MAT 226B, Winter 2020
Homework 2

(due by Thursday, February 13, 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 10π would be printed out as `3.141592653589793e+01` in 15-digit floating-point format.

---

1. Let $A = \left[a_{jk}\right] \in \mathbb{R}^{n\times n}$ be a sparse matrix with `nnz` nonzero entries. As described in class, the indices (j,k) of the nonzero entries can be described by two integer vectors J and I of length `nnz` and $n+1$, respectively. Recall that in terms of J and I, the indices of the nonzero entries are given by

$$\bigl(J(i),k\bigr), \quad i = I(k), I(k)+1, \ldots, I(k+1)-1, \quad k = 1,2,\ldots,n.$$

(a) Write an efficient Matlab function that for any matrix A given in Matlab's sparse matrix format generates the corresponding vectors J and I. To check the efficiency of your function, run it on the 50000×50000 matrix A given in the binary Matlab file "`HW2_P1_a.mat`". Print out

$$J(100000), \quad J(150000), \quad J(200000), \quad J(250000), \quad J(300000)$$

and

$$I(10000), \quad I(20000), \quad I(30000), \quad I(40000), \quad I(50000).$$

(b) Run your function on the small sparse matrix A provided in the binary Matlab file "`HW2_P1_b.mat`". Print out the vectors J and I generated by your function.

2. Suppose that at the beginning of the k-th step of sparse LU factorization, the matrix $U^{(k)}$ has the following sparsity structure:

$$U^{(k)} = \left[\ u_{il}\ \right] = \begin{bmatrix} * & 0 & 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & * & * \\ * & 0 & * & * & 0 & * & 0 & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & * & 0 \\ 0 & * & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 & 0 & * & 0 & * \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & * & 0 & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & * \\ * & * & * & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Moreover, the nonzero entries $u_{il} \neq 0$ of $U^{(k)}$ are assumed to satisfy

$$\min_{i,l\,:\, u_{il}\neq 0} |u_{il}| \;\geq\; \frac{1}{7} \max_{i,l} |u_{il}|.$$

(a) Determine the number of fill-in elements and their location if the k-th step is performed without pivoting.

(b) Determine the number of fill-in elements and their location if the k-th step is performed with a pivot element determined via the practical Markowitz criterion (with parameter $\alpha = 0.1$) presented in class.

3. Consider the two-dimensional Poisson's equation on the unit square,

$$-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2} = f(x,y), \qquad 0 < x, y < 1,$$

together with boundary conditions of the form

$$\begin{aligned} v(x,0) &= b_0(x), && 0 < x < 1, \\ v(x,1) &= b_1(x), && 0 < x < 1, \\ v(0,y) &= c_0(y), && 0 < y < 1, \\ v(1,y) &= c_1(y), && 0 < y < 1. \end{aligned} \tag{1}$$

(a) Generalize the fast elliptic solver for zero boundary conditions, which was presented in class, to general boundary conditions of the form (1).

(b) Write a Matlab program that implements your solver from (a) using FFTs. The inputs should be the integer m determining the mesh size $h := 1/(m+1)$, an $m \times m$ matrix of the values $f_{jk} := f(x_j, y_k)$ of the right-hand side function f at the grid points $(x_j, y_k) := (jh, kh)$, and 4 vectors of length m containing the values of the functions b_0, b_1, c_0, and c_1 at the grid points on the 4 pieces of the boundary. The output should be an $m \times m$ matrix of approximate solutions v_{jk}. You should also produce three-dimensional plots of f and v. Use the FFT built into Matlab. Your program should not have to be more than a few lines long if you use all the features of Matlab that you can.

(c) To test your Matlab program, use test cases that have solutions of the form

$$v(x,y) = x^{\alpha} \cos(\beta \pi x) \sin(\gamma \pi y), \tag{2}$$

where $\alpha \geq 0$ and β, $\gamma \geq 1$ are integer-valued parameters. Determine the functions f, b_0, b_1, c_0, and c_1 so that the function (2) is indeed the solution of the above Poisson's equation.

(d) Use your FFT-based solver to compute approximate values

$$v_{jk} \approx v(x_j, y_k)$$

for the exact solution at the grid points (x_j, y_k). For each of your runs, determine the absolute error

$$\max_{j,k=1,2,\ldots,m} \left| v_{jk} - v(x_j, y_k) \right| \tag{3}$$

of your computed solution.

Use the following 5 test cases:

(i) $\alpha = 0$, $\beta = 1$, $\gamma = 1$;

(ii) $\alpha = 1$, $\beta = 2$, $\gamma = 1$;

(iii) $\alpha = 2$, $\beta = 2$, $\gamma = 3$;

(iv) $\alpha = 5$, $\beta = 2$, $\gamma = 3$;

(v) $\alpha = 5$, $\beta = 3$, $\gamma = 5$.

In each of these 5 cases, choose m large enough so that the error (3) is sufficiently small. Print out the m you have used, as well as the value of the corresponding error (3). In addition, write your driver file such that it produces on-screen plots of the right-hand side function f, the exact solution v, and your computed solution v_{jk} for the 5 test cases (i)–(v).

4. (a) Let $M \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix, $v_1, v_2, \ldots, v_k \in \mathbb{R}^n$ be k linearly independent vectors, and let $x^\star, x_0 \in \mathbb{R}^n$. Consider the minimization problem

$$\|x^\star - x_k\|_M = \min_{x \in x_0 + S} \|x^\star - x\|_M\,, \tag{4}$$

where $S := \operatorname{span}\{\, v_1, v_2, \ldots, v_k \,\} \subset \mathbb{R}^n$.

Use elementary multivariate calculus to show that problem (4) has a unique solution $x_k \in x_0 + S$.

Hint: Write x_k in the form

$$x_k = x_0 + \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k, \quad \alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{R},$$

and show that for an optimal x_k, the corresponding α_j's satisfy a system of linear equations with a nonsingular coefficient matrix.

(b) Let $Ax = b$ is a linear system with $A \in \mathbb{R}^{n\times n}$, $A \succ 0$, and $b \in \mathbb{R}^n$, and let $x_0 \in \mathbb{R}^n$.

Use your result from part (a) to show that the CG iterates x_k are uniquely defined by the CG method's minimization property.

5. Let $A =\in \mathbb{R}^{n\times n}$ be a general symmetric positive definite matrix, and consider the problem of solving the linear system $Ax = b$.

(a) Determine the exact number of flops (additions, subtractions, multiplications, divisions, and square roots) that are required to perform the k-th iteration of the CG algorithm presented in class as efficiently as possible.

(b) Suppose you know that

$$\kappa(A) = 9 \quad \text{and} \quad \left\|A^{-1}b - x_0\right\|_A = \frac{1}{2}.$$

Based only on this information, find the minimum number k of iterations that the CG method (in exact arithmetic) needs to generate an iterate x_k with

$$\left\|A^{-1}b - x_k\right\|_A \le 10^{-12}.$$

(c) Let k be the integer you found in part (b). Compare the flop count of running the CG algorithm presented in class for k iterations with the flop count for solving $Ax = b$ by means of Cholesky factorization. Using only the term with the highest power of n in each of the two flop counts, estimate the matrix size n for which k CG iterations require only one tenth of the arithmetic operations that the Cholesky approach requires.