 1)')
legend('With tr(Z)', 'With -tr(Z)', 'With (d_{13} + d_{21})^2')
hold off

figure ()
plot (1: length (DMethods_error_V(1, :)), DMethods_error_V(1, :))
hold on
plot (1: length (DMethods_error_V(2, :)), DMethods_error_V(2, :))
hold on
plot (1: length (DMethods_error_V(3, :)), DMethods_error_V(3, :))
title ('Differences Across Methods in Estimating Fixed X = (.95, .05)')
legend('With tr(Z)', 'With -tr(Z)', 'With (d_{13} + d_{21})^2')
```

Hw4Problem8.m

```
function [ ] = PermADMM(idx, beta, iter, A)

x1(1) = rand(1);
x2(1) = rand(1);
x3(1) = rand(1);
y (:, 1) = rand (3,1) ;
syms x_1 x_2 x_3

for k=1: iter
    order = randperm(3,3);

    if order (1)==1
        x1(k+1) = solve (idx*x_1 + A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
                A(:, 2)*x2(k) + A(:, 3)*x3(k)) == 0, x_1);
        if order (2)==2
            x2(k+1) = solve (idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
                    A(:, 2)*x_2 + A(:, 3)*x3(k)) == 0, x_2);

            x3(k+1) = solve (idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
                A(:, 2)*x2(k+1) + A(:, 3)*x_3) == 0, x_3);

        else
            x3(k+1) = solve (idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
                A(:, 2)*x2(k) + A(:, 3)*x_3) == 0, x_3);

            x2(k+1) = solve (idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
                    A(:, 2)*x_2 + A(:, 3)*x3(k+1)) == 0, x_2);

        end


    elseif order (1) == 2

        x2(k+1) = solve (idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k) + ...
```

```
                        A(:, 2)*x_2 + A(:, 3)*x3(k)) == 0, x_2);


        if order(2) == 1
            x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
                        A(:, 2)*x2(k+1) + A(:, 3)*x3(k)) == 0, x_1);


            x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
                        A(:, 2)*x2(k+1) + A(:, 3)*x_3) == 0, x_3);



        else
            x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k) + ...
                        A(:, 2)*x2(k+1) + A(:, 3)*x_3) == 0, x_3);

            x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
                        A(:, 2)*x2(k+1) + A(:, 3)*x3(k+1)) == 0, x_1);

        end

    else

        x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k) + ...
                        A(:, 2)*x2(k) + A(:, 3)*x_3) == 0, x_3);

        if order(2) == 1
            x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
                            A(:, 2)*x2(k) + A(:, 3)*x3(k+1)) == 0, x_1);

                        x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
                            A(:, 2)*x_2 + A(:, 3)*x3(k+1)) == 0, x_2);

        else

            x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k) + ...
                            A(:, 2)*x_2 + A(:, 3)*x3(k+1)) == 0, x_2);

            x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
                            A(:, 2)*x2(k+1) + A(:, 3)*x3(k+1)) == 0, x_1);
        end
    end

    y(:, k+1)=y(:, k)+beta * (A(:, 1)*x1(k+1) + A(:, 2)*x2(k+1) + A(:, 3)*x3(k+1));




    if mod(k,10) == 0
        disp( sprintf ('----- beta: %d ------', beta));
        disp( sprintf ('------- Iteration: %d ----------', k));
        disp('      x1         x2         x3')
        disp([  x1(k), x2(k), x3(k)])

    end

end

if idx == 0
    add = ' without objective function';
else
    add = ' with objective function';
end

figure()
subplot(3,1,1)
plot(1:k+1, x1, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Permuation based estimate of x_1 with beta=', num2str(beta), add))

subplot(3,1,2)
plot(1:k+1, x2, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Permuation based estimate of x_1 with beta=', num2str(beta), add))

subplot(3,1,3)
plot(1:k+1, x3, '*', 1:k+1, zeros(1, k+1), '-')
title( strcat('Permuation based estimate of x_1 with beta=', num2str(beta), add))
hold off
```

PermADMM.m

```
clear all
close all

i;
```

```
npoints = 10000;
x = generate_x_points (npoints);
s = generate_s_points (npoints);


p = [6,12];
pname = {'six', 'twelve'};
cutoffs = [0,-10];
cname = {'zero', 'negTen'};



Values = zeros(4, npoints, 4);


k = 1;
for i=1:size(p,2)
    for j=1:size(cutoffs,2)
        [out, point] = evaluate_candidacy(x,s,p(i), cutoffs(j));
        Values(1,:, k) = out;
        Values(2:4,1: size(point,2),k) = point;
        k = k + 1 ;
    end



end

%% p = 6

p6 = Values(:,:, 1:2);
outputs6 = squeeze(p6(1,:,:) );
points6 = p6(2:4,:,:) ;

figure()
scatter3(points6(1, points6(1,:,1) > 0,1), points6(2, points6(2,:,1) > 0,1), points6(3, points6(3,:,1) > 0,1),'r')
hold on
scatter3(points6(1, points6(1,:,2) > 0,2), points6(2, points6(2,:,2) > 0,2), points6(3, points6(3,:,2) > 0,2),'b')
legend('Less Than Zero', 'Less Than -10')
title('$\psi_{6}(\mathbf{x},\mathbf{s})$','Interpreter', 'LaTex', ' fontsize ', 18)
xlabel('x_1'); ylabel('x_2'); zlabel('x_3')

%
p12 = Values(:,:, 3:4);
outputs12 = squeeze(p12(1,:,:) );
points12 = p12(2:4,:,:) ;



figure()
scatter3(points12(1, points12(1,:,1) > 0,1), points12(2, points12(2,:,1) > 0,1), points12(3, points12(3,:,1) > 0,1),'r')
hold on
scatter3(points12(1, points12(1,:,2) > 0,2), points12(2, points12(2,:,2) > 0,2), points12(3, points12(3,:,2) > 0,2),'b')
legend('Less Than Zero', 'Less Than -10')
title('$\psi_{12}(\mathbf{x},\mathbf{s})$','Interpreter', 'LaTex',' fontsize ', 18)
xlabel('x_1'); ylabel('x_2'); zlabel('x_3')

%%

clear all
close all

i;
npoints = 10000;
x = generate_x_points (npoints);
s = generate_s_points_two (npoints);


p = [6,12];
pname = {'six', 'twelve'};
cutoffs = [0,-45];
cname = {'zero', 'negTen'};



Values = zeros(4, npoints, 4);


k = 1;
for i=1:size(p,2)
    for j=1:size(cutoffs,2)
        [out, point] = evaluate_candidacy(x,s,p(i), cutoffs(j));
        Values(1,:, k) = out;
        Values(2:4,1: size(point,2),k) = point;
        k = k + 1 ;
    end

```

```
end

%% p = 6

p6 = Values (:,:,  1:2);
outputs6 = squeeze(p6 (1,:,:) );
points6 = p6 (2:4,:,:) ;

figure ()
scatter3 ( points6 (1, points6 (1,:,1)  > 0,1), points6 (2,  points6 (2,:,1) > 0,1),  points6 (3, points6 (3,:,1)  > 0,1),'r')
hold on
scatter3 ( points6 (1, points6 (1,:,2)  > 0,2), points6 (2,  points6 (2,:,2) > 0,2),  points6 (3, points6 (3,:,2)  > 0,2),'b')
legend('Less Than Zero', 'Less Than -10')
title ('$\psi_{6}(\mathbf{x},\mathbf{s})$','Interpreter', 'LaTex', ' fontsize ', 18)
xlabel ('x_1');  ylabel ('x_2');  zlabel ('x_3')

%
p12 = Values (:,:,  3:4);
outputs12 = squeeze(p12 (1,:,:) );
points12 = p12 (2:4,:,:) ;



figure ()
scatter3 ( points12 (1, points12 (1,:,1)  > 0,1), points12 (2,  points12 (2,:,1) > 0,1),  points12 (3, points12 (3,:,1)  > 0,1),'r')
hold on
scatter3 ( points12 (1, points12 (1,:,2)  > 0,2), points12 (2,  points12 (2,:,2) > 0,2),  points12 (3, points12 (3,:,2)  > 0,2),'b')
legend('Less Than Zero', 'Less Than -10')
title ('$\psi_{12}(\mathbf{x},\mathbf{s})$','Interpreter', 'LaTex',' fontsize ', 18)
xlabel ('x_1');  ylabel ('x_2');  zlabel ('x_3')
```

Problem3_Client.m

```
function alpha_new = choose_alpha(alpha, d,x, x0, a,b,f)
% finds appropriate alpha for next step
while f(x, x0, alpha, b) <= f(x + alpha * d(1:end-1), x0 + alpha*d(end), a, b)
    alpha = alpha / 2;

end
alpha_new = alpha;

end
```

choose_alpha.m

```
function [ spoints ] = generate_s_points ( npoints )
    y = - rand(1,npoints);
    spoints = vertcat (1 - y, 1- y, -y);

end
```

generate_s_points.m

```
function [x_samples] = generate_x_points ( npoints )
        x1 = linspace (0.0001, .9999, npoints);
        x2 = rand( size (x1)).*( 1 - x1);
        x3 = 1 - x1 - x2;

        x_samples = vertcat (x1,x2,x3);


end
```

generate_x_points.m