

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

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

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

$$\begin{aligned}\nabla f(x) &= x^2 + 1 \\ \nabla^2 f(x) &= 2x\end{aligned}$$

Then the SOLC condition is

$$\begin{aligned}\|(x+d)^2 - x^2 - 2xd\| &\leq \beta \|d\|^2 \\ \|d^2\| &\leq \beta \|d\|^2 \\ 1 &\leq \beta\end{aligned}$$

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

$$\begin{aligned}\|xd^2\| &\leq 2\beta \|2xd^2\| \\ \frac{1}{2} &\leq \beta\end{aligned}$$

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

We have

$$\begin{aligned}\nabla f(x) &= \frac{1}{x} \\ \nabla^2 f(x) &= -\frac{1}{x^2}\end{aligned}$$

SOLC condition

$$\left\| \frac{1}{x+d} - \frac{1}{x} + \frac{1}{x^2} \right\| \leq \beta \|d\|$$

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

$$\left\| \frac{d^2}{x^2(x+d)} \right\| \leq \beta \|d\|$$

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

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

$$\left\| \frac{xd^2}{x^2(x+d)} \right\| \leq \beta \left\| \frac{d^2}{x^2} \right\|$$

$$\left\| \frac{x}{(x+d)} \right\| \leq \beta$$

$$\left\| \frac{1}{1+x^{-1}d} \right\| \leq \beta$$

$$\leq \frac{1}{1-\|x^{-1}d\|} \leq \beta$$

$$\lim_{\|x^{-1}d\| \rightarrow 0} \frac{1}{1-\|x^{-1}d\|} = 1$$

$$\lim_{\|x^{-1}d\| \rightarrow \frac{1}{2}} \frac{1}{1-\|x^{-1}d\|} = 2$$

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

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

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

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

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

$$\begin{aligned} & \left\| \frac{-x}{1 + e^{-x-d}} + \frac{x}{1 + e^{-x}} - \frac{dx}{(1 + e^{-x})^2} \right\| \leq \beta \left| \frac{-d}{x^2} \right| \\ & \left\| \frac{x^3 e^{-x} (e^{-d} - 1)(1 + e^{-x}) - dx^3 (1 + e^{-x-d})}{d^2 (1 + e^{-x-d})(1 + e^{-x})^2} \right\| \leq \beta \\ & \lim_{x \rightarrow \infty, d \neq 0} \left| \frac{x^3 e^{-x} (e^{-d} - 1)(1 + e^{-x}) - dx^3 (1 + e^{-x-d})}{d^2 (1 + e^{-x-d})(1 + e^{-x})^2} \right| = \infty \\ & \lim_{d \rightarrow 0} \left| \frac{x^3 e^{-x} (e^{-d} - 1)(1 + e^{-x}) - dx^3 (1 + e^{-x-d})}{d^2 (1 + e^{-x-d})(1 + e^{-x})^2} \right| = \infty \end{aligned}$$

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

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

Or to minimize the log-logistic-loss

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

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

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

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

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

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

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

$$= \sum_{i,c_i=1} a_{ij} a_{ik} \frac{\exp[\mathbf{z}]}{(1 + \exp[\mathbf{z}])^2} + \sum_{i,c_i=-1} a_{ij} a_{ik} \frac{\exp[\bar{\mathbf{z}}]}{(1 + \exp[\bar{\mathbf{z}}])^2}$$

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

$$\nabla_{x_0, x_0} = \sum_{i,c_i=1} \frac{\exp[\mathbf{z}]}{(1 + \exp[\mathbf{z}])^2} + \sum_{i,c_i=-1} \frac{\exp[\bar{\mathbf{z}}]}{(1 + \exp[\bar{\mathbf{z}}])^2}$$

Thus the $i, j, (i, j) \in \{0, \dots, n\}$ 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

$$\min_x f(x) = x_1 + x_2$$

$$\text{Such that } :x_1 + x_2 + x_3 = 1$$

$$(x_1, x_2, x_3) \geq 0$$

(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

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

Differentiating with respect to x_1, x_2 we have

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

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

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

$$\nabla_x \rightarrow 1 - \frac{\mu}{x} + \frac{\mu}{1 - 2x} = 0$$

$$x(1 - 2x) - \mu(1 - 2x) + \mu x = 0$$

$$2x^2 - x(3\mu + 1) + \mu = 0$$

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

Now noting that as $\lim_{\mu \rightarrow \infty}$ 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, let's take the limit as $\mu \rightarrow \infty$

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

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

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

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

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

and

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

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

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

Differentiating with respect to x_1, x_2 we have

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

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

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

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 problems 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 $0 < \mu' < \mu$

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

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

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

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

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

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

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

Adding together these two equations

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

Thus

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

Therefore we have that

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

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

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

And thus that:

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

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

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

Adding the two equation, we have

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

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

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

Continuing we have that

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

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 $\mathbf{D} \in \mathbb{S}^n$ 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 $\text{trace}(\mathbf{D}) = \sum_j \lambda_j$ where λ_j is the j th eigenvalue of the matrix \mathbf{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

$$\begin{aligned} \text{trace}(\mathbf{D}) &= \sum_j \lambda_j \\ \log[\det(I + \mathbf{D})] &= \log\left[\prod_j (1 + \lambda_j)\right] = \sum_j \log(1 + \lambda_j) \end{aligned}$$

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

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

Problem 6

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

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

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

$$\begin{aligned} \min_z \quad & z \\ \text{such that :} \quad & Ax + b \preceq z\mathbf{1} \end{aligned}$$

The dual is then

$$\begin{aligned} \max_y \quad & \mathbf{b}^T y \\ \text{such that :} \quad & A^T y = 0 \\ & \mathbf{1}^T y = 1 \\ & y \succeq 0 \end{aligned}$$

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

$$\min_x \log \left[\sum_{i=1}^m \exp(\mathbf{a}_i^T \mathbf{x} + b_i) \right]$$

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

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

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

$$\max_z \quad b^T z - \sum_j z_j \log[z_j]$$

Such that:

$$\begin{aligned} A^T z &= 0 \\ \mathbf{1}^T z &= 1 \\ z &\succeq 0 \end{aligned}$$

In this case we have that z^* is also feasible for the dual of the piecewise 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

$$\sum_j z_j \log\left[\frac{1}{z_j}\right] \leq \log\left[\sum_j 1\right] = \log m$$

Thus we have that

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

Furthermore it holds that

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

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

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

$$\log\left[\sum_i \exp(r_i)\right] \geq m$$

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

$$\begin{aligned}z_2 - \log(m) &\leq z_1 \leq z_2 \\ 0 &\leq z_2 - z_1 \leq \log(m)\end{aligned}$$

and the proof is complete.

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

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

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 $\gamma \rightarrow \infty$.

This problem can be reformulated as:

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

Such that:

$$Ax + b = y$$

Forming the Lagrangian we have

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

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:

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

Thus we see that if $\lambda_k < 0$, setting $y_k = c, y_i = 0 \forall i \neq k$ then $\lim_{c \rightarrow -\infty}$ 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

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

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

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:

$$\max_{\lambda} \quad b^T \lambda - \frac{1}{\gamma} \sum_i \lambda_i \log[\lambda_i]$$

such that: $A^T \lambda = 0$

$$\mathbf{1}^T \lambda = 1$$

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 $\gamma \rightarrow \infty$, then $z_3 \rightarrow z_1$.

Code

```
1 function [ ] = ADMM(idx, beta, iter, A)
2
3 x1(1) = rand(1);
4 x2(1) = rand(1);
5 x3(1) = rand(1);
6 y(:, 1) = rand(3,1);
7 syms x_1 x_2 x_3
```

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

ADMM.m

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

```

9 for k=1:iter
11     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);
13     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);
15     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);
17
19     y(:, k+1)=y(:, k)+beta * (A(:, 1)*x1(k+1) + A(:, 2)*x2(k+1) + A(:, 3)*x3(k+1));
21
23
25     if mod(k,10) == 0
26         disp( sprintf('----- beta: %d -----', beta));
27         disp( sprintf('----- Iteration: %d -----', k));
28         disp('      x1      x2      x3')
29         disp([ x1(k), x2(k), x3(k)])
31     end
33 end
35 if idx == 0
36     add = ' without objective function';
37 else
38     add = ' with objective function';
39 end
41 figure ()
42 subplot (3,1,1)
43 plot (1:k+1, x1, '*-', 1:k+1, zeros (1, k+1), '-')
44 title ( strcat ('Estimate of x_1 for beta=', num2str(beta), 'and ', add))
45 subplot (3,1,2)
46 plot (1:k+1, x2, '*-', 1:k+1, zeros (1, k+1), '-')
47 title ( strcat ('Estimate of x_2 for beta=', num2str(beta), 'and ', add))
48 subplot (3,1,3)
49 plot (1:k+1, x3, '*-', 1:k+1, zeros (1, k+1), '-')
50 title ( strcat ('Estimate of x_3 for beta=', num2str(beta), 'and ', add))
51 hold off
53 end

```

```

1 function [x, x0, iter, fvals, gvals, hvals] = Newton(f, df, hessian, x_initial, x0_intial, a, b, ALPHA, MAX_ITER, TOL, debug)
3
5     x = x_initial;
6     x0 = x0_intial;
7
9     x_cat_prev = [x; x0];
10
12     H_prev = inv(hessian(x, x0, a, b));
13
15     iter = 1;
16     fvals = [];
17     gvals = [];
18     hvals = [];
19     fvals(iter) = f(x, x0, a, b);
20     gvals(iter) = norm(df(x, x0, a, b));
21     hvals(iter) = norm(inv(H_prev));
22
24     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);
26
28     if debug
29         disp( sprintf('----- Iteration: %d -----', iter));
30         disp('      x_1      x_2      x_0      f(x)      delta_F      delta_x      lambda      alpha')
31         disp([ x(1), x(2), x0, fvals(iter), NaN, NaN, 0, NaN])
32     end
33
35     norm_grad = 10000;
36     delta_f = 10000;
37     delta_x = 10000;

```

```

728 35 alpha = ALPHA;
729
730 37
731 39 while iter < MAX_ITER
732 41     if abs(norm_grad) < TOL
733         disp( sprintf('-----GRADIENT NORM IS BELOW TOLERANCE CONVERGENCE OF FUNCTION AFTER %d ITERATIONS-----', iter))
734         disp( sprintf('-----FINAL Iteration: %d-----', iter));
735         disp('      x_1      x_2      x_0      f(x)      delta_F      delta_x      alpha      NORMGRAD      HessianNotm')
736         disp([ x(1), x(2), x0, fvals(iter), delta_f, delta_x, alpha, gvals(iter) hvals(iter)])
737         break;
738     end
739     if delta_x < 1e-8
740         iter = iter + 1;
741         disp( sprintf('CHANGE IN X IS TINY, CONVERGENCE OF FUNCTION AFTER %d ITERATIONS', iter))
742         disp( sprintf('-----FINAL Iteration: %d-----', iter));
743         disp('      x_1      x_2      x_0      f(x)      delta_F      delta_x      alpha      NORMGRAD      HessianNotm')
744         disp([ x(1), x(2), x0, fvals(iter), delta_f, delta_x, alpha, gvals(iter) hvals(iter)])
745         break;
746     end
747     iter = iter + 1;
748
749     g_prev = df(x,x0,a,b);
750     d = H_prev*g_prev;
751     alpha = choose_alpha(ALPHA, -d, x, x0, a,b,f)
752     x_cat_new = x_cat_prev - ALPHA*d;
753
754     x = x_cat_new(1:end-1);
755     x0 = x_cat_new(end);
756     g_new = df(x,x0,a,b);
757
758     fvals(iter) = f(x,x0, a, b);
759     gvals(iter) = norm(g_new);
760
761     q = g_new - g_prev;
762     p = - ALPHA*d;
763
764     H_new = H_prev + (p*p')/(p'*q) - (H_prev*(q*q')*H_prev)/(q'*H_prev*q);
765     hvals(iter) = norm(inv(H_new));
766
767     delta_f = fvals(iter) - fvals(iter - 1);
768     delta_x = norm(x_cat_new - x_cat_prev, 2);
769     norm_grad = norm(df(x,x0,a,b),2);
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

DFP.m

```

779 1 function [x, x0, iter, fvals, gvals, hvals] = Newton(f, df, hessian, x_initial, x0_intial, a, b, ALPHA, MAX_ITER, TOL, debug)

```

```

780 3
781 x = x_initial ;
782 x0 = x0_initial ;
783
784 x_cat_prev = [x; x0];
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831

```

```

3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85

```

```

x = x_initial ;
x0 = x0_initial ;

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

```



```

832 figure ();
833 subplot (3,1,1)
834 plot (1: iter , fvals , 'LineWidth',2); grid on;
835 title (' Objective Function Value Newton'); xlabel(' Iteration '); ylabel('f(x)');
836 subplot (3,1,2)
837 plot (1: iter , gvals , 'LineWidth',2); grid on;
838 title ('Norm of Gradient of Objective Function Newton'); xlabel(' Iteration '); ylabel('g(x)');
839 subplot (3,1,3)
840 plot (1: iter , hvals , 'LineWidth',2); grid on;
841 title ('Norm of Hessian of Objective Function Newton'); xlabel(' Iteration '); ylabel('F(x)');
842 end

```

Newton.m

```

845 1 clear all
846 2 close all
847 3
848 4 Hw3_path = '/Users/jacobperricone/Desktop/STANFORD/w16/git_cme307/HW3'
849 5 addpath(genpath(Hw3_path))
850 6
851 7 %% Problem 5
852 8 a = [0, 1, 0; 0 0, 1];
853 9 b = [0,-1, 0; 0, 0,-1];
854 10 mu = [0, 10^-5];
855 11
856 12 x_initial = [.5;1.25];
857 13 x0_initial = .124;
858 14 ALPHA = .05;
859 15 MAX_ITER = 25090;
860 16 TOL = .000001;
861 17
862 18 f = @(x,x0, a, b) sum(log(1 + exp(-a'*x - x0))) + sum(log(1 + exp(b'*x + x0)));
863 19 f2 = @(x,x0, a, b, mu) sum(log(1 + exp(-a'*x - x0))) + sum(log(1 + exp(b'*x + x0)));
864 20 df = @(x,x0, a, b) ...
865 21 [sum((-a(1,:)' .* exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
866 22 + sum((b(1,:) ' .* exp(b'*x + x0)) / (1 + exp(b'*x + x0))); ...
867 23 sum((-a(2,:)' .* exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
868 24 + sum((b(2,:) ' .* exp(b'*x + x0)) / (1 + exp(b'*x + x0))); ...
869 25 sum((-exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
870 26 + sum((exp(b'*x + x0)) / (1 + exp(b'*x + x0))];
871 27
872 28 df2 = @(x,x0, a, b,mu) ...
873 29 [sum((-a(1,:)' .* exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
874 30 + sum((b(1,:) ' .* exp(b'*x + x0)) / (1 + exp(b'*x + x0))); ...
875 31 sum((-a(2,:)' .* exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
876 32 + sum((b(2,:) ' .* exp(b'*x + x0)) / (1 + exp(b'*x + x0))); ...
877 33 sum((-exp(-a'*x - x0)) / (1 + exp(-a'*x - x0))) ...
878 34 + sum((exp(b'*x + x0)) / (1 + exp(b'*x + x0))];
879 35
880 36 %%
881 37 newa = vertcat (a,ones(1, size(a,2)));
882 38 tmpa = zeros (size(a,2) , size(a,1)*(size(a,2) + 1));
883 39 tmpa(1,1:3) = newa(:,1)';
884 40 tmpa(2,4:6) = newa(:,2)';
885 41 tmpa(3,7:9) = newa(:,3)';
886 42 tmpa = reshape(newa*tmpa,3,3,size(a,2));
887 43
888 44 newb = vertcat (b,ones(1, size(b,2)));
889 45 tmpb = zeros (size(b,2) , size(b,1)*(size(b,2) + 1));
890 46 tmpb(1,1:3) = newb(:,1)';
891 47 tmpb(2,4:6) = newb(:,2)';
892 48 tmpb(3,7:9) = newb(:,3)';
893 49 tmpb = reshape(newb*tmpb,3,3,size(b,2));
894 50
895 51 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));
896 52 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));
897 53
898 54 hessian = @(x,x0,a,b)(sum(z(x,x0,a) .* tmpa,3) + sum(zbar(x,x0,b) .* tmpb,3));
899 55
900 56 %%
901 57
902 58
903 59
904 60 [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);
905 61 [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);
906 62 [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);
907 63
908 64 [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);
909 65

```

```

884 [x_CGD, x0_CGD, iter_CGD, fvals_CGD, gvals_CGD] = CGD(f2, df2, x_initial, x0_initial, a, b, MAX_ITER, TOL, 0, 0);
885
886 [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);
887 %% Generate random data
888 figure()
889 subplot(2,1,1)
890 loglog(1:iter_N, fvals_N)
891 hold on
892 loglog(1:iter_DFP, fvals_DFP)
893 hold on
894 loglog(1:iter_SDM, fvals_SDM)
895 hold on
896 loglog(1:iter_ASDM, fvals_ASDM)
897 hold on
898 loglog(1:iter_CGD, fvals_CGD)
899 hold on
900 loglog(1:iter_BB, fvals_BB)
901 legend('Newton', 'DFP', 'SDM', 'ASDM', 'CGD', 'BB')
902 title('Function Value vs. Iteration (LogSpace)')
903 xlabel('$\log[\mbox{Iteration}]$', 'Interpreter', 'LaTeX')
904 ylabel('$\log[f(x)]$', 'Interpreter', 'LaTeX')
905
906 subplot(2,1,2)
907 loglog(1:iter_N, gvals_N)
908 hold on
909 loglog(1:iter_DFP, gvals_DFP)
910 hold on
911 loglog(1:iter_SDM, gvals_SDM)
912 hold on
913 loglog(1:iter_ASDM, gvals_ASDM)
914 hold on
915 loglog(1:iter_CGD, gvals_CGD)
916 hold on
917 loglog(1:iter_BB, gvals_BB)
918 legend('Newton', 'DFP', 'SDM', 'ASDM', 'CGD', 'BB')
919 title('Norm Gradient Value vs. Iteration (LogSpace)')
920 xlabel('$\log[\mbox{Iteration}]$', 'Interpreter', 'LaTeX')
921 ylabel('$\log[g(x)]$', 'Interpreter', 'LaTeX')
922
923 %%
924 figure()
925 subplot(3,1,1)
926 loglog(1:iter_N, fvals_N)
927 hold on
928 loglog(1:iter_DFP, fvals_DFP)
929 hold on
930 legend('Newton', 'DFP')
931 title('Function Value vs. Iteration (LogSpace)')
932 xlabel('$\log[\mbox{Iteration}]$', 'Interpreter', 'LaTeX')
933 ylabel('$\log[f(x)]$', 'Interpreter', 'LaTeX')
934
935 subplot(3,1,2)
936 loglog(1:iter_N, gvals_N)
937 hold on
938 loglog(1:iter_DFP, gvals_DFP)
939 hold on
940 legend('Newton', 'DFP')
941 title('Norm Gradient Value vs. Iteration (LogSpace)')
942 xlabel('$\log[\mbox{Iteration}]$', 'Interpreter', 'LaTeX')
943 ylabel('$\log[g(x)]$', 'Interpreter', 'LaTeX')
944
945 subplot(3,1,3)
946 loglog(1:iter_N, hvals_N)
947 hold on
948 loglog(1:iter_DFP, hvals_DFP)
949 hold on
950 legend('Newton', 'DFP')
951 title('Norm Hessian Value vs. Iteration (LogSpace)')
952 xlabel('$\log[\mbox{Iteration}]$', 'Interpreter', 'LaTeX')
953 ylabel('$\log[H(x)]$', 'Interpreter', 'LaTeX')

```

Problem2_Client.m

```

927
928
929
930 1 clear all
931   close all
932
933 %% Problem 7
934 % Part (a) and (b)
935
936 A = [1, 1, 1; 1, 1, 2; 1, 2, 2];
937 beta = [1, 1, 10];
938
939 for i=1:3
940     % Without the objective function.
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

936     ADMM(0, beta(i), 100, A)
937 13
938     % With the objective function .
939 15     ADMM(1, beta(i), 100, A)
940 end
941 17
942 19
943 21 %% Random Permutation
944 % Part (c)
945 23
946 for i=1:3
947 25
948     % Without the objective function and with permutation.
949 27     PermADMM(0, beta(i), 100, A)
950
951     % With objective function and permutation.
952 29     PermADMM(1, beta(i), 100, A)
953 31
954 end

```

Problem7.m

```

951 function [out, points] = evaluate_candidity(x, s, p, cutoff)
952 2
953     out = p*diag(log(x'*s))' - sum(log(x.*s),1);
954 4     points = x(:, out < cutoff);
955 6
956 end

```

evaluate_candidity.m

```

958 function [spoints] = generate_s_points_two(npoints)
959 1
960     y = rand(1, npoints);
961 3     spoints = vertcat(1 + y, y, y);
962 5
963 end

```

generate_s_points_two.m

```

964 function [Z] = CVXSolZ(a, A, d, idx)
965 2
966 %UNTITLED4 Summary of this function goes here
967 % Detailed explanation goes here
968 C = 10*rand(4,4);
969 cvx_expert true
970 cvx_begin sdp quiet
971     variable Z(4,4) symmetric
972     minimize(sum(dot(C, Z)) + idx(1)*trace(Z) + ...
973         idx(2)*power(2,norm(Z(3,1:2)' - a(1:2, 3)) + norm(Z(4, 1:2)' - a(1:2, 1))))
974     %idx(2)*power(2,sqrt(power(2,Z(3,1)-a(1,3))+power(2,Z(3,2)-a(2,3)))) + ...
975     %sqrt(power(2,Z(4,1)-a(1,1))+power(2,Z(4,2)-a(2,1))))
976     subject to
977         sum(dot(A(:,1)*A(:,1)', Z)) == 1;
978         sum(dot(A(:,2)*A(:,2)', Z)) == 1;
979         sum(dot(A(:,3)*A(:,3)', Z)) == 2;
980
981         for i = 1:2
982             sum(dot([a(:, i); -1; 0] * [a(:, i); -1; 0]', Z)) ...
983                 == d(i)^2;
984             sum(dot([a(:, i+1); 0; -1] * [a(:, i+1); 0; -1]', Z)) ...
985                 == d(i+2)^2;
986         end
987
988         sum(dot([0; 0; 1; -1] * [0; 0; 1; -1]', Z)) == d(5)^2 ;
989
990         Z >= 0;
991     cvx_end
992 32
993 end

```

CVXSolZ.m

```

984
985
986 1 clear all
987 3 close all
988 %%
989 a = [1, -1, 0; 0, 0, 2];

```

```

988 5 idx =[1, 0; -1, 0; 0, 1; 0, -1];
989
990 7 DMethods.error_C = [];
991   DMethods.error_V = [];
992   for i = 1:3
993       %
994       Methods.error_C = [];
995       Methods.error_V = [];
996       [X_1, Y_1] = meshgrid(linspace(-1.5, 1.5, 20)', linspace(-.5, 2.5, 20)');
997
998       Cx1_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
999       Cx2_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
1000       Ctotal_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
1001
1002       Vx1_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
1003       Vx2_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
1004       Vtotal_error_SDP = zeros(size(X_1, 1), size(Y_1, 1));
1005
1006       A1 = [1; 0; 0; 0]; A2 = [0; 1; 0; 0]; A3 = [1; 1; 0; 0];
1007       A = [A1, A2, A3];
1008
1009       for j=1:size(X_1, 1)
1010           for l=1:size(Y_1, 1)
1011               x1_center = [0, 1];
1012               dist_x1_center = pdist2(x1_center, a(:, 1:2)');
1013
1014               x1_vertex = [.95, .05];
1015               dist_x1_vertex = pdist2(x1_vertex, a(:, 1:2)');
1016
1017               x2 = [X_1(l, j), Y_1(l, j)];
1018               dist_x2 = pdist2(x2, a(:, 2:3)');
1019
1020               dhat_center = pdist2(x1_center, x2);
1021               dhat_vertex = pdist2(x1_vertex, x2);
1022
1023               d = [dist_x1_center(1), dist_x1_center(2), dist_x2(1), ...
1024                   dist_x2(2), dhat_center];
1025
1026               % Minimize the trace of Z
1027               Z = CVXSolZ(a, A, d, idx(i, :));
1028
1029               x_1 = [Z(3, 1), Z(3, 2)]';
1030               x_2 = [Z(4, 1), Z(4, 2)]';
1031
1032               Cx1_error_SDP(j, l) = norm(x_1 - x1_center);
1033               Cx2_error_SDP(j, l) = norm(x_2 - x2);
1034               Ctotal_error_SDP(j, l) = Cx1_error_SDP(j, l) + Cx2_error_SDP(j, l);
1035               Cdistance(j, l) = dhat_center;
1036
1037               Methods_error_C = [Methods_error_C, Cx1_error_SDP(j, l) + Cx2_error_SDP(j, l)];
1038
1039               d = [dist_x1_vertex(1), dist_x1_vertex(2), dist_x2(1), ...
1040                   dist_x2(2), dhat_vertex];
1041
1042               % Minimize the trace of Z
1043               Z = CVXSolZ(a, A, d, idx(i, :));
1044
1045               x_1 = [Z(3, 1), Z(3, 2)]';
1046               x_2 = [Z(4, 1), Z(4, 2)]';
1047
1048               if isnan(x_1(1)) || isnan(x_1(2))
1049                   disp('YOOx1')
1050                   x_1;
1051               end
1052
1053               if isnan(x_1(1)) || isnan(x_1(2))
1054                   disp('YOOx2')
1055                   x_2;
1056               end
1057
1058               Vx1_error_SDP(j, l) = norm(x_1 - x1_vertex);
1059               Vx2_error_SDP(j, l) = norm(x_2 - x2);
1060               Vtotal_error_SDP(j, l) = Vx1_error_SDP(j, l) + Vx2_error_SDP(j, l);
1061               Vdistance(j, l) = dhat_vertex;
1062
1063               Methods_error_V = [Methods_error_V, Vx1_error_SDP(j, l) + Vx2_error_SDP(j, l)];
1064           end
1065       end
1066   end

```

```

1040 93 end
1041
1042 95 %%
1043 97 idx =[1, 0; -1, 0; 0, 1; 0, -1];
1044 99 col = idx(i,:);
1045 if col(1) == 1 || col(2) == 0
1046     addinfo = ' Objective is to Minimize C\cdot Z + tr(Z).';
1047
1048 elseif col(1) == -1 col(2) == 0
1049     addinfo = ' Objective is to Minimize C\cdot Z - tr(Z).';
1050
1051 elseif col(1) == 0 || col(2) == 1
1052     addinfo = ' Objective is to Minimize C\cdot Z + (d_{13} + d_{21})^2';
1053
1054 else
1055     addinfo = ' Objective is to Minimize C\cdot Z - (d_{13} + d_{21})^2';
1056 end
1057
1058 111 %%
1059 113 figure ()
1060 115 subplot (3,1,1)
1061 117 mesh(X_1, Y_1, Cx1_error_SDP)
1062 119 colormap hsv
1063 121 alpha (.4)
1064 123 colorbar
1065 125 view(-30,30); camlight; axis image
1066
1067 127 title ( strcat ('Magnitude of X_1 Error with X_1 Fixed at (x = 0, y = 1) in Hull', addinfo), 'FontSize', 10)
1068 129 xlabel ('X Data')
1069 131 ylabel ('Y Data')
1070 133 zlabel ('Error')
1071
1072 135 [mx,k] = min(Cx1_error_SDP(:));
1073 137 [ix,jx] = ind2sub( size(Cx1_error_SDP),k);
1074 139 dim = [10 .605 .3 .3];
1075 141 str = strcat ('Minimum Error of (', num2str(Cx1_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1076 143 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1077
1078 145 hold on
1079 147 tmp = a;
1080 149 tmp(:,4) = tmp(:,1);
1081 151 plot3(tmp(1,:), tmp(2,:), zeros( size(tmp(2,:)) ), '—r')
1082 153 plot3(x1_center(1), x1_center(2), 0, '—*b','MarkerSize',10)
1083 155 xlabel ('X Data')
1084 157 ylabel ('Y Data')
1085 159 hold off
1086
1087 161 subplot (3,1,2)
1088 163 surf(X_1, Y_1, Cx2_error_SDP)
1089 165 colormap hsv
1090 167 alpha (.4)
1091 169 colorbar
1092 171 view(-30,30); camlight; axis image
1093
1094 173 title ( strcat ('Magnitude of X_2 Error with X_1 Fixed at (x = 0, y = 1) in Hull', addinfo), 'FontSize', 10)
1095 175 xlabel ('X Data')
1096 177 ylabel ('Y Data')
1097 179 zlabel ('Error')
1098
1099 181 [mx,k] = min(Cx2_error_SDP(:));
1100 183 [ix,jx] = ind2sub( size(Cx2_error_SDP),k);
1101 185 dim = [10 .295 .3 .3];
1102 187 str = strcat ('Minimum Error of (', num2str(Cx2_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1103 189 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1104
1105 191 hold on
1106 193 tmp = a;
1107 195 tmp(:,4) = tmp(:,1);
1108 197 plot3(tmp(1,:), tmp(2,:), zeros( size(tmp(2,:)) ), '—r')
1109 199 plot3(x1_center(1), x1_center(2), 0, '—*b','MarkerSize',10)
1110 201 xlabel ('X Data')
1111 203 ylabel ('Y Data')
1112 205 hold off
1113
1114 207 subplot (3,1,3)
1115 209 surf(X_1, Y_1, Ctotal_error_SDP)
1116 211 colormap hsv
1117 213 alpha (.4)
1118 215 colorbar

```

```

1092 181 view(-30,30); camlight; axis image
1093
1094 183
1095 185 title ( strcat ('Magnitude of Total Error with X_1 Fixed at (x = 0, y = 1) in Hull', addinfo), 'FontSize', 10)
1096 187 xlabel ('X Data')
1097 187 ylabel ('Y Data')
1097 189 zlabel ('Error')
1098 189
1098 191 [mx,k] = min(Ctotal_error_SDP (:));
1099 191 [ix ,jx] = ind2sub( size ( Ctotal_error_SDP ),k);
1099 193 dim = [.10 .009 .3 .3];
1100 193 str = strcat ('Minimum Error of (', num2str(Ctotal_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1100 195 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1101
1102 197 hold on
1102 197 tmp = a;
1103 199 tmp (:,4) = tmp (:,1);
1103 199 plot3 (tmp (1,:), tmp (2,:), zeros ( size (tmp (2,:)) ), '—r')
1104 201 plot3 (x1_center (1), x1_center (2), 0, '—*b',MarkerSize',10)
1104 201 xlabel ('X Data')
1105 203 ylabel ('Y Data')
1105 203 hold off
1106 205 %
1107 205 figure ()
1107 207 subplot (3,1,1)
1108 207 surf (X_1, Y_1, Vx1_error_SDP)
1108 209 colormap hsv
1109 209 alpha (4)
1109 211 colorbar
1110 211 view(-30,30); camlight; axis image
1111
1112 213
1112 215 title ( strcat ('Magnitude of X_1 Error with X_1 Fixed at (x = .95, y = .05) in Hull', addinfo), 'FontSize', 10)
1113 215 xlabel ('X Data')
1113 217 ylabel ('Y Data')
1114 217 zlabel ('Error')
1114
1115 219
1115 221 [mx,k] = min(Vx1_error_SDP (:));
1116 221 [ix ,jx] = ind2sub( size ( Vx1_error_SDP ),k);
1116 223 dim = [.10 .595 .3 .3];
1117 223 str = strcat ('Minimum Error of (', num2str(Vx1_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1118 225 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1118
1119 227 hold on
1119 227 tmp = a;
1120 229 tmp (:,4) = tmp (:,1);
1120 229 plot3 (tmp (1,:), tmp (2,:), zeros ( size (tmp (2,:)) ), '—r')
1121 231 plot3 (x1_vertex (1), x1_vertex (2), 0, '—*b',MarkerSize',10)
1122 231 xlabel ('X Data')
1122 233 ylabel ('Y Data')
1123 233 hold off
1123
1124 235
1124
1125 237 subplot (3,1,2)
1126 237 surf (X_1, Y_1, Vx2_error_SDP)
1126 239 colormap hsv
1127 241 alpha (4)
1127 241 colorbar
1128 243 view(-30,30); camlight; axis image
1129
1129
1130 245 title ( strcat ('Magnitude of X_2 Error with X_1 Fixed at (x = .95, y = .05 ) in Hull', addinfo), 'FontSize', 10)
1131 247 xlabel ('X Data')
1131 247 ylabel ('Y Data')
1132 249 zlabel ('Error')
1132
1133 251
1133 253 [mx,k] = min(Vx2_error_SDP (:));
1134 253 [ix ,jx] = ind2sub( size ( Vx2_error_SDP ),k);
1134 255 dim = [.10 .295 .3 .3];
1135 255 str = strcat ('Minimum Error of (', num2str(Vx2_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1136 257 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1136
1137 257 hold on
1137 257 tmp = a;
1138 259 tmp (:,4) = tmp (:,1);
1138 259 plot3 (tmp (1,:), tmp (2,:), zeros ( size (tmp (2,:)) ), '—r')
1139 261 plot3 (x1_vertex (1), x1_vertex (2), 0, '—*b',MarkerSize',10)
1140 263 xlabel ('X Data')
1140 263 ylabel ('Y Data')
1141 265 hold off
1141
1142 265
1143 267 subplot (3,1,3)
1143 267 surf (X_1, Y_1, Vtotal_error_SDP)

```

```

1144
1145 colormap hsv
1146 alpha (.4)
1147 colorbar
1148 view(-30,30); camlight; axis image
1149
1150 title ( strcat ('Magnitude of Total Error with X_1 Fixed at (x = .95,y = .05) in Hull', addinfo), 'FontSize', 10)
1151 xlabel ('X Data')
1152 ylabel ('Y Data')
1153 zlabel ('Error')
1154 [mx,k] = min(Vtotal_error_SDP (:));
1155 [ix,jx] = ind2sub( size (Vtotal_error_SDP),k);
1156 dim = [ .10 .011 .3 .3];
1157 str = strcat ('Minimum Error of (', num2str(Vtotal_error_SDP(ix,jx)),')', ' at X_2: (', num2str(X_1(ix,jx)), ', ', num2str(Y_1(ix,jx)),')');
1158 annotation ('textbox',dim,'String',str,'FitBoxToText','on', 'FontSize',10);
1159
1160 hold on
1161 tmp = a;
1162 tmp(:,4) = tmp(:,1);
1163 plot3(tmp(1,:), tmp(2,:), zeros( size (tmp(2,:)) ), 'r')
1164 plot3(x1.vertex(1), x1.vertex(2), 0, 'b','MarkerSize',10)
1165 xlabel ('X Data')
1166 ylabel ('Y Data')
1167 hold off
1168
1169 DMethods_error_C = [DMethods_error_C; Methods_error_C];
1170 DMethods_error_V = [DMethods_error_V; Methods_error_V];
1171
1172 end
1173 figure ()
1174 plot (1:length(DMethods_error_C(1, :)), DMethods_error_C(1, :))
1175 hold on
1176 plot (1:length(DMethods_error_C(2, :)), DMethods_error_C(2, :))
1177 hold on
1178 plot (1:length(DMethods_error_C(3, :)), DMethods_error_C(3, :))
1179 title ('Differences Across Methods in Estimating Fixed X = (0, 1)')
1180 legend('With tr(Z)', 'With -tr(Z)', 'With (d_{13} + d_{21})^2')
1181 hold off
1182
1183 figure ()
1184 plot (1:length(DMethods_error_V(1, :)), DMethods_error_V(1, :))
1185 hold on
1186 plot (1:length(DMethods_error_V(2, :)), DMethods_error_V(2, :))
1187 hold on
1188 plot (1:length(DMethods_error_V(3, :)), DMethods_error_V(3, :))
1189 title ('Differences Across Methods in Estimating Fixed X = (.95, .05)')
1190 legend('With tr(Z)', 'With -tr(Z)', 'With (d_{13} + d_{21})^2')
1191

```

Hw4Problem8.m

```

1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195

```

```

1 function [ ] = PermADMM(idx, beta, iter, A)
2
3 x1(1) = rand(1);
4 x2(1) = rand(1);
5 x3(1) = rand(1);
6 y(:, 1) = rand(3,1);
7 syms x_1 x_2 x_3
8
9 for k=1: iter
10     order = randperm(3,3);
11
12     if order(1)==1
13         x1(k+1) = solve(idx*x_1+A(:, 1)'*y(:, k) + beta * A(:, 1)'*(A(:, 1)*x_1 + ...
14             A(:, 2)*x2(k) + A(:, 3)*x3(k)) == 0, x_1);
15         if order(2)==2
16             x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
17                 A(:, 2)*x_2 + A(:, 3)*x3(k)) == 0, x_2);
18
19             x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
20                 A(:, 2)*x2(k+1) + A(:, 3)*x_3) == 0, x_3);
21
22         else
23             x3(k+1) = solve(idx*x_3+A(:, 3)'*y(:, k) + beta * A(:, 3)'*(A(:, 1)*x1(k+1) + ...
24                 A(:, 2)*x2(k) + A(:, 3)*x_3) == 0, x_3);
25
26             x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k+1) + ...
27                 A(:, 2)*x_2 + A(:, 3)*x3(k+1)) == 0, x_2);
28
29         end
30
31     elseif order(1) == 2
32
33         x2(k+1) = solve(idx*x_2+A(:, 2)'*y(:, k) + beta * A(:, 2)'*(A(:, 1)*x1(k) + ...

```



```

1248 5 npoints = 10000;
1249 x = generate_x.points (npoints);
1250 s = generate_s.points (npoints);
1251 9
1252 p = [6,12];
1253 pname = {'six', 'twelve'};
1254 cutoffs = [0,-10];
1255 cname = {'zero', 'negTen'};
1256 15
1257 Values = zeros (4, npoints, 4);
1258 19
1259 k = 1;
1260 for i=1:size (p,2)
1261     for j=1:size (cutoffs ,2)
1262         [out, point] = evaluate_candacy (x,s,p(i), cutoffs (j));
1263         Values (1,., k) = out;
1264         Values (2:4,1: size (point ,2),k) = point;
1265         k = k + 1 ;
1266     end
1267 end
1268 31
1269 %% p = 6
1270 p6 = Values (:, 1:2);
1271 outputs6 = squeeze(p6 (1,:,:) );
1272 points6 = p6 (2:4,:,:) ;
1273 39
1274 figure ()
1275 scatter3 (points6 (1, points6 (1,,:1) > 0,1), points6 (2, points6 (2,,:1) > 0,1), points6 (3, points6 (3,,:1) > 0,1), 'r')
1276 hold on
1277 scatter3 (points6 (1, points6 (1,,:2) > 0,2), points6 (2, points6 (2,,:2) > 0,2), points6 (3, points6 (3,,:2) > 0,2), 'b')
1278 legend('Less Than Zero', 'Less Than -10')
1279 title ('$\psi_{6}(\mathbf{x},\mathbf{s})$', 'Interpreter', 'LaTeX', 'fontsize', 18)
1280 xlabel('x_1'); ylabel('x_2'); zlabel('x_3')
1281 47
1282 %%
1283 p12 = Values (:, 3:4);
1284 outputs12 = squeeze(p12 (1,:,:) );
1285 points12 = p12 (2:4,:,:) ;
1286 51
1287 figure ()
1288 scatter3 (points12 (1, points12 (1,,:1) > 0,1), points12 (2, points12 (2,,:1) > 0,1), points12 (3, points12 (3,,:1) > 0,1), 'r')
1289 hold on
1290 scatter3 (points12 (1, points12 (1,,:2) > 0,2), points12 (2, points12 (2,,:2) > 0,2), points12 (3, points12 (3,,:2) > 0,2), 'b')
1291 legend('Less Than Zero', 'Less Than -10')
1292 title ('$\psi_{12}(\mathbf{x},\mathbf{s})$', 'Interpreter', 'LaTeX', 'fontsize', 18)
1293 xlabel('x_1'); ylabel('x_2'); zlabel('x_3')
1294 61
1295 %%
1296 clear all
1297 close all
1298 67
1299 i;
1300 npoints = 10000;
1301 x = generate_x.points (npoints);
1302 s = generate_s.points_two (npoints);
1303 71
1304 p = [6,12];
1305 pname = {'six', 'twelve'};
1306 cutoffs = [0,-45];
1307 cname = {'zero', 'negTen'};
1308 77
1309 Values = zeros (4, npoints, 4);
1310 81
1311 k = 1;
1312 for i=1:size (p,2)
1313     for j=1:size (cutoffs ,2)
1314         [out, point] = evaluate_candacy (x,s,p(i), cutoffs (j));
1315         Values (1,., k) = out;
1316         Values (2:4,1: size (point ,2),k) = point;
1317         k = k + 1 ;
1318     end
1319 end
1320 91

```

```

1300
1301 end
1302 %% p = 6
1303
1304 p6 = Values ( :, 1:2);
1305 outputs6 = squeeze(p6 (1,:,:) );
1306 points6 = p6 (2:4,:,:) ;
1307
1308 figure ()
1309 scatter3 ( points6 (1, points6 (1,:,1) > 0,1), points6 (2, points6 (2,:,1) > 0,1), points6 (3, points6 (3,:,1) > 0,1), 'r' )
1310 hold on
1311 scatter3 ( points6 (1, points6 (1,:,2) > 0,2), points6 (2, points6 (2,:,2) > 0,2), points6 (3, points6 (3,:,2) > 0,2), 'b' )
1312 legend('Less Than Zero', 'Less Than -10')
1313 title ('$\psi_{6}(\mathbf{x},\mathbf{s})$', 'Interpreter', 'LaTeX', 'fontsize', 18)
1314 xlabel('x_1'); ylabel('x_2'); zlabel('x_3')
1315
1316 %
1317 p12 = Values ( :, 3:4);
1318 outputs12 = squeeze(p12 (1,:,:) );
1319 points12 = p12 (2:4,:,:) ;
1320
1321 figure ()
1322 scatter3 ( points12 (1, points12 (1,:,1) > 0,1), points12 (2, points12 (2,:,1) > 0,1), points12 (3, points12 (3,:,1) > 0,1), 'r' )
1323 hold on
1324 scatter3 ( points12 (1, points12 (1,:,2) > 0,2), points12 (2, points12 (2,:,2) > 0,2), points12 (3, points12 (3,:,2) > 0,2), 'b' )
1325 legend('Less Than Zero', 'Less Than -10')
1326 title ('$\psi_{12}(\mathbf{x},\mathbf{s})$', 'Interpreter', 'LaTeX', 'fontsize', 18)
1327 xlabel('x_1'); ylabel('x_2'); zlabel('x_3')

```

Problem3_Client.m

```

1321
1322 1 function alpha_new = choose_alpha(alpha, d,x, x0, a,b,f)
1323 % finds appropriate alpha for next step
1324 while f(x, x0, alpha, b) <= f(x + alpha * d(1:end-1), x0 + alpha*d(end), a, b)
1325     alpha = alpha / 2;
1326 end
1327 alpha_new = alpha;
1328 end

```

choose_alpha.m

```

1329
1330 1 function [spoints] = generate_s_points (npoints)
1331 y = - rand(1,npoints);
1332 spoints = vertcat (1 - y, 1 - y, -y);
1333 end

```

generate_s_points.m

```

1335
1336 1 function [x_samples] = generate_x_points (npoints)
1337 x1 = linspace (0.0001, .9999, npoints);
1338 x2 = rand(size(x1)).*(1 - x1);
1339 x3 = 1 - x1 - x2;
1340 x_samples = vertcat (x1,x2,x3);
1341 end

```

generate_x_points.m