MAT 226B, Winter 2018

Final Project

(due by Friday, March 23, 11:59 pm)

General Instructions

- You are required to submit your final project by uploading a single pdf file to Canvas.
- If at all possible, use a text processing tool (such as LaTeX) for the preparation of your final project. If your submission includes scanned-in hand-written parts, make sure that you write clearly and that you present your solutions in a well-organized fashion. If I cannot read it, I cannot grade it!
- Feel free to discuss the final project with other students, but you do need to submit your own write-up and your own Matlab codes. If there are students with codes or results 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.
- 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
- Include printouts of all the Matlab programs that you have used in this final project.

Consider partial differential equations of the form

$$\begin{aligned} -u_{xx} - u_{yy} &= f, \quad (x,y) \in R := R_1 \cup R_2, \\ u &= g, \quad (x,y) \in \partial R. \end{aligned} \tag{1}$$

Here, R_1 and R_2 are the rectangular regions

$$R_1 := (0,1) \times (0,3) \quad \text{and} \quad R_2 := (a, a+4) \times (b, b+1), \tag{2}$$

where $0 < a < 1$ and $0 \le b \le 2$ are real parameters. Note that for these parameter values, the subregions R_1 and R_2 overlap. The goal of this project is to implement some of the domain decomposition approaches discussed in class, using an FFT-based fast elliptic solver for all solves on the subregions R_1 and R_2.

Part 1: FFT-based fast elliptic solver

Adapt your FFT-based fast elliptic solver from Problem 1 of Homework 3 to rectangular regions of the form $\big(c, c+i_c\big) \times \big(d, d+i_d\big)$, where c, d are real and $i_c, i_d \geq 1$ are integers. Use the same grid spacing $h = h_x = h_y$ in x-direction and y-direction and assume that h is of the form

$$h = \frac{1}{2^m}, \quad \text{where } m \geq 1 \text{ is an integer.} \tag{3}$$

Produce a Matlab implementation of such an adapted FFT-based fast elliptic solver. You are required to use this Matlab implementation for all solves on the subregions R_1 and R_2 in Parts 3 and 4 below.

Part 2: Bilinear interpolation

For the discretized alternating Schwarz method, we need a procedure that sets up the interpolating operator I_R^Γ. The purpose of this part is to develop such a procedure.
Let

$$p_1 := (x_1, y_1), \quad p_2 := (x_1, y_2), \quad p_3 := (x_2, y_1), \quad p_4 := (x_2, y_2)$$

be given points in the xy-plane with $x_1 \neq x_2$ and $y_1 \neq y_2$, and let $z_i \in \mathbb{R}$, $i = 1, 2, 3, 4$. Show that there exists a unique bilinear function of the form

$$b(p) = b(x, y) = c_1 + c_2 x + c_3 y + c_4 xy.$$

where $c_1, c_2, c_3, c_4 \in \mathbb{R}$, such that

$$b(p_i) = z_i, \quad i = 1, 2, 3, 4.$$

For the case that R is a rectangular domain in the xy-plane covered by a regular grid and an artificial boundary Γ lies inside R, outline a procedure that sets up the interpolating operator I_R^Γ for the discretized alternating Schwarz method. Use bilinear interpolation at the four closest grid points for your procedure.

Part 3: Matching grids

In this part, we assume that both parameters a and b in (2) are multiples of the grid spacing h. This guarantees that we have matching grids on R_1 and R_2.

(a) Write Matlab programs that generate the "data" A and b of the discretized version,

$$Av = b,$$

of (1), as well as the submatrices A_1 and A_2 of A corresponding to the subregions R_1 and R_2, respectively. Design your program such that you can use any functions $f : R \mapsto \mathbb{R}$ and $g : \partial R \mapsto \mathbb{R}$.

(b) For each of the following two domain decomposition methods, write a Matlab program that solves the discretized problem $Av = b$:

(1) Multiplicative Schwarz method;

(2) CG method with additive Schwarz preconditioning.

The program for each of these methods should allow as inputs an arbitrary initial guess $v^{(0)}$ for the solution of $Av = b$, a "small" convergence tolerance `tol` for the domain decomposition method, and a large integer $n_{\max}$ to be used as a safeguard against an excessive number of iterations of the domain decomposition method.

For method (2), apply the preconditioner M from the left, i.e., solve

$$M^{-1}Av = M^{-1}b,$$

and employ Matlab's "`pcg`" function.

(c) To test each of your programs, choose the functions f and g such that

$$u(x, y) = y^{\alpha} \sin(\beta\pi x) \cos(\gamma\pi y), \tag{4}$$

where $\alpha \geq 0$ and β, $\gamma > 0$ are parameters, is the exact solution of problem (1). Note that this family of functions was also used in Problem 1 of Homework 3.

Run the following 6 cases:

(i) $a = \frac{1}{2}$, $b = 0$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(ii) $a = \frac{1}{2}$, $b = 0$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$;
(iii) $a = \frac{1}{4}$, $b = 1$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(iv) $a = \frac{1}{4}$, $b = 1$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$;
(v) $a = \frac{1}{2}$, $b = 2$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(vi) $a = \frac{1}{2}$, $b = 2$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$.

Here, a, b and α, β, γ are the parameters in (2) and (4), respectively.

For each of these 6 cases, run both methods (1) and (2). Thus there is a total of 12 runs.

For each of your runs, determine the relative maximal error of your computed approximate solution at the grid points:

$$\frac{\max_{j,k}\big|v_{jk} - u(x_j, y_k)\big|}{\max_{j,k}\big|u(x_j, y_k)\big|}. \tag{5}$$

Here, v_{jk} denotes the computed approximate solution at grid point (x_j, y_k), and the maximum is taken over all grid points. For each run, choose m in the grid spacing (3) large enough and the convergence tolerance `tol` small enough so that the relative

maximal error (5) is below 5×10^{-4}. For each run, print out the m you have used, the corresponding value of (5), and the total numbers of solves on R_1 and solves on R_2 used in the domain decomposition method. For each run, submit a three-dimensional plot of the computed solution v_{jk}.

Comment on your results and the efficiency of methods (1) and (2).

Part 4: Nonmatching grids

In this part, we assume that neither a nor b in (2) are multiples of the spacing h, so that we do not have matching grids on R_1 and R_2.

(a) Write Matlab programs that generate the matrices

$$A_1,\ A_2,\ B_{\Gamma_1},\ B_{\Gamma_2},\ I_{R_2}^{\Gamma_1},\ I_{R_1}^{\Gamma_2}$$

and the vectors

$$b_1 := h^2 f_1 - B_{\Sigma_1} g_{\Sigma_1},\ b_2 := h^2 f_2 - B_{\Sigma_2} g_{\Sigma_2}$$

that are needed to set up the linear system

$$\begin{bmatrix} A_1 & B_{\Gamma_1} I_{R_2}^{\Gamma_1} \\ B_{\Gamma_2} I_{R_1}^{\Gamma_2} & A_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{6}$$

which underlies the discretized alternating Schwarz method for two subregions R_1 and R_2. Employ bilinear interpolation at the four nearest grid points, as outlined in Part 2 above, to set up the interpolation matrices $I_{R_2}^{\Gamma_1}$ and $I_{R_1}^{\Gamma_2}$.

(b) For each of the following two domain decomposition methods, write a Matlab program that solves the discretized problem (6):

(1) Discretized alternating Schwarz method;

(2) GMRES with alternating Schwarz as preconditioner.

The program for method (1) should allow as input an arbitrary initial guess $v_1^{(0)}$; The program for method (2) should allow as inputs arbitrary initial guesses for $v_1^{(0)}$ and $v_2^{(0)}$. Both programs should allow as inputs a "small" convergence tolerance `tol` for the domain decomposition method, and a large integer $n_{\max}$ to be used as a safeguard against an excessive number of iterations of the domain decomposition method.

For method (2), apply the preconditioner M from the left, i.e., solve

$$M^{-1} A v = M^{-1} b,$$

and employ Matlab's "`gmres`" function. Compute the required matrix-vector products with $A' = M^{-1}A$ using the approach described in class. Note that the main work for each product is one solve on R_1 and one solve on R_2.

(c) To test each of your programs, choose the same functions f and g as in Part 3.

Run the following 6 cases:

(i) $a = \frac{2}{5}$, $b = \frac{1}{5}$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(ii) $a = \frac{2}{5}$, $b = \frac{1}{5}$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$;
(iii) $a = \frac{1}{5}$, $b = \frac{6}{5}$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(iv) $a = \frac{1}{5}$, $b = \frac{6}{5}$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$;
(v) $a = \frac{2}{5}$, $b = \frac{9}{5}$, $\alpha = 0$, $\beta = 1$, $\gamma = \frac{1}{2}$;
(vi) $a = \frac{2}{5}$, $b = \frac{9}{5}$, $\alpha = 2$, $\beta = \frac{7}{2}$, $\gamma = 2$.

Here, a, b and α, β, γ are the parameters in (2) and (4), respectively.

For each of these 6 cases, run both methods (1) and (2). Thus there is a total of 12 runs.

For each of your runs, choose m in the grid spacing (3) large enough and the convergence tolerance `tol` small enough so that the relative maximal error (5) is below 5×10^{-4}. For each run, print out the m you have used, the corresponding value of (5), and the total numbers of solves on R_1 and solves on R_2 used in the domain decomposition method. For each run, submit a three-dimensional plot of the computed solution v_{jk}.

Comment on your results and the efficiency of methods (1) and (2).