Foundations of Computational Math 1 Fall 2022

Programming assignment 3

For **symmetric positive definite** matrix problems, solving $Ax = b$ via

- Proconditioned Steepest Descent (PSD)
- Preconditioned Conjugate Gradient (PCG)

Use problems with known and unknown solutions to provide sufficient evidence of the correctness of your code presented in an appropriate and convincing manner. For problems with a known solution x^*, you can monitor the relative error $\frac{\|x_k - x^*\|}{\|x^*\|}$. For problems with known or unknown solutions monitoring $\frac{\|r_k\|}{\|b\|}$ is a reasonable assessment of how well the current estimate of the solution solves the system.

Routines

- The codes should be able to operate without preconditioning, i.e. with the preconditioner setting to I.
- You may use environments like MATLAB or libraries like LAPACK to generate matrices for testing. **Be sure to cite the libraries/routines or environments/commands used in your solutions**.

Algorithm 1: The preconditioned Steepest Descent method

Data: A, P, b
Result: $\tilde{x} = x_k$
x_0 arbitrary; $r_0 = b - Ax_0$;
Solve $Pz_0 = r_0$;
for $k = 0, 1, \cdots$ *until convergence* **do**
  $\omega_k = Az_k$
  $\alpha_k = \frac{z_k^T r_k}{z_k^T w_k}$
  $x_{k+1} \leftarrow x_k + \alpha_k z_k$
  $r_{k+1} \leftarrow r_k - \alpha_k \omega_k$
  Solve $Pz_{k+1} = r_{k+1}$
end

Algorithm 2: The preconditioned Conjugate Gradient method

Data: A, P, b
Result: $\tilde{x} = x_k$
x_0 arbitrary; $r_0 = b - Ax_0$;
Solve $Pz_0 = r_0$; $p_0 = z_0$
for $k = 0, 1, \cdots$ *until convergence* **do**
  $v_k = Ap_k$
  $\alpha_k = \frac{r_k^T z_k}{p_k^T v_k}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k v_k$
  Solve $Pz_{k+1} = r_{k+1}$
  $\beta_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k}$
  $p_{k+1} = z_{k+1} + \beta_k p_k$
end

Task 1: Influence of the Preconditioners

1.1 Description

Let A be a symmetric positive definite matrix. Implement support in the codes for preconditioning in PSD and PCG using a symmetric positive definite preconditioner defined by the following set:

- No Preconditioner: $P = I$
- Jacobi's preconditioner, i.e., diagonal preconditioning with $P_{Jacobi} = \text{diag}(A)$ where $\text{diag}(A)$ is the diagonal matrix containing the diagonal elements of A.
- The symmetric Gauss-Seidel preconditioner P_{SGS}: If A is symmetric positive definite then so is $P_{SGS} = (D - E)D^{-1}(D - E^T)$ and its Cholesky factorization is easily determined to be $P_{SGS} = (D - E)D^{-\frac{1}{2}}D^{-\frac{1}{2}}(D - E^T) = CC^T$.

1.2 Test requirements:

(i) Design tests that generate A for $n = 10$ and $n = 100$, and compare the performance of the SD and CG with and without preconditioning in terms of the error and residual behaviors. You have seen many ways to generate matrices that can be adapted to create symmetric positive definite matrices. Of particular interest in this problem, is describing how you generate the matrices, what type of behavior you observe/expect for each of the preconditioners, and if there is any correlation between the manner in which you generate the matrices and the performance of the preconditioners, i.e., different generation techniques may produce A with different characteristics that help or hinder the performance of a particular preconditioner.

(ii) If A is symmetric positive definite then P_{Jacobi} is symmetric (diagonal) and positive definite since the diagonal elements of A must be positive. P_{SGS} has a Cholesky factorization when A is symmetric positive definite. So it too is symmetric positive definite.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Preconditioner $P = CC^T$ where C is a lower triangular matrix.

Here are C's for different preconditioners:

- No preconditioner: $C = I$
- Jacobi's preconditioner, i.e., diagonal preconditioning with $P_{Jacobi} = \text{diag}(A)$ where $\text{diag}(A)$ is the diagonal matrix containing the diagonal elements of A.

$$C = \begin{bmatrix} \sqrt{a_{11}} & 0 & \cdots & 0 \\ 0 & \sqrt{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{a_{nn}} \end{bmatrix}$$

- The symmetric Gauss-Seidel preconditioner P_{SGS}. If A is symmetric positive definite then so is $P_{SGS} = (D - E)D^{-1}(D - E^T)$ and its Cholesky factorization is easily determined to be $P_{SGS} = (D - E)D^{-\frac{1}{2}}D^{-\frac{1}{2}}(D - E^T) = CC^T$. You may, of course, also include the use other Richardson's Family preconditioners.

$$C = \begin{bmatrix} \frac{a_{11}}{\sqrt{a_{11}}} & 0 & \cdots & 0 \\ \frac{a_{21}}{\sqrt{a_{11}}} & \frac{a_{22}}{\sqrt{a_{22}}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{\sqrt{a_{11}}} & \frac{a_{n2}}{\sqrt{a_{22}}} & \cdots & \frac{a_{nn}}{\sqrt{a_{nn}}} \end{bmatrix}$$

(iii) In both PSD and PCG, you need to solve $Pz_{k+1} = r_{k+1}$:

- For Jacobi's preconditioner, since P_{Jacobi} is a diagonal matrix, you can solve $Pz_{k+1} = r_{k+1}$ directly with $O(n)$ computational cost.
- For the symmetric Gauss-Seidel (SGS) preconditioner, you may apply forward substitution and backward substitution which are similar to the previous programming assignment. That is, solve $Cy = r_{k+1}$ first and then solve $C^T z_{k+1} = y$.

Task 2: Correctness test

$$A_{test} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix} \text{ and } b_{test} = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}$$

- For PSD and PCG, report the iteration steps k with $P = I, P_{Jacobi}$, and P_{SGS} with stop condition $\|r_k\|_2 = \|b - Ax_k\|_2 < 10^{-6}$.

Task 3: Influence of the Spectrum (Extra credit 20 points)

Let $\Lambda \in \mathbb{R}^{n\times n}$ be a diagonal matrix with positive elements and consider solving systems of the form $\Lambda x = b$. Recall that the behavior of CG and SD depends only on the spectrum in the sense that if $A = Q\Lambda Q^T$ is transformed into the diagonal coordinate system by using the eigenvectors then the solution iterates in the two coordinate systems are related via Q and Q^T and the errors at each step have the same 2-norm, i.e. convergence behavior is identical in both coordinate systems.

1. We have seen several convergence results for SD, CG, (and stationary Richardson's family). Design a set of experiments that choose Λ to influence the behavior of these methods. All methods should be run **without preconditioning**.

2. Clearly identify the convergence theorems that you are testing with each of the sets of experiments, state the predictions of behavior the theorems predict, and relate the observed behavior to those predictions. Make appropriate use of details of the behavior of representative iterations as well as statistics and distributions of errors, residuals, and iteration counts from a larger set of runs.

3. **Test matrices suggestions:** A key issue is a careful and organized characterization of the attributes of the spectrum of A and how you can generate spectra in a systematic manner to test the predictions made by the convergence theory. Here it is not important to do massive amounts of testing but make sure that you clearly describe the experiments and their purpose.

   Several ways to derive test matrices

   - nonsingular matrices from factorizations (possibly structured to impose structure on A)
   - nonsingularity and controlling the spectrum diagonal dominance (strict and irreducible)
   - Gershgorin theorem implications for controlling the spectrum
   - modifications to random matrices using some of the techniques above
   - generating orthogonal matrices and combining with diagonal and other matrices to generate A
   - random matrices with after the fact guarantees of nonsingularity and other constraints

   In addition to these techniques, you can make use of matrix families with known eigenvalues and/or eigenvectors and, more importantly, analyze the spectrum in such a way as to accelerate convergence.

Requirement of Code

- The techniques used to gain an efficient implementation in space and computation must be described clearly and concisely.
- The solution of systems $Pz_k = r_k$ required in each iteration must be done efficiently in $O(n)$ computations using the efficient storage scheme for Jacobi's preconditioners. For SGS preconditioner, you may use a denser storage approach.
- Your code should collect information at each iteration on
  - $\frac{\|e^{(k)}\|_2}{\|x^*\|_2} = \frac{\|x_k - x^*\|_2}{\|x^*\|_2}$
  - $\frac{\|r_k\|_2}{\|b\|_2}$

– **In your report:**

The information collected should be displayed in an appropriate organized manner to support your answers to the questions above. For representative problem instances, when considering the trends involved in the evolution of the values $\frac{\|r_k\|_2}{\|b\|_2}$ and $\frac{\|e^(k)\|_2}{\|x^*\|_2}$, in addition to plotting the quantities vs. the iteration number k, you should plot the logs vs the iteration number k. In addition to representative examples, you should present global summaries using statistics and distributions of errors (histograms), residuals, and iteration counts from larger sets of runs.

Submission of Results

Expected results comprise:

- A document describing your solutions as prescribed in the notes on writing up a programming solution posted on the Canvas.
- The source code, makefiles, and instructions on how to compile and execute your code including the Math Department machine used, if applicable.
- Code documentation should be included in each routine. (You don't need to paste your code in the writing report).
- All text files that do not contain code or makefiles must be PDF files. **Do not send Microsoft word files of any type.**

These results should be submitted by 11:59 PM on the due date. Submission of results is to be done via Canvas.