MAT 226B, Winter 2018

Homework 3

(due by Wednesday, February 21, 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 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

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

(a) Generalize the fast FFT-based 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 FFT-based solver from (a) using Matlab's "`fft`". 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}.

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

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

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

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

of your computed solution. Choose m large enough so that (3) is below 5×10^{-4} and print out the m you have used, together with the corresponding value of (3). Use the following 5 test cases:

(i) $\alpha = 0$, $\beta = 1$, $\gamma = 0.5$;
(ii) $\alpha = 1$, $\beta = 1.5$, $\gamma = 2$;
(iii) $\alpha = 2$, $\beta = 3$, $\gamma = 0.5$;
(iv) $\alpha = 5$, $\beta = 3$, $\gamma = 1$;
(v) $\alpha = 5$, $\beta = 5$, $\gamma = 3$.

Submit printouts of the three-dimensional 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).

2. Consider two-dimensional partial differential equations of the following form:

$$-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2} + \sigma v(x,y) = f(x,y), \quad (x,y) \in R := (0,a) \times (0,b), \qquad u = g(x,y), \quad (x,y) \in \partial R. \tag{4}$$

Here, $a, b, \sigma \in \mathbb{R}$ are constants with $a, b > 0$ and $\sigma \geq 0$, f is a given function on R, and g is a given function on ∂R. In class, we derived a fast FFT-based solver for such problems for the special case $a = b = 1$, $\sigma = 0$, and $g \equiv 0$.

The purpose of this problem is to generalize the solver presented in class to problems of the form (4). To this end, we discretize (4) using grid points

$$(x_j, y_k) := (jh_x, kh_y), \quad j = 0, 1, \ldots, m_x + 1, \quad k = 0, 1, \ldots, m_y + 1,$$

where $m_x, m_y \geq 1$ are integers and

$$h_x := \frac{a}{m_x + 1} \quad \text{and} \quad h_y := \frac{b}{m_y + 1}.$$

The usual centered differences are employed to approximate $\frac{\partial^2 v(x,y)}{\partial x^2}$ and $\frac{\partial^2 v(x,y)}{\partial y^2}$.

(a) Show that the resulting discretization of (4) can be written in the form

$$T_{m_x} V + \alpha V T_{m_y} + \beta V = \tilde{F}, \tag{5}$$

where the entries v_{jk} of the matrix $V \in \mathbb{R}^{m_x \times m_y}$ are approximations to the solution $v(x,y)$ of (4) at the interior grid points:

$$v_{jk} \approx u(x_j, y_k), \quad j = 1, 2, \ldots, m_x, \quad k = 1, 2, \ldots, m_y.$$

In (5), T_{m_x} and T_{m_y} are suitably defined tridiagonal matrices, $\alpha, \beta \in \mathbb{R}$, and $\tilde{F} \in \mathbb{R}^{m_x \times m_y}$. Determine the explicit form of α, β, and $\tilde{F}$.

(b) Generalize the FFT-based algorithm presented in class so that you can use it to efficiently solve (5). How does the number of flops of your algorithm depend on m_x and m_y?

3. Let $t_0, t_1, \ldots, t_{n-1} \in \mathbb{R}$ be given numbers such that the Toeplitz matrix

$$T = \left[t_{|k-j|}\right]_{j,k=1,2,\ldots,n} \in \mathbb{R}^{n \times n}$$

is symmetric positive definite. Let $\mathcal{C}_n$ denote the set all circulant matrices $C \in \mathbb{R}^{n \times n}$. Recall that

$$\|A\|_F = \left(\sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk}^2 \right)^{1/2}$$

is the *Frobenius norm* of $A = \big[a_{jk}\big]_{j,k=1,2,\ldots,n} \in \mathbb{R}^{n\times n}$.

Determine a circulant matrix $C_T \in \mathcal{C}_n$ such that

$$\|C_T - T\|_F = \min_{C\in\mathcal{C}_n} \|C - T\|_F.$$

Show that C_T is unique. Show that C_T is symmetric.

4. In Problem 5 of Homework 1, we developed a fast `fft`-based algorithm for computing matrix-vector products with Toeplitz matrices. This algorithm can be employed to solve symmetric positive definite Toeplitz systems

$$Tx = b, \quad \text{where} \quad T = \big[t_{|k-j|}\big]_{j,k=1,2,\ldots,n} \in \mathbb{R}^{n\times n},\ T \succ 0, \quad \text{and} \quad b \in \mathbb{R}^n, \tag{6}$$

using the CG method.

   (a) Write a Matlab routine based on your algorithm from Problem 5(d) of Homework 1 and Matlab's "`pcg`" for solving linear systems (6) by means of the CG method.

   (b) The circulant matrix C_T from Problem 3 can be used as a left preconditioner for (6): instead of (6), we solve the equivalent linear system

$$C_T^{-1}Tx = C_T^{-1}b \tag{7}$$

   by means of the preconditioned CG method.

   Based on Matlab's "`pcg`", write a Matlab routine for solving linear systems (7) by means of the preconditioned CG method. Note that "`pcg`" requires a routine for solving systems with C_T, or equivalently, computing matrix-vector products with C_T^{-1}. To this end, note that

$$C_T^{-1} = \frac{1}{n}\,\overline{F}\begin{bmatrix} \frac{1}{\lambda_0} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\lambda_{n-1}} \end{bmatrix} F,$$

   where F is the matrix defined in Problem 5 of Homework 1 and the λ_k's are the eigenvalues of C_T. Use the same approach as in Problem 5(b) of Homework 1 to write a Matlab function for computing the matrix-vector product with C_T^{-1} via three calls to Matlab's "`fft`" command.

   (c) Test your Matlab routines from (a) and (b) on Toeplitz matrices T with

$$t_j = \frac{1}{\big(1+\sqrt{j}\big)^p}, \quad j = 0, 1, \ldots, n-1.$$

   Here, $p > 0$ is a parameter.

First, solve systems of size $n = 10$ for the parameter values $p = 1$, $p = 0.1$, and $p = 0.01$. For all three cases, print out your computed solution x.

Second, solve systems of size $n = 10^6$ for the parameter values $p = 1$ and $p = 0.1$. For both cases, print out the entries

$$x(1),\ x(100000),\ x(500000),\ x(700000),\ x(1000000)$$

of your computed solution x.

For all your runs, use the right-hand side

$$b = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n,$$

the initial vector $x_0 = 0 \in \mathbb{R}^n$, and the convergence tolerance $\texttt{tol} = 10^{-9}$ for "`pcg`".

5. Let $A = \big[a_{jk}\big] \in \mathbb{R}^{n\times n}$ be a sparse symmetric positive definite matrix. In class, we discussed an algorithm for computing an incomplete Cholesky factor L of A with a prescribed sparsity pattern E.

   In the following, we use the choice

   $$E = \big\{\, (j,k) \;\big|\; 1 \le k \le j \le n \text{ and } a_{jk} \neq 0 \,\big\}.$$

   Let J and I denote the integer vectors that describe the sparsity pattern E in compressed sparse column (CSC) format.

   (a) Write a Matlab function that efficiently generates the integer vectors J and I for a given sparse matrix A.

   Hint: Recall that internally, Matlab stores sparse matrices in CSC format. One can use "`find(J-K == 0)`" to efficiently construct the pointer vector I. Here, J and K are the vectors of row and column indices produced by "`find(tril(A))`".

   (b) Write a Matlab function that efficiently computes the entries

   $$l_{jk}, \quad (j,k) \in E, \tag{8}$$

   of the incomplete Cholesky factor L of A. The input of your function should be A and the integer vectors J and I from (a), and the output should be a vector V_L that contains the entries (8) in the column-wise order given by J and I.

   (c) Write a Matlab function that uses J, I, and V_L from (b) to efficiently compute the solution of lower-triangular linear systems with coefficient matrix L.

(d) Write a Matlab function that uses J, I, and V_L from (b) to efficiently compute the solution of upper-triangular linear systems with coefficient matrix L^T.

(e) Use Matlab's "`pcg`" to compute the solution of symmetric positive definite linear systems $Ax = b$ using the CG method without preconditioning and with the incomplete Cholesky preconditioner generated by your function from (b). Employ your functions from (c) and (d) for the solution of linear systems with L and L^T.

First, solve the system of size $n = 25$ provided in "`pcg_small.mat`". For both cases, print out the computed solution x, the corresponding relative residual norm

$$\frac{\|b - Ax\|_2}{\|b\|_2}, \tag{9}$$

and the number of CG iterations to reach convergence. Print out the vectors J, I, V_L describing your incomplete Cholesky factor L.

Second, solve the system of size $n = 262144$ provided in "`pcg_large.mat`". For both cases, print out the entries

$$x(1),\ x(10000),\ x(100000),\ x(200000),\ x(262144)$$

of your computed solution x, the corresponding relative residual norm (9), and the number of CG iterations to reach convergence. Print out the first, 10-th, 100-th, 1000-th, and last entries of the vectors J, I, V_L describing your incomplete Cholesky factor L.

For all your runs, use the initial vector

$$x_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n$$

and the convergence tolerance `tol` $= 10^{-9}$ for "`pcg`".