6.1 (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 $A^T Ax = A^T b$. 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 $\Phi = RH_m^{(\text{norm})} S \in \mathbb{R}^{n+p \times m}$.

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

1Hint.] 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 $R = \text{sparse}(1:(n+p), \text{inds}[1:(n+p)], \text{ones}(n+p), n+p, m)$.}
(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 na"ively 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?

(e) (3 points) Implement the conjugate gradient algorithm for solving the positive definite linear system $A^T A x = A^T b$ 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 $A = \text{randn}(m, n) \times \text{spdiags(linspace(.001, 100, n))}$ (in Matlab) and $A = \text{randn}(m, n) \times \text{spdiagm(linspace(.001, 100, n))}$ (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 function 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.