6.1 (8 points) As part of this homework, you will submit short reviews of the submitted project progress reports. On Canvas, we will assign two reports per student (on or before 5/17). You can find the projects assigned under the same assignment tab on Canvas. Point out the strengths and weaknesses of the proposed approaches. Please try to give constructive feedback.

- Please read the reports in-depth. Summarize the entire report in a short paragraph.
- Please try to give constructive feedback. Point out the strong and weak points. The following are possible points you can comment on: writing style, clarity, technical soundness and experimental evaluation (when applicable).
- You can suggest alternative approaches, e.g., use of different optimization methods, ideas or techniques to try out in that particular problem, any references you know that will be helpful, or any other application areas you find relevant.
- You can ask questions about anything that is missing or not clear in the report and provide comments regarding how to fix the issues. These questions and comments will be addressed in the final report.
- Finally, you will assign a tentative score to each midterm report based on two criteria: clarity/organization and technical content. Please see the grading rubric in the Canvas folder 'Files/Project/Grading', and assign a score from 1 to 5 for each criteria (i.e., two scores per report) with 1 being the lowest and 5 the highest. Provide the justification in your review.

The poster presentations will be on May 27 during class hours.

After completing the reviews, you can submit them through the assignment “Midterm Progress Report” in the assignment section on Canvas.

6.2 (3 points) Consider the convex function $f(x) = \max(x, 0)$ for $x \in \mathbb{R}$.

(a) Describe the subdifferential operator $F(x) = \{(x, \partial f(x)) : x \in \mathbb{R}\}$ and plot its graph.

(b) Find the resolvent operator $(I + \lambda F)^{-1}(x)$ using the graphical method using the graphical approach outlined in the lecture slides and plot its graph.

(c) Find the proximal operator of $f(x)$ directly using its definition and verify that it matches with the resolvent operator in part (b).
6.3 (7 points) **Solving LPs via alternating projections.** Consider an LP in standard form,

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \succeq 0,
\end{align*}
\]

with variable \( x \in \mathbb{R}^n \), and where \( A \in \mathbb{R}^{m \times n} \). A tuple \((x, \nu, \lambda) \in \mathbb{R}^{2n+m}\) is primal-dual optimal if and only if

\[
Ax = b, \quad x \succeq 0, \quad -A^T \nu + \lambda = c, \quad \lambda \succeq 0, \quad c^T x + b^T \nu = 0.
\]

These are the KKT optimality conditions of the LP. The last constraint, which states that the duality gap is zero, can be replaced with an equivalent condition, \( \lambda^T x = 0 \), which is complementary slackness.

(a) (1 point) Let \( z = (x, \nu, \lambda) \) denote the primal-dual variable. Express the optimality conditions as \( z \in A \cap C \), where \( A \) is an affine set, and \( C \) is a simple cone. Give \( A \) as
\[
A = \{ z \mid Fz = g \},
\]
for appropriate \( F \) and \( g \).

(b) (1 point) Explain how to compute the Euclidean projections onto \( A \) and also onto \( C \).

(c) (2 points) Implement alternating projections to solve the standard form LP. Use \( z^{k+1/2} \) to denote the iterate after projection onto \( A \), and \( z^{k+1} \) to denote the iterate after projection onto \( C \). Your implementation should exploit factorization caching in the projection onto \( A \), but you don’t need to worry about exploiting structure in the matrix \( F \).

Test your solver on a problem instance with \( m = 100, n = 500 \). Plot the residual \( \|z^{k+1} - z^{k+1/2}\|_2 \) over 1000 iterations. (This should converge to zero, although perhaps slowly.)

Here is a simple method to generate LP instances that are feasible. First, generate a random vector \( \omega \in \mathbb{R}^n \). Let \( x^* = \max\{\omega, 0\} \) and \( \lambda^* = \max\{-\omega, 0\} \), where the maximum is taken elementwise. Choose \( A \in \mathbb{R}^{m \times n} \) and \( \nu^* \in \mathbb{R}^m \) with random entries, and set \( b = Ax^*, c = -A^T \nu^* + \lambda^* \). This gives you an LP instance with optimal value \( c^T x^* \).

(d) (3 points) Implement Dykstra’s alternating projection method as shown in the [lecture slides](#) and try it on the same problem instances from part (c). Verify that you obtain a speedup, and plot the same residual as in part (c).

6.4 (7 points) **Randomized preconditioners for conjugate gradient methods.** In this question, we explore the use of some randomization methods for solving overdetermined least-squares problems, focusing on conjugate gradient methods. Letting \( A \in \mathbb{R}^{m \times n} \) be a matrix (we assume that \( m \gg n \)) and \( b \in \mathbb{R}^m \), we wish to minimize

\[
f(x) = \frac{1}{2} \|Ax - b\|_2^2 = \frac{1}{2} \sum_{i=1}^m (a_i^T x - b_i)^2,
\]
where the \( a_i \in \mathbb{R}^n \) denote the rows of \( A \).

Given \( m \in \{2^i, i = 1, 2, \ldots \} \), the (unnormalized) Hadamard matrix of order \( m \) is defined recursively as

\[
H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad H_m = \begin{bmatrix} H_{m/2} & H_{m/2} \\ H_{m/2} & -H_{m/2} \end{bmatrix}.
\]

The associated normalized Hadamard matrix is given by \( H_{m}^{(\text{norm})} = H_m / \sqrt{m} \), which evidently satisfies \( H_{m}^{(\text{norm})^T} H_{m}^{(\text{norm})} = I_{m \times m} \). Moreover, via a recursive algorithm it is possible to compute \( H_m x \) in time \( O(m \log m) \), which is much faster than \( m^2 \) for a general matrix.

To solve the least squares minimization problem using conjugate gradients, we must solve \( A^T A x = A^T b \). In class, we discussed that using a preconditioner \( M \) such that \( M \approx A^{-1} \) can give substantial speedup in computing solutions to large problems. Consider the following scheme to generate a randomized preconditioner, assuming that \( m = 2^i \) for some \( i \):

1. Let \( S = \text{diag}(S_{11}, \ldots, S_{mm}) \), where \( S_{jj} \) are random \( \{-1, +1\} \) signs
2. Let \( p \in \mathbb{Z}_+ \) be a small positive integer, say 20 for this problem.
3. Let \( R \in \{0, 1\}^{n+p \times m} \) be a row selection matrix, meaning that each row of \( R \) has only 1 non-zero entry, chosen uniformly at random. (The location of these non-zero columns is distinct.)
4. Define \( \Phi = RH_{m}^{(\text{norm})} S \in \mathbb{R}^{n \times m} \)

We then define the matrix \( M \) via its inverse \( M^{-1} = A^T \Phi^T \Phi A \in \mathbb{R}^{n \times n} \).

(a) (1 point) How many FLOPs (floating point operations) are required to compute the matrices \( M^{-1} \) and \( M \), respectively, assuming that you can compute the matrix-vector product \( H_m v \) in time \( m \log m \) for any vector \( v \in \mathbb{R}^m \)\? (b) (1 point) How many FLOPs are required to naively compute \( A^T A \), assuming \( A \) is dense (using standard matrix algorithms)\? (c) (1 point) How many FLOPs are required to compute \( A^T A v \) for a vector \( v \in \mathbb{R}^n \) by first computing \( u = A v \) and then computing \( A^T u \)? (d) (1 point) Suppose that conjugate gradients runs for \( k \) iterations. Using the preconditioned conjugate gradient algorithm with \( M = (A^T \Phi^T \Phi A)^{-1} \), how many total floating point operations have been performed? How many would be required to directly solve \( A^T A x = A^T b \)? How large must \( k \) be to make the conjugate gradient method slower? 

\(^1\)Hint. To do this in Matlab, generate a random permutation \( \text{inds} = \text{randperm}(m) \), then set \( R = \text{sparse}(1:(n+p), \text{inds}(1:(n+p)), \text{ones}(n+p,1)), n+p, m) \). In Julia, set \( R = \text{sparse}(1:(n+p), \text{inds}[1:(n+p)], \text{ones}(n+p), n+p, m) \).
(e) (3 points) Implement the conjugate gradient algorithm for solving the positive
definite linear system $A^T Ax = A^T b$ both with and without the preconditioner $M$.
To generate data for your problem, set $m = 2^{12}$ and $n = 400$, then generate the
matrix $A$ by setting $A = \text{randn}(m, n) * \text{spdiags(linspace(.001, 100, n))}$
(in Matlab) and $A = \text{randn}(m, n) * \text{spdiagm(linspace(.001, 100, n))}$ (in
Julia), and let $b = \text{randn}(m, 1)$. For simplicity in implementation, you may
directly pass $A^T A$ and $A^T b$ into your conjugate gradient solver, as we only wish
to explore how the methods work. (In Matlab, the \texttt{pcg} method may be useful.)
Plot the norm of the residual $r^k = A^T b - A^T A x^k$ (relative to $\|A^T b\|_2$) as a func-
tion of iteration $k$ for each of your conjugate gradient procedures. Additionally,
compute and print the condition numbers $\kappa(A^T A)$ and $\kappa(M^{1/2} A^T A M^{1/2})$.
Include your code.