Graded Homework 4 Foundations of Computational Math 2 Spring 2021

The solutions are due by 11:59PM on November 19, 2021

Written Exercises

There are no written problems in this assignment.

Programming Exercise

Part 1: Codes

For symmetric positive definite matrix problems, implement

- A matrix-vector product for a symmetric matrix that supports the structures discussed below.
- A factorization routine and an associated solution routine for a symmetric positive definite tridiagonal matrix.
- A routine that solves a lower or upper triangular system defined by a lower triangular matrix L and the upper triangular matrix L^T, i.e., that upper triangular matrix is specified by transposing a lower triangular matrix. The matrix should not be transposed within the data structure but the appropriates should be accessed from the data structure storing L.
- Routines that use the last two routines to solve the preconditioning equation $Mz_k = r_k$ on each step (see the list of preconditioners to be supported below).
- Preconditioned Richardson's Stationary method (PRF)
- Preconditioned Steepest Descent (PSD)
- Preconditioned Conjugate Gradient (PCG)

The codes for PSD and PCG should be based on the application of a two-sided symmetric positive definite preconditioner to $Ax = b$. Specifically,

$$\tilde{A} = C^{-1}AC^{-T}, \quad \tilde{x} = C^T x, \quad \tilde{b} = C^{-1}b$$

$$\tilde{x}_{k+1} = \tilde{x}_k + \tilde{\alpha}_k \tilde{r}_k.$$

Note that the notes and homework problems have already dealt with this for SD and CG and your solutions should be based on those discussions. In particular, code is given in the notes where $M = C^2$ and $Mz_k = r_k$ is solved on each step rather than explicitly using the two-sided form of the transformed system above. Here we will define M directly as a symmetric positive definite matrix.

The codes should be able to operate both with preconditioning, $M \neq I$, and without preconditioning, $M = I$. Note that the PRF code can be implmented as a stationary option in PSD (with or without preconditioning.)

The matrix-vector product routine should support and exploit the following structure in the matrix

1. A is a symmetric matrix, i.e., no zero/nonzero pattern. The code should not duplicate the storage of elements that are known to be the same, i.e., $\alpha_{ij} = \alpha_{ji}$ and only one should be stored.

2. A is symmetric and banded with the nonzeros constrained to the main diagonal, k subdiagonals and k superdiagonals. It should be able to operate with $k = 0$, i.e., a diagonal matrix, and up to $k = 6$.

Use problems with known solutions to provide sufficient evidence of the correctness of your codes presented in an appropriate and convincing manner and to probe the empirical validation of the predictions of the theory we have discussed.

For problems with a known solution, x^*, you can monitor the relative error $\|x_k - x^*\|/\|x^*\|$ when discussing the convergence behavior of the methods on a particular problem. This is, of course, not a real convergence check but is useful for empirical evaluations.

For problems with known or unknown solutions monitoring $\|r_k\|/\|b\|$ is a reasonable assessment of how well the current estimate of the solution solves the system. Note that when $M \neq I$, i.e., preconditioning is used the "preconditioned residual" z_k is also available. So there is also the ratio $\|z_k\|/\|M^{-1}b\|$ that can be tracked. Note in your discussions if you see any significantly different behavior for the two residuals.

Codes should be able to run in both single and double precision, but for this assignment double precision will be used to represent "truth" or infinite precision compared to single precision. Therefore, all of your problems should be initially generated as a double precision matrix, A_{dp} and two double precision vectors x^* (the known solution) and b_{dp} (the righthand side vector). Given A_{dp} and x^*, b_{dp} should be computed in double precision with your matrix-vector product routine.

You should copy A_{dp} to a single precision version A_{sp} and b_{dp} to a single precision b_{sp} and pass the single precision versions to the iterative methods as input. The known solution x^* should be specified in double precision and The iterative methods and all associated routines should run in single precision.

Double precision should be used to check the computed single precision vectors with the "truth" stored as double precision vectors. So $\|DP(x_k) - x^*\|/\|x^*\|$ should be computed with the known solution x^* in its double precision as it was specified when the problem was

constructed. $DP(x_k)$ represents the computed x_k, which is single precision, copied into a double precision vector $DP(x_k)$.

Similarly, $\|r_k\|/\|b\|$ can be computed in double precision when it is used for assessing the performance of a method. If it is used as a termination criterion then of course it is part of the iterative method and should be in single precision within the code.

Part 2: Influence of the Spectrum

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 RF, 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.

(4.1.a) We have seen several convergence results for RF, SD and CG related to the spectrum and bounds on the stepsize for RF (that are related to bounds on the stepsize for SD). Design a set of experiments that choose Λ to influence the behavior of these methods. All methods should be run without preconditioning in this part of the assignment.

(4.1.b) 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 a representative iteration as well as statistics and distributions of errors, residuals, and iteration counts from a set of runs.

(4.1.c) **You need not run large sets for this part of the assignment. A moderate number of well-chosen problems is sufficient if the discussion is clear and the data is presented in a precise and compact manner.**

(4.1.d) Consider RF and SD and the choice of stepsize. What happens when RF is run with α_{opt}? How does the convergence behavior related to SD with its nonstationary and locally optimal choice of α_k? (Recall in this set of tests you are omniscient and have all of the eigenvalues)

A method that is not convergent for a particular $Ax = b$ does not necessarily diverge for all x_0. What happens with RF when α approaches and possibly exceeds $2/\lambda_{max}$? How would you choose Λ, b, and x_0 to show that, at least in exact arithmetic, you can make the iteration diverge or converge while $\alpha > 2/\lambda_{max}$.

You can perform similar experiments on SD by taking the usual $\alpha_k = r_k^T r_k / r_k^T A r_k$ and scaling it by some $0 < \mu \leq 2$, i.e., used the stepsize $\tilde{\alpha}_k = \mu(r_k^T r_k)/(r_k^T A r_k)$. Does SD still converge? Does it slow down? What happens if you take $\mu > 2$?

(4.1.e) Comment on the general effect of single precision on the ability of the algorithms to solve systems to a particular accuracy. Specifically, how does precision limit show up in the trends of the "exact" solution or its associated residual? You can also run a few of the problems completely in double precision, i.e., the problem specified by A_{dp}, x_{dp} and b_{dp} solved by the iterative method run entirely in double precision to approximate what an "infinite" precision computation would produce.

Part 3: Influence of the Preconditioners

(4.1.a) Let A be a banded symmetric positive definite matrix with nonzeros restricted to the k subdiagonals, k superdiagonals, and an all positive main diagonal.

(4.1.b) Your preconditioning routines discussed above should support preconditioning in PSD and PCG using a symmetric positive definite preconditioner defined by at the following:

- Jacobi's preconditioner, i.e., diagonal preconditioning with $M_1 = diag(A)$ where $diag(A)$ is the diagonal matrix containing the diagonal elements of A.
- The Symmetric Gauss-Seidel preconditioner M_{SGS}. If A is symmetric positive definite, $D = diag(A)$ and $A = D - L - L^T$ with $-L$ the elements in the strict lower triangular part of A, i.e., α_{ij} with $i > j$. The matrix L therefore has elements $e_i^T(L)e_j = -\alpha_{ij}$ with $i > j$ and 0 in all other positions in L. Be very careful with the signs as it is crucial to the definition of the preconditioner that they be correct. For example for $n = 4$,

$$A = \begin{pmatrix} 30 & 1 & 2 & 3 \\ 1 & 40 & 4 & 5 \\ 2 & 4 & 50 & 6 \\ 3 & 5 & 6 & 60 \end{pmatrix}, \quad D = \begin{pmatrix} 30 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 60 \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -2 & -4 & 0 & 0 \\ -3 & -5 & -6 & 0 \end{pmatrix}.$$

  If A is symmetric positive definite then so is $M_{SGS} = (D - L)D^{-1}(D - L^T)$ and its Cholesky factorization is easily determined to be $M_{SGS} = (D - L)D^{-1/2}D^{-1/2}(D - L^T) = CC^T$. So no computation is needed to form the factorization used in the iteration. Note that if A is banded this structure is also reflected in L
- Block diagonal preconditioning using 2×2 diagonal blocks of A. For exam-

ple, for $n = 8$

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} & \alpha_{17} & \alpha_{18} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} & \alpha_{25} & \alpha_{26} & \alpha_{27} & \alpha_{28} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} & \alpha_{35} & \alpha_{36} & \alpha_{37} & \alpha_{38} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} & \alpha_{45} & \alpha_{46} & \alpha_{47} & \alpha_{48} \\ \alpha_{51} & \alpha_{52} & \alpha_{53} & \alpha_{54} & \alpha_{55} & \alpha_{56} & \alpha_{57} & \alpha_{58} \\ \alpha_{61} & \alpha_{62} & \alpha_{63} & \alpha_{64} & \alpha_{65} & \alpha_{66} & \alpha_{67} & \alpha_{68} \\ \alpha_{71} & \alpha_{72} & \alpha_{73} & \alpha_{74} & \alpha_{75} & \alpha_{76} & \alpha_{77} & \alpha_{78} \\ \alpha_{81} & \alpha_{82} & \alpha_{83} & \alpha_{84} & \alpha_{85} & \alpha_{86} & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

The block diagonal preconditioner using 2×2 diagonal blocks is

$$M_2 = \begin{pmatrix} \alpha_{11} & \alpha_{12} & & & & & & \\ \alpha_{21} & \alpha_{22} & & & & & & \\ & & \alpha_{33} & \alpha_{34} & & & & \\ & & \alpha_{43} & \alpha_{44} & & & & \\ & & & & \alpha_{55} & \alpha_{56} & & \\ & & & & \alpha_{65} & \alpha_{66} & & \\ & & & & & & \alpha_{77} & \alpha_{78} \\ & & & & & & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

Note that in order to solve the systems on each iteration that involes P_2 you must use the Cholesky factorizations of each of the 2×2 diagonal blocks. Formulas for these are easily derived and the factors should be computed at the beginning of the solution process so only forward and backward solves are required on each iteration.

- Tridiagonal preconditioner, i.e., M_3 consists of the elements of A on the main diagonal, first subdiagonal, and first superdiagonal. Clearly this should only be used for $k \geq 2$ otherwise $M_3 = A$. Note that this means you must implement a factorization routine for symmetric tridiagonal matrices to compute the Cholesky factorization or LDL^T if you prefer. (assuming M_3 is positive definite). This factorization can be computed once at the beginning iteration and all solutions of the preconditioning system on subsequent iterations are found by a forward and backward solve. Make sure you implement this factorization efficiently and reliably. If you encounter a symmetric that is not positive definite you should return an indication that the preconditioner failed to exist which along with lack of convergence are the two main failure modes of iterative methods. Note also that if A is generated as a strictly diagonally dominant symmetric positive definite matrix (banded or not) the M_3 must be symmetric positive definite. It is recommended that you start with that construction of A for testing M_3.

The tridiagonal preconditioner for A above is

$$M_3 = \begin{pmatrix} \alpha_{11} & \alpha_{12} & & & & & & \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & & & & & \\ & \alpha_{32} & \alpha_{33} & \alpha_{34} & & & & \\ & & \alpha_{43} & \alpha_{44} & \alpha_{45} & & & \\ & & & \alpha_{54} & \alpha_{55} & \alpha_{56} & & \\ & & & & \alpha_{65} & \alpha_{66} & \alpha_{67} & \\ & & & & & \alpha_{76} & \alpha_{77} & \alpha_{78} \\ & & & & & & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

(4.1.c) Design tests that generate banded A for various values of k and n 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 banded 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.

If A is symmetric positive definite then M_1 is symmetric (diagonal) and positive definite since the diagonal elements of A must be positive. M_{SGS} has a Cholesky factorization when A is symmetric positive definite. So it too is symmetric positive definite. M_2 must be symmetric positive definite. This is easily seen since $v^T Av > 0$ by assumption and, for example, taking $v = \nu_1 e_1 + \nu_2 e_2$, i.e., any vector in the $span[e_1, e_2]$, demonstrates that the first 2×2 diagonal block is positive definite. This easily generalizes to the other blocks.

While M_3 is by definition symmetric, it does not have to be postive definite and therefore, since we are requiring that the preconditioner be positive definite, P_3 may not be usable on all matrices in the test. The Cholesky factorization will fail, i.e., it will not produce lower triangular C with positive diagonal elements such that $M_3 = CC^T$. For example, the following A is positive definite but the corresponding P_3 is not.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \quad M_3 = \begin{pmatrix} 1 & 1 & & & \\ 1 & 2 & 2 & & \\ & 2 & 3 & 3 & \\ & & 3 & 4 & 4 \\ & & & 4 & 5 \end{pmatrix}$$

Note that C when it exists for M_3 must be banded with only a positive main diagonal and a single subdiagonal. This must be exploited in your Cholesky routine and in your preconditioned iteration. This is also true of the Cholesky factor of A; if A has k subdiagonals then so does L in $A = LL^T$. This can be used to generate test matrices.

Requirements of Code

- The techniques used to gain an efficient implementation in space and computation must be described clearly and concisely.
- For the banded symmetric matrix-vector multiplication, your code must use $O(n)$ storage for the matrix A and the preconditioner for any size n.
- The computation of the banded symmetric matrix vector product $Av \rightarrow w$ must be done efficiently in $O(n)$ computations using the efficient storage scheme.
- The solution of systems $Mz_k = r_k$ required in each iteration must be done efficiently in $O(n)$ computations using the efficient storage scheme for the appropriate preconditioners. For any other preconditioners you attempt you may use a dense storage approach.
- Your code should collect information at each iteration on
  - $\|e^{(k)}\|_2/\|x^*\|_2 = \|x_k - x^*\|_2/\|x^*\|_2$ assuming $Ax^* = b$ and that x^* is known and not small in norm. The precision should be as described above.
  - Similarly $\|r_k\|_2/\|b\|_2$ and $\|z_k\|_2/\|Mb\|_2$ using precision as described above.
- 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 $\|r_k\|_2/\|b\|_2$ and $\|e^{(k)}\|_2/\|x_{true}\|_2$, in addition to plotting the quantities vs. the iteration number k, you should plot the logs vs. the iteration number k. Recall the asymptotic behavior discussion where one would expect linear behavior in the log of the error and residual norms. The observed behavior should be consistent with the theory in so far as roundoff allows. In addition to respresentative examples you should present global summaries using statistics and distributions of errors (histograms), residuals, and iteration counts from larger sets of runs.
- However, as mentioned above, you do not have to resort to large sets of runs for this assignment. Carefully, choose sets of examples and present them appropriately with your description of why the data was collected, what you observed, and what you concluded.

Test Matrices

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

There are many ways to derive test matrices:

- nonsingular matrices from factorizations (possibly structured to impose structure on A), e.g., random lower triangular L with positive diagonal elements and using $A = LL^T$. This does not directly control the spectrum however. You may use other library routines to compute the eigenvalues and to compare to the eigenvalues of the the preconditioned matrix $C^{-1}AC^{-1}$ in this case.

- Nonsingularity and controlling the spectrum can be done using diagonal dominance (strict in column and row for symmetric matrices).

- Modifications to random matrices using some of the techniques above can also be used (see the Gershgorin theorems in the text);

- Generating orthogonal matrices and combining with diagonal and other matrices to generate A, i.e., choose eigenvalues as the elements of Λ then apply, say, elementary reflectors based on a series of randomly chosen vector u_i with $\|u_i\|_2 = 1$:

$$A = (I - 2u_k u_k^T) \dots (I - 2u_1 u_1^T)\Lambda(I - 2u_1 u_1^T) \dots (I - 2u_k u_k^T).$$

  This A will be dense, symmetric positive definite with the spectrum given by Λ.

- Λ can be changed into a banded matrix if the transformations applied to the left and right are constructed from plane rotations.

For example, given

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_8 \end{pmatrix}$$

you can apply orthogonal matrices $Q_k \dots Q_1 \Lambda Q_1^T \dots Q_k^T$ where the Q_i's have forms such as

$$Q_1 = \begin{pmatrix} \gamma_1 & \sigma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\sigma_1 & \gamma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma_2 & \sigma_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\sigma_2 & \gamma_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma_3 & \sigma_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\sigma_3 & \gamma_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4 & \sigma_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\sigma & \gamma_4 \end{pmatrix}$$

where $\gamma_i = \cos\theta_i$ and $\sigma_i = \sin\theta_i$ with randomly chosen angles;

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & \gamma_1 & \sigma_1 & 0 & 0 & 0 & 0 & 0 & \\ 0 & -\sigma_1 & \gamma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_2 & \sigma_2 & 0 & 0 & 0 & \\ 0 & 0 & 0 & -\sigma_2 & \gamma_2 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \gamma_3 & \sigma_3 & 0 & \\ 0 & 0 & 0 & 0 & 0 & -\sigma_3 & \gamma_3 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \end{pmatrix}$$

where these are different angles;

$$Q_3 = \begin{pmatrix} \gamma_1 & 0 & \sigma_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & \sigma_2 & 0 & 0 & 0 & 0 \\ -\sigma_1 & 0 & \gamma_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\sigma_2 & 0 & \gamma_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma_3 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma_4 & 0 & \sigma_4 \\ 0 & 0 & 0 & 0 & -\sigma_3 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sigma_4 & 0 & \gamma_4 \end{pmatrix}$$

where these are different angles. The banded structure of the product must be verified and can be controlled by what planes are used by each of orthogonal matrices, Q_i.

In addition to these techniques you can make use of matrix families with known eigenvalues and/or eigenvectors. For example, Toeplitz tridiagonal matrices parameterized by a single parameter

$$T_\alpha = \begin{pmatrix} \alpha & -1 & 0 & \dots & \dots & \dots & 0 \\ -1 & \alpha & -1 & 0 & \dots & \dots & 0 \\ 0 & -1 & \alpha & -1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & -1 & \alpha & -1 & 0 \\ 0 & \dots & \dots & 0 & -1 & \alpha & -1 \\ 0 & \dots & \dots & \dots & 0 & -1 & \alpha \end{pmatrix}$$

have eigenvalues

$$\lambda_j = \alpha - 2\cos j\theta, \quad \theta = \frac{\pi}{n+1} q_j = \big(\sin(j\theta),\ \sin(2j\theta), \dots, \sin(nj\theta)\big)^T$$

Feel free to consult the literature for other such families.

You may use environments like MATLAB or libraries like LAPACK to generate matrices for testing and, more importantly, to analyze the spectrum of the preconditioned matrices to see if the preconditioners have altered the spectrum in such a way as to accelerate convergence. Recall, that the PCG and PSD iterations can be shown to be equivalent to

solving a system with $\tilde{A} = C^{-1}AC^{-1}$ or $\tilde{A} = L^{-1}AL^{-T}$ where the preconditioner $P = C^2$ or $P = LL^T$. This equivalent symmetric form should be when computing the spectrum of the precondtioned matrix since codes for symmetric eigenvalue problems are plentiful and reliable. Clearly, this type of numerical evaluation of the spectrum for both A and $\tilde{A}$ should be done only for problem sizes where the computation is reasonable. **Be sure to cite the libraries/routines or environments/commands used in you solutions.**