# MAT 226B, Winter 2020

# Homework 4

(due by Thursday, March 12, 11:59 pm)

---

**General Instructions**

- You are required to submit each of your assignments via file upload 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. For each assignment, upload two files: a single pdf file of your assignment and a single zip file that contains all your Matlab files. To prepare the zip file, create a directory that contains all your Matlab files and then zip that directory. For each of the subproblems that require Matlab computations include a single driver file so that I can run and check your program.
- If at all possible, use a text processing tool (such as LaTeX) for the preparation of your assignments. 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 assignment, I will not be able to grade it!
- You are required to solve the problems on the homework sets and on the final project yourself! If there are students with solutions or program codes that were obviously copied or are trivial modifications of each other, 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 $10\pi$ would be printed out as `3.141592653589793e+01` in 15-digit floating-point format.

---

1. Let $A \in \mathbb{C}^{n\times n}$ and $r, c \in \mathbb{C}^n$ with $r, c \neq 0$. Assume that $d(A, r) \geq 2$ and $d(A^T, c) \geq 2$.

   (a) Show that a breakdown occurs in step $k = 1$ of the nonsymmetric Lanczos process if, and only if,
   $$c^T r = 0.$$

(b) Show that a breakdown occurs in step $k = 2$ of the nonsymmetric Lanczos process if, and only if,

$$c^T r \neq 0 \quad \text{and} \quad \left(c^T r\right)\left(c^T A^2 r\right) = \left(c^T A r\right)^2.$$

2. Write a Matlab function that implements the Arnoldi process for general $A \in \mathbb{C}^{n\times n}$ and $r \in \mathbb{C}^n$, $r \neq 0$, as presented in class. Use an input parameter `KMAX` to limit the maximum number of Arnoldi steps. The output of your function should be the upper-Hessenberg matrix $H_k \in \mathbb{C}^{k\times k}$ and the matrix $V_k \in \mathbb{C}^{n\times k}$ with the first $k$ Arnoldi vectors as its columns, where $k$ is the iteration index at termination.

   To debug your function, use $A$ and $r$ provided in the Matlab file "`HW4_P2.mat`". Run the Arnoldi process for $k = 5$, $k = 10$, and $k = 20$ steps. For each case, employ Matlab's "`eig`" to compute the eigenvalues of $H_k$ and print out these eigenvalues. Compare the eigenvalues of $H_{20}$ and $A$.

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

   (a) To debug your function, use the $27 \times 27$ matrix

   ```
   A = make_3d_laplacian(3)
   ```

   and the vector $r$ provided in the Matlab file"`HW4_P3a.mat`". Run the Hermitian Lanczos process for $k = 7$ steps and compute the eigenvalues (using Matlab's "`eig`") of $T_7$. Compare these eigenvalues with the exact eigenvalues of $A$, which are given by

   $$2\Big(3 - \cos\frac{i\pi}{4} - \cos\frac{j\pi}{4} - \cos\frac{\ell\pi}{4}\Big), \quad i, j, \ell = 1, 2, 3.$$

   (b) Use your function with the vector $r$ provided in the Matlab file "`HW4_P3b.mat`" to obtain approximate eigenvalues of the $262144 \times 262144$ matrix

   ```
   A = make_3d_laplacian(64).
   ```

   Compute the approximate eigenvalues obtained from the Lanczos matrices $T_k$ for

   $$k = 100,\ 200,\ 400,\ 800,\ 1200,\ 1600,\ 2000.$$

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 $A$. Note that the 262144 eigenvalues of $A$ are given by

$$2\Big(3 - \cos\frac{i\pi}{65} - \cos\frac{j\pi}{65} - \cos\frac{\ell\pi}{65}\Big), \quad i, j, \ell = 1, 2, \ldots, 64.$$

What can you say about the quality of the approximate eigenvalues obtained from the Lanczos process?

4. Write a Matlab function that implements the nonsymmetric Lanczos process for general $A \in \mathbb{C}^{n\times n}$ and $r, c \in \mathbb{C}^n$, $r, c \neq 0$, as presented in class. Use an input parameter `KMAX` to limit the maximum number of Lanczos steps. Store only the last two right Lanczos vectors, $v_k$ and $v_{k-1}$, and the last two left Lanczos vectors, $w_k$ and $v_{w-1}$, so that you can run the program for any number of steps. The output of your function should be the tridiagonal matrix $T_k \in \mathbb{C}^{k\times k}$, where $k$ is the iteration index at termination.

   (a) To debug your function and to make sure it works properly for complex matrices, use $A$, $r$, and $c$ provided in the Matlab file "`HW4_P4a.mat`". Run the nonsymmetric Lanczos process for $k = 5$, $k = 10$, and $k = 20$ steps. For each case, employ Matlab's "`eig`" to compute the eigenvalues of $T_k$ and print out these eigenvalues. Compare the eigenvalues of $T_{20}$ and $A$.

   We now consider the preconditioned matrix $A' := M_1^{-1} A M_2^{-1}$ that arises when the preconditioner

$$M := (D_0 - F)D_0^{-1}(D_0 - G) = M_1 M_2, \quad M_1 := D_0 - F, \quad M_2 := D_0^{-1}(D_0 - G),$$

   is applied to the given matrix

$$A = D_0 - F - G$$

   with nonsingular diagonal part $D_0$, strictly lower-triangular part $-F$, and strictly upper-triangular part of $-G$.

   (b) Employ your Matlab implementations of the Arnoldi process and the nonsymmetric Lanczos process to obtain some eigenvalues of the preconditioned matrix $A'$ for the matrix $A$ (of size $n = 2197$) that is provided in the Matlab file "`HW4_P4b.mat`" along with starting vectors $r$ and $c$. The matrix $A'$ is small enough so that you can compute all of its 2197 eigenvalues by means of Matlab's "`eig`". By comparing your computed approximate eigenvalues with the 'exact' eigenvalues of $A'$ produced by "`eig`", identify the approximate eigenvalues that have converged. Try several runs with increasing $k$ to obtain as many converged eigenvalues as possible. For each of the two algorithms, produce a plot that shows the 'exact' eigenvalues and the converged approximate eigenvalues of $A'$ .

(c) Repeat the experiments of (b) for the matrix (of size $n = 262144$) and the starting vectors provided in the Matlab file "`HW4_P4c.mat`". Identify converged eigenvalues of $A'$ by comparing the approximate eigenvalues obtained after $k$ steps with the ones obtained after $k - 1$ steps. Try several runs with increasing $k$ to obtain as many converged eigenvalues as possible. For each of the two algorithms, produce a plot that shows the converged approximate eigenvalues of $A'$.