Program 5 Foundations of Computational Math 2 Spring 2019

Due date: 11:59PM on 15 Monday April 2019

Part 1: Codes

For symmetric positive definite matrix problems, implement

- Symmetric Gauss-Seidel (SGS)
- Preconditioned Minimal Residual (PMR)
- Preconditioned Nonstatitonary Descent (PD)
- Preconditioned Steepest Descent (PSD)
- Preconditioned Conjugate Gradient (PCG)

PMR should be derived from the application of a two-sided symmetric preconditioner to $Ax = b$. Specifically,

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

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

and you must derive a PMR iteration in terms of x_k, A, b, and C and explain how you implement it in a manner similar to all of the other codes. Note that the notes and homework problems have already dealt with this for SD and CG. Of course, PMR should also work on nonsymmetric matrices with a symmetric preconditioner. This, in general may not work that well, i.e., you probably should use a nonsymmetric precondition in that case. Your SGS and PMR codes should also include a version where the symmetric banded matrix-vector multiplication is replaced by a dense possibly nonsymmetric matrix for running with the symmetric preconditioner or no preconditioner. SGS should be used with no additional preconditioner, i.e., take $C = I$,

The Nonstationary PD family includes PSD when the direction vector for each iteration is taken as the residual, i.e., $p_k = r_k$. For controlling the experiments, you may wish to impose, without loss of generality, $p_k^T r_k > 0$. Recall that the convergence analysis of PD required that $p_k^T r_k$ does not tend to 0. The codes should be able to operate without preconditioning, i.e., with the preconditioner set to I. The PSD code should be able to operate in a stationary mode, i.e., with a fixed α. The codes must be able to operate in both single and double precision. The matrix-vector product routine should assume that the matrix 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, up to

$k = 6$. You should also use an implementation that assumes A is symmetric and dense (you should exploit the symmetry in your matrix vector multiplication in this case). This will make it easier to generate matrices with particular spectrum values without worrying about imposing banded structure.

Use problems with known and unknown solutions to provide sufficient evidence of the correctness of your codes presented in an appropriate and convincing manner. For problems with a known solution, x^*, you can monitor the relative error $\|x_k - x^*\|/\|x^*\|$. 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.

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 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.

(1.1) We have seen several convergence results for the nonstationary descent family, 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.

(1.2) 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 a representative iteration as well as statistics and distributions of errors, residuals, and iteration counts from larger sets of runs.

(1.3) Run both single and double precision experiments and determine the general effect of precision on the ability of the algorithms to solve systems to a particular accuracy. Specifically, how does precision limit the accuracy in terms of relative error or size of the residual relative to b.

(1.4) Of course, the nonstationary descent family PD is not a specific method, i.e., SD is a particular member of the family. So for the experiments using this family you must manipulate your choice of p_k to investigate its influence on performance. You can intuitively think of SD and CG as two extremes of this family. SD is purely local in its choice of p_k and CG is an optimial choice of p_k.

Part 3: Influence of the Preconditioners

(1.1) Let A be a banded symmetric positive definite matrix with nonzeros restricted to the k subdiagonals, k superdiagonals, and an all positive main diagonal. Implement support in the codes for preconditioning using a symmetric positive definite preconditioner defined with the following three techniques:

- Jacobi's preconditioner, i.e., diagonal preconditioning with $P_1 = diag(A)$ where $diag(A)$ is the diagonal matrix containing the diagonal elements of A. You can also use any of the precondtioners that define a stationary method. For example here you should also consider $\tilde{P}_1 = P_{SGS}$. If A is symmetric positive definite then so is $P_{SGS} = (D - L)D^{-1}(D - L^T)$ and its Cholesky factorization is easily determined to be $P_{SGS} = (D - L)D^{-1/2}D^{-1/2}(D - L^T) = CC^T$.
- block diagonal preconditioning using 2×2 diagonal blocks of A. For example, 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

$$P_2 = \begin{pmatrix} \alpha_{11} & \alpha_{12} & & & & & & \\ \alpha_{21} & \alpha_{22} & & & & & & \\ & & \alpha_{33} & \alpha_{34} & & & & \\ & & \alpha_{43} & \alpha_{44} & & & & \\ & & & & \alpha_{55} & \alpha_{56} & & \\ & & & & \alpha_{65} & \alpha_{66} & & \\ & & & & & & \alpha_{77} & \alpha_{78} \\ & & & & & & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

  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 compted at the beginning of the solution process so only forward and backward solves are required on each iteration.
- tridiagonal preconditioner, i.e., P_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 $P_3 = A$. Note that this means you must implement a factorization routine for symmetric tridiagonal matrices to compute the Cholesky factorization (assuming P_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.

The tridiagonal preconditioner for A above is

$$P_3 = \begin{pmatrix} \alpha_{11} & \alpha_{12} & & & & & & \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & & & & & \\ & \alpha_{32} & \alpha_{33} & \alpha_{34} & & & & \\ & & \alpha_{43} & \alpha_{44} & \alpha_{45} & & & \\ & & & \alpha_{54} & \alpha_{55} & \alpha_{56} & & \\ & & & & \alpha_{65} & \alpha_{66} & \alpha_{67} & \\ & & & & & \alpha_{76} & \alpha_{77} & \alpha_{78} \\ & & & & & & \alpha_{87} & \alpha_{88} \end{pmatrix}$$

- Preconditioning by $P = A$ and using a finite number of steps of an iterative method to approximate the solution of the preconditioner equation on each iteration of the main method. Since the $A = P$ is assumed symmetric positive definite the iterative method you use to solve $Pz_k = p_k$ must be convergent for such matrices. You should include using a few steps of SGS as one of the preconditioners for say CG in your solutions.

(1.2) 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 P_1 is symmetric (diagonal) and positive definite since the diagonal elements of A must be positive. Similarly, P_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 P_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 L with positive diagonal elements such that $P_3 = LL^T$. For example, the following A is positive definite but the corresponding P_3 is not.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \quad P_3 = \begin{pmatrix} 1 & 1 & & & \\ 1 & 2 & 2 & & \\ & 2 & 3 & 3 & \\ & & 3 & 4 & 4 \\ & & & 4 & 5 \end{pmatrix}$$

Note that L when it exists for P_3 must be banded with only a 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.

Part 4: Nonsymmetric Matrices

You should use the dense nonsymmetric matrix-vector product version of SGS and PMR to perform a small number of experiments that demonstrate the convergence (or lack thereof) results for SGS and PMR discussed in the notes and homeworks. You can generate a nonsymmetic matrix (dense) with a specific spectrum using the Real Schur Decomposition. Recall, that any complex matrix can be written as $A = QRQ^H$ where $Q^HQ = QQ^H = I$, i.e., Q is an unitary matrix, and R is upper triangular with the eigenvalues (possibly complex) of A on the main diagonal. If A is real then any complex eigenvalues occur in complex conjugate pairs. For real A the Real Schur Decomposition has the form $A = QRQ^T$ where Q is real, $Q^TQ = QQ^T = I$, and R is real but possibly block upper triangular. In particular, real eigenvalues appear as a 1×1 block (a single element) on the diagonal of R while a pair of complex conjugate eigenvalues appear as a 2×2 real block on the block diagonal of R where the pair of eigenvalues of A are also the eigenvalues of the 2×2 block.

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 $Av \to w$ must be done efficiently in $O(n)$ computations using the efficient storage scheme.

- 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 the appropriate preconditioners. For any other preconditioners you attempt you may use a dense storage approach. ¡

- Your must produce single and double precision versions of the codes.

- Your code should collect information at each iteration on

  - $\|e^{(k)}\|_2/\|x_{true}\|_2 = \|x_k - x_{true}\|_2/\|x_{true}\|_2$ assuming $Ax_{true} = b$ and that x_{true} is known and not small in norm.

  - $\|r_k\|_2/\|b\|_2$

- 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.

Test Matrices

For Part 2 the 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. In Part 2 it is not important to do massive amounts of testing but make sure that you clearly describe the experiments and their purpose.

You have seen many 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

The only complications here are the requirements of symmetric and positive definite and you should describe clearly how you are generating your matrices for Part 3.

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.**

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 class website.
- 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.
- 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 FSU Dropbox at `https://dropbox.fsu.edu`. Drop the files off for Zhifeng Deng using his MyFSU email `zd16d@my.fsu.edu`. (Note: **do not use `zdeng@math.fsu.edu` to drop off the files**) You should login to FSU Dropbox using your MyFSU login. If for some reason you cannot use FSU Dropbox please email the files to Zhifeng at the email address above.