Domain decomposition for PDEs

Luca Formaggia

November 10, 2024

1 Introduction

Domain decomposition techniques are a common tool for implementing parallelization to solve a problem governed by partial differential equations. There are several techniques; notable references are [?, ?]. Here we deal with the simplest one, the Schwarz method, which may be implemented as *solver* or as a *preconditioner*, see also [?].

Let us start from an example in one dimension, to give a general idea. We want to solve

$$\begin{cases} -\mu \dfrac{d^2}{dx^2}u(x) + cu(x) = f(x), \ x \in (a,b) \\ u(a) = u_a, \quad u(b) = u_b, \end{cases} \tag{1}$$

with $\mu > 0$, $c \geq 0$ and f a given function.

Let us first consider the method in the continuous setting. The Schwarz method (as a solver) starts from a partition of the interval (a,b) in a set of N non-overlapping subdomains

$$(\tilde{a}_i, \tilde{b}_i), \quad i = 1, \dots N,$$

where $\tilde{a}_1 = a$, $\tilde{b}_N = b$, $\tilde{b}_i = \tilde{a}_{i+1}$, for $i = 1, \dots, N-1$ and δ and their extension to N *partially overlapping subdomains* with overlap $\delta > 0$,

$$(a_i, b_i), \quad i = 1, \dots N,$$

where $a_1 = a$, $b_N = b$, $a_i = \tilde{a}_i - \delta/2$, for $i = 2, \dots N$, and $b_i = \tilde{b}_i + \delta/2$, for $i = 1, \dots N-1$, so that $b_i - a_{i+1} = \delta$, for $i = 1, \dots, N-1$.

Consider the *restriction* operators R_i that, given any function g defined in $[a,b]$ return its restriction $R_i g$ in $[a_i, b_i]$:

$$\begin{cases} R_i g(x) = g(x), \quad x \in [a_i, b_i] \\ R_i g(x) = 0, \quad x \notin [a_i, b_i] \end{cases}$$

and a *prolongation operator* $P_i : (\tilde{a}_i, \tilde{b}_i) \to (a,b)$:

$$\begin{cases} P_i g_i(x) = g_i(x), \quad x \in (\tilde{a}_i, \tilde{b}_i) \\ P_i g_i(x) = 0, \quad \text{otherwise} \end{cases}$$

With those definitions the Schwarz iteration reads:

- Consider an initial set of functions $u_i^{(0)}$ defined in each extended subdomain (a_i, b_i). They can be rather arbitrary.
  - for $k = 1, \dots$, solve (in parallel) on each (extended) subdomain (a_i, b_i) the problem

$$\begin{cases} -\mu \dfrac{d^2}{dx^2}u_i^{(k)} + cu_i^{(k)} = R_i f \\ u_i^{(k)}(a_i) = u_{i-1}^{(k-1)}(a_i) \text{ if } i \neq 1, \ \text{else } u_1^{(k)}(a_1) = u_a \\ u_i^{(k)}(b_i) = u_{i+1}^{(k-1)}(b_i) \text{ if } i \neq N, \ \text{else } u_N^{(k)}(b_N) = u_b \end{cases} \qquad i = 1, \dots N$$

  - Check if we are at convergence:
$$\max_{i=1}^{N} \|u_i^{(k)} - u_i^{(k-1)}\| \leq \epsilon$$
for a given $\epsilon > 0$ and appropriate norm $\|\cdot\|$.

- Reconstruct full solution
$$u = \sum_{i=1}^{N} P_i u_i^{(k)}.$$

In practice, we solve a Dirichlet problem in each subdomain with the boundary condition taken from the adjacent subdomain at the previous iteration, apart from the actual domain boundary where we need to enforce the boundary condition of the global problem.

Remark. The theory says that the method converges, and the convergence depends on the number of subdomains and the length of the overlap. Note that the prolongation operator acts on the non-expanded domain. Technically, we have something to do with the two values at the interface between two subdomains. Normally, one takes an average. In fact, a different prolongation operator may be devised that averages the values over the whole overlapped region.

Remark. Clearly, you do not need all $u_i^{(k)}$, only those in the last two iterations. The problem is highly parallelizable; the only communication is at the boundary of each subdomain.

1.1 The algebraic setting

Let's now reformulate problem (1) in a discrete setting using finite differences. We will then consider the equivalent Schwarz formulation for the discrete problem. To this aim, we first consider a subdivision of the interval $[a, b]$ into n elements $[x_k, x_{k+1}]$, for $k = 0, \dots, n-1$ of constant length $h = x_{k+1} - x_k$, $x_0 = a$ and $x_n = b$. A possible finite-difference formulation reads: Find u_k, $k = 0, \dots n$, where u_k approximates $u(x_k)$, that satisfies

$$\begin{cases} \mu \dfrac{-u_{k-1} + 2u_k - u_{k+1}}{h^2} + cu_k = f(x_k), \quad k = 1, \dots n-1 \\ u_0 = u_a, \quad u_n = u_b \end{cases} \tag{2}$$

which can be written as
$$A\mathbf{u} = \mathbf{b},$$
with $\mathbf{u} = [u_0, \dots, u_n]^T$, $\mathbf{b} = [u_a, f(x_1), \dots, f(x_{n-1}), u_b]$ and

$$A = \begin{bmatrix} 1 & 0 & \dots & & & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + c & -\frac{1}{h^2} & 0 & \dots & & \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + c & -\frac{1}{h^2} & 0 & \dots & \\ & \vdots & \vdots & \vdots & \dots & 0 & 1 \end{bmatrix} \tag{3}$$

This system forms the basis for the discrete Schwarz methods.

2 Schwarz solver (discrete version)

The subdivision in subdomains in the discrete setting corresponds to partitions of the node indexes. If we indicate with the index j the local node index in each partition i, with $j = 0, \dots n_i - 1$, and with k the Schwarz iteration index, it is clear that in the discrete setting the local problems will take the form

$$\begin{cases} \mu \dfrac{-u_{i,j-1}^{(k)} + 2u_{i,j}^{(k)} - u_{i,j+1}^{(k)}}{h^2} + cu_{i,j}^{(k)} = f_j, \quad j = 1, \dots n_i - 2 \\ u_{i,0}^{(k)} = u_{i-1,n_{i-1}-1-l}^{(k-1)}, \text{ if } i \neq 1, \text{ else } u_{1,0}^{(k} = u_a \\ u_{i,n_i}^{(k)} = u_{i+1,l}^{(k-1)}, \text{ if } i \neq N, \text{ else } u_{N,n_N-1}^{(k)} = u_b \end{cases} \tag{4}$$

Here, $u_{i,j}^{(k)}$, $j = 0, \ldots, n_i$, are the local unknowns associated to the i-th subdomain, $l > 0$ is an integer representing *overlap* as number of mesh elements overlapping. We can define a local-to-global map $\sigma_i(j)$ that returns the *global index* of the local node j of partition i. Then $f_j = f(x_{\sigma_i(j)})$. The partitions of the mesh nodes may be built by first partitioning the indexes $1 \ldots n$ into N non-overlapping partitions of adjacent nodes and then adding l nodes on the two ends (apart from the two boundary partitions that have to respect the actual domain boundary). The test of convergence may be performed by computing the 2-norm between two successive iterates, and the final global solution is easily reconstructed considering only the indexes of the non-overlapping partitions.

$$u_{\sigma_i(j)} = u_{i,j}^{(k)}, \quad j = l, n_j - 1 - l, \quad i = 1, \ldots, N$$

with the obvious modification for the adjacent partitions to the boundary.

In the discrete setting, we can also construct *projection operators* P_i, which in this case are matrices with n_i rows and n columns. Each row j has elements equal to zero except for the element $\sigma_i(j)$, which is equal to 1. Then, given a vector $\mathbf{b} \in \mathbb{R}^n$ representing quantities in x_i, the vector $\mathbf{b}_i = P_i \mathbf{b}$ is its local projection on the i-th partition.

It may be noted that, apart from some changes at the first and last row to impose the local boundary conditions, problems (4) may be derived from (2) as

$$A_i \mathbf{u}_i = \mathbf{b}_i, \tag{5}$$

where $\mathbf{b}_i = P_i \mathbf{b}$ and $A_i = P_i A P_i^T$.

3 Schwarz iteration as preconditioner

Schwarz iteration may be used as a preconditioner. Let us indicate with $S\mathbf{b}$ the solution of (5) (a single iteration of Schwarz) with *homogeneous boundary conditions* and $\mathbf{b}$ as the right-hand side. For simplicity, let us consider a preconditioned Richardson iteration (but any classic iterative scheme may be used). Its non-preconditioned version computes approximations of our problem starting from an initial iterate $\mathbf{u}^{(0)}$, and performs, for $k = 0, 1, \ldots$,

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1)} + \alpha \mathbf{r}^{(k)}, \tag{6}$$

where $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{u}^{(k)}$ is the residual. You iterate until the residual is sufficiently small (when it converges,...). Here, $\alpha > 0$ is a suitably chosen parameter (see any good book on numerical methods for details).

A preconditioned version using Schwarz is given by

- Compute the preconditioned residual by solving (possible in parallel)

$$S\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$$

- perform the iteration with the precomputed residual

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1)} + \alpha \mathbf{z}^{(k)}.$$

The idea is that at the global level you do only matrix-vector multiplications, which are easily parallelizable, and to speed up the convergence you use a parallel preconditioner exploiting the fact that if the subdomains are sufficiently small, the solution of the problem in each subdomain is cheap.

As it is, the preconditioner is not so efficient, since it is too "local". It can be improved by adding a "coarse correction," see the provided papers. The simplest way to build the correction is to take the central node of each subdomain and build the discrete problem with just those nodes.

4 What do you have to do

For the hands-on, you should

- implement s class to solve the problem (1). Note that you have a tridiagonal system so you can use the Thomas algorithm given in the Example. But you may choose instead to rely on the Eigen library.
- Implement the Schwarz iterator as a solver, with a class that is composed.
- implement the parallel version in MPI or OpenMP.
- have a way to shod the results (choose the graphic library or tool you prefer).

Possible further developments (maybe an exam project):

- Extend to two dimensional problems
- Write the preconditioner version and use it as preconditioner for the iterative solvers contained in LinearAlgebra/IML_Eigen in the Example repo (you have to use the Eigen library for the linear algebra).
- Use algebraic techniques to build the course grid operator.
- Implement on a GPU.