MAT 226B, Winter 2018

Homework 5

(due by Friday, March 16, 11:59 pm)

General Instructions

- You are required to submit your homework by uploading a single pdf file to Canvas. Note that the due dates set in Canvas are hard deadlines. I will not accept any submissions outside of Canvas or after the deadline.
- If at all possible, use a text processing tool (such as LaTeX) for the preparation of your homework. If you submit scanned-in hand-written assignments, make sure that you write clearly and that you present your solutions in a well-organized fashion. If I cannot read your homework, I will not be able to grade it!
- Feel free to discuss the problems on the homework sets with other students, but you do need to submit your own write-up of the solutions and your own MATLAB codes. If there are students with solutions that were obviously copied, then each involved student (regardless of who copied from whom) will only get the fraction of the points corresponding to the number of involved students.
- Test cases for computational problems are often provided as binary Matlab files. For example, suppose the file "`LS.mat`" contains the coefficient matrix A and the right-hand side b of a system of linear equations. The Matlab command "`load('LS.mat')`" will load A and b into Matlab.
- When you are asked to print out numerical results, print real numbers in 15-digit floating-point format. You can use the Matlab command "`format long e`" to switch to that format from Matlab's default format. For example, the number 10π would be printed out as `3.141592653589793e+01` in 15-digit floating-point format
- When you are asked to write Matlab programs, include printouts of your codes in your homework.

---

1. Consider the following generalization of the two-dimensional Poisson's problem on the unit square:

$$\begin{aligned} -u_{xx} - u_{yy} + \gamma u_x &= f(x,y), \quad (x,y) \in R := (0,1) \times (0,1), \\ u &= g(x,y), \quad (x,y) \in \partial R. \end{aligned} \tag{1}$$

   Here $\gamma > 0$ is a parameter.

We discretize (1) using grid points

$$(x_j, y_k) := (jh, kh), \quad j, k = 0, 1, \ldots, m+1,$$

where $m \geq 1$ is an integer and $h := 1/(m+1)$. We employ centered differences to approximate u_{xx} and u_{yy} and the approximation

$$u_x\big|_{x=x_j, y=y_k} \approx \frac{u(x_{j+1}, y_k) - u(x_{j-1}, y_k)}{2h}$$

for u_x. The resulting linear system is of the form

$$Av = b, \quad A = A_0 + \gamma A_1, \tag{2}$$

where A_0 is a symmetric positive definite matrix corresponding to the discretization of $-u_{xx} - u_{yy}$ and A_1 is a skew-symmetric matrix corresponding to the discretization of u_x. Note that the matrix A is nonsymmetric if $\gamma \neq 0$.

Use Matlab's "`gmres`" function with left preconditioning

$$M = A_0 = M_1 M_2, \quad \text{where} \quad M_1 = A_0 \quad \text{and} \quad M_2 = I,$$

to solve the linear system (2). Employ your FFT-based algorithm from Problem 1 of Homework 3 for the solution of linear systems with M_1. This can be done by calling Matlab's GMRES routine with a function handle of a function that returns $A'v$, where

$$A' = M_1^{-1} A = I + \gamma M_1^{-1} A_1. \tag{3}$$

Your Matlab program should only use solves with M_1 and multiplications with A_1, but no multiplications with A or A_0.

To test your Matlab program, solve problem (1) with the functions

$$f(x, y) = x^3 \, y^2 \, e^{2-x-y} \quad \text{and} \quad g(x, y) = 1$$

for each of the parameter values

$$\gamma = 1, \ 10, \ 50, \ 100, \ 1000. \tag{4}$$

In each case, choose $m = 100$ for the disretization and run GMRES (without restarts) with zero initial guess $v_0 = 0$ and convergence tolerance

`tol = 1e-10`.

Print out the number of GMRES iterations and the relative residual norm `RELRES` of the final GMRES iterate.

How do your iteration counts depend on γ?

2. Let $A \in \mathbb{R}^{n\times n}$, $A \succ 0$, $b, x_0 \in \mathbb{R}^n$, and $r_0 := b - Ax_0$. Let

$$x_k^{\mathrm{CG}}, \quad k = 1, 2, \ldots, d(A, r_0),$$

denote the iterates produced by the CG algorithm applied to $Ax = b$ with initial guess x_0.

Show that for every $k = 1, 2, \ldots, d(A, r_0)$, x_k^{CG} satisfies the following relation

$$x_k^{\mathrm{CG}} = x_0 + V_k z_k, \quad \text{where} \quad z_k := \|r_0\|_2 T_k^{-1} e_1^{(k)}.$$

Here, $e_1^{(k)}$ denotes the first unit vector of length k, T_k is the tridiagonal matrix generated by k steps of the Hermitian Lanczos process applied to A and r_0, and V_k denotes the $n \times k$ matrix that contains the corresponding first k Lanczos vectors as columns.

3. Let $A \in \mathbb{C}^{n\times n}$ be a large sparse Hermitian matrix and $b \in \mathbb{C}^n$, $b \neq 0$. Consider the vector-valued function $g : \mathbb{C} \mapsto \mathbb{C}^n$ defined by

$$g(\lambda) := e^{A\lambda} b \quad \text{for all} \quad \lambda \in \mathbb{C}.$$

How can you use the Hermitian Lanczos process to generate approximations

$$g_k(\lambda) \approx g(\lambda), \quad k = 1, 2, \ldots, d(A, b),$$

that for each $\lambda \in \mathbb{C}$ can be evaluated by computing the value of only a $k \times k$ matrix function (instead of an $n \times n$ matrix function for $g(\lambda)$)?

Hint: Choose an appropriate initial vector r for the Hermitian Lanczos process.

4. Write a Matlab program that implements the Arnoldi process for matrices $A \in \mathbb{C}^{n\times n}$ and starting vectors $r \in \mathbb{C}^n$, $r \neq 0$, as presented in class. Use an input parameter $k_{\max}$ to limit the maximum number of Arnoldi steps. The output of your program should be the upper-Hessenberg matrix $\widetilde{H}_k \in \mathbb{C}^{(k+1)\times k}$ and the matrix $V_{k+1} \in \mathbb{C}^{n\times(k+1)}$ with the first $k + 1$ Arnoldi vectors, where k is the iteration index at termination of your algorithm.

Employ your implementation of the Arnoldi process to compute approximate eigenpairs of the matrix $A = A(\gamma) = A'$, where A' is the preconditioned matrix defined in (3); as in Problem 1, use $m = 100$. For each of the 5 values of γ in (4), use the starting vector r provided in the Matlab file "`HW5_P4.mat`" and $k_{\max} = 300$. Employ Matlab's "`eig`" function to compute the eigenpairs $\left(\tilde{\lambda}_j, z_j\right)$, $j = 1, 2, \ldots, k$, of H_k. Use the formula derived in class, to compute the residual norm

$$\rho_j = \left\| A\tilde{x}_j - \tilde{\lambda}_j \tilde{x}_j \right\|_2$$

of the approximate eigenpairs $\left(\tilde{\lambda}_j, \tilde{x}_j\right)$, $j = 1, 2, \ldots, k$, of A, without computing the approximate eigenvectors $\tilde{x}_j$.

For each of your 5 runs (one for each value of γ), print out

$$\min_{1\le j\le k} \rho_j \quad \text{and} \quad \max_{1\le j\le k} \rho_j,$$

together with all the approximate eigenvalues $\tilde{\lambda}_j$ for which the minimum and maximum is attained, and submit a plot that shows all k approximate eigenvalues of A.

5. Write a Matlab program that implements the Hermitian Lanczos process for matrices $A = A^H \in \mathbb{C}^{n\times n}$ and starting vectors $r \in \mathbb{C}^n$, $r \neq 0$, as presented in class. Use an input parameter $k_{\max}$ to limit the maximum number of Lanczos steps. Your program should only store the last two Lanczos vectors, v_k and v_{k-1}, so that you can run the program for any number of steps. The output of your routine should be the tridiagonal matrix $T_k \in \mathbb{R}^{k\times k}$, where k is the iteration index at termination of your algorithm.

   To test your program, use $n \times n$ matrices A generated by

   `A = make_3d_laplacian(m).`

   Note that $n = m^3$ and that the exact eigenvalues $\lambda = \lambda_{i,j,\ell}$ of A are given by

$$\lambda_{i,j,\ell} = 2\Big(3 - \cos\frac{i\pi}{m+1} - \cos\frac{j\pi}{m+1} - \cos\frac{\ell\pi}{m+1}\Big), \quad i,j,\ell = 1,2,\ldots,m.$$

   (a) Test your program on the 8×8 matrix A corresponding to $m = 2$ and the starting vector r provided in the Matlab file "`HW5_P5a.mat`". Run the algorithm for $k_{\max} = 4$ steps and compute the eigenvalues of T_4 with Matlab's "`eig`" command. Compare these eigenvalues with the exact eigenvalues of A.

   (b) Employ your program to obtain approximate eigenvalues of the 328509×328509 matrix A corresponding to $m = 69$, using the starting vector r provided in the Matlab file "`HW5_P5b.mat`". Compute the approximate eigenvalues obtained from the Lanczos matrices T_k for

$$k = 250,\ 500,\ 1000,\ 2000.$$

   For each of these values of k, print out the 10 smallest and the 10 largest eigenvalues of T_k and compare them with the 10 smallest and the 10 largest exact eigenvalues of A. What can you say about the quality of the approximate eigenvalues obtained from the Lanczos process?