TMA4280: Introduction to Supercomputing

Numerical solution of the Poisson problem: domain decomposition methods

Spring 2012

©Einar M. Rønquist
Department of Mathematical Sciences
NTNU, N-7491 Trondheim, Norway

1 Domain decomposition methods

Consider the computational domain depicted in Figure 1.

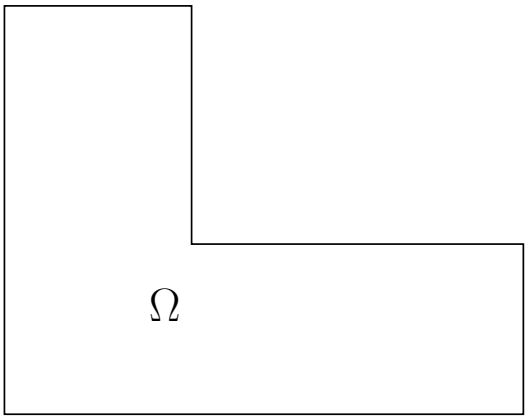

Figure 1: An L-shaped domain Ω.

We decompose Ω into three subdomains as depicted in Figure 2.

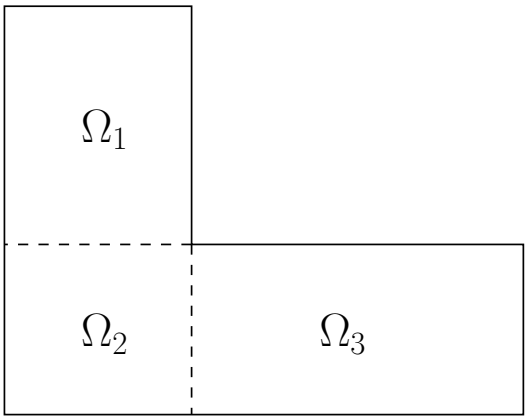

Figure 2: A decomposition of Ω into three rectangular subdomains.

Our objective is here to solve the Poisson problem defined over Ω. We have already looked at how this can be done using the conjugate gradient method in combination with domain decomposition and parallel processing. We will here look at yet another alternative way to solve the Poisson problem, involving the solution of local Poisson subproblems over Ω_i, $i = 1, 2, 3$. In addition, since each subdomain Ω_i here represents a rectangle discretized using a structured grid, we can also use fast solvers for each individual subproblem.

Before we proceed with this Poisson problem, we consider the simpler problem

$$\begin{aligned} -\nabla^2 u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where the domain Ω is depicted in Figure 3.

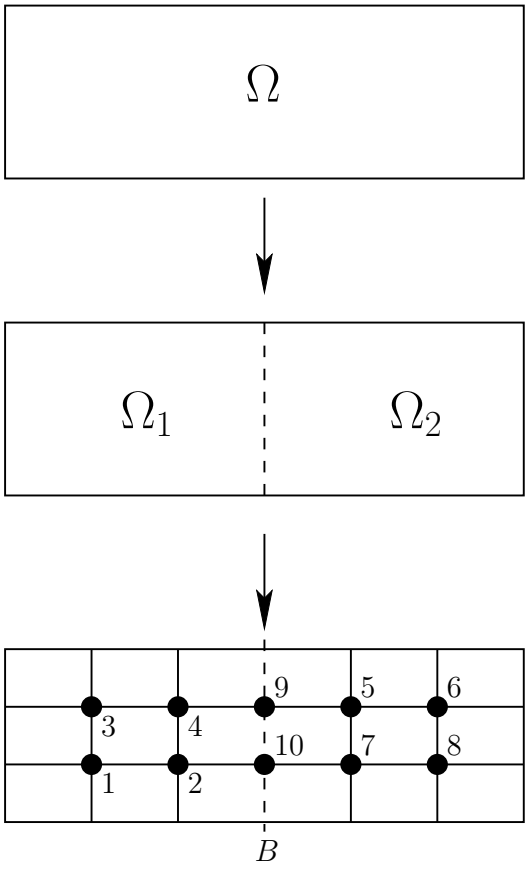

Figure 3: A rectangular domain Ω decomposed into two equal subdomains and discretized using finite differences. A global numbering of all the internal nodes is given.

Using the 5-point stencil and the global numbering depicted in Figure 3, we can express the algebraic system of equations as

$$\underline{A}\,\underline{u} = \underline{f}, \tag{1}$$

where the matrix $\underline{A}$ (the discrete Laplace operator), the solution vector $\underline{u}$, and the right hand side $\underline{f}$ are detailed in Figures 4, 5, and 6, respectively.

$$\underline{\underline{A}} = \frac{1}{h^2} \left[\begin{array}{cccc|cccc|cc} 4 & -1 & -1 & 0 & & & & & 0 & 0 \\ -1 & 4 & 0 & -1 & & & & & 0 & -1 \\ -1 & 0 & 4 & -1 & & \mathbf{0} & & & 0 & 0 \\ 0 & -1 & -1 & 4 & & & & & -1 & 0 \\ \hline & & & & 4 & -1 & -1 & 0 & -1 & 0 \\ & & & & -1 & 4 & 0 & -1 & 0 & 0 \\ & \mathbf{0} & & & -1 & 0 & 4 & -1 & 0 & -1 \\ & & & & 0 & -1 & -1 & 4 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 4 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right]$$

Figure 4: The matrix $\underline{\underline{A}}$ (the discrete Laplace operator) resulting from using the finite difference grid depicted in Figure 3. As usual, h is the grid spacing in each spatial direction.

$$\underline{u} = \left[\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \hline u_5 \\ u_6 \\ u_7 \\ u_8 \\ \hline u_9 \\ u_{10} \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \underline{u}_1 \\ \\ \\ \\ \underline{u}_2 \\ \\ \\ \underline{u}_B \end{array}$$

Figure 5: The solution vector $\underline{u}$ using the global numbering depicted in Figure 3. Here, the subvector $\underline{u}_1$ represents the degrees-of-freedom in the interior of subdomain Ω_1, the subvector $\underline{u}_2$ represents the degrees-of-freedom in the interior of subdomain Ω_2, while the subvector $\underline{u}_B$ represents the degrees-of-freedom along the subdomain interface.

$$\underline{f} = \begin{bmatrix} \underline{f}_1 \\ \underline{f}_2 \\ \underline{f}_B \end{bmatrix} \in \mathbb{R}^{10}$$

Figure 6: The right hand side $\underline{f}$ using the global numbering depicted in Figure 3.

The system of equations for this problem with two subdomains can thus be expressed succinctly in terms of *matrix blocks* (see Figure 4) and subvectors as

$$\begin{bmatrix} \underline{A}_{11} & \underline{0} & \underline{A}_{1B} \\ \underline{0} & \underline{A}_{22} & \underline{A}_{2B} \\ \underline{A}_{B1} & \underline{A}_{B2} & \underline{A}_{BB} \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \underline{u}_B \end{bmatrix} = \begin{bmatrix} \underline{f}_1 \\ \underline{f}_2 \\ \underline{f}_B \end{bmatrix} \tag{2}$$

where the blocks

$\underline{A}_{ii}, \quad i = 1, 2$ — represent couplings between degrees-of freedom in the interior of Ω_i,

$\underline{A}_{iB}, \quad i = 1, 2$ and $\underline{A}_{Bi}, \quad i = 1, 2$ — couplings between degrees-of-freedom in the interior of Ω_i and the interface B (note: $\underline{A}_{iB} = \underline{A}_{Bi}^T$), and

$\underline{A}_{BB}$ — couplings between degrees-of-freedom along the interface B.

First, assume that $\underline{u}_B$ is known. Since

$$\underline{A}_{11}\underline{u}_1 + \underline{A}_{1B}\underline{u}_B = \underline{f}_1$$

we get

$$\underline{A}_{11}\underline{u}_1 = \underline{f}_1 - \underbrace{\underline{A}_{1B}\underline{u}_B}_{\substack{\text{similar to the} \\ \text{treatment of} \\ \text{non-homogeneous} \\ \text{boundary conditions} \\ (u \neq 0 \text{ on } \partial\Omega_1)}}$$

Hence,

$$\underline{u}_1 = \underline{A}_{11}^{-1}(\underline{f}_1 - \underline{A}_{1B}\underline{u}_B). \tag{3}$$

Similarly,

$$\underline{u}_2 = \underline{A}_{22}^{-1}(\underline{f}_2 - \underline{A}_{2B}\underline{u}_B). \tag{4}$$

Even though $\underline{u}_B$ is unknown, we can still formally express $\underline{u}_1$ and $\underline{u}_2$ as expressed in (3) and (4).

From (2), the discrete equations for $\underline{u}_B$ are:

$$\underline{A}_{B1}\underline{u}_1 + \underline{A}_{B2}\underline{u}_2 + \underline{A}_{BB}\underline{u}_B = \underline{f}_B. \tag{5}$$

Eliminating $\underline{u}_1$ and $\underline{u}_2$ from (5) by using (3) and (4) gives

$$\underline{A}_{B1}\underline{A}_{11}^{-1}(\underline{f}_1 - \underline{A}_{1B}\underline{u}_B) + \underline{A}_{B2}\underline{A}_{22}^{-1}(\underline{f}_2 - \underline{A}_{2B}\underline{u}_B) + \underline{A}_{BB}\underline{u}_B = \underline{f}_B.$$

Rearranging the terms gives:

$$[\,\underline{A}_{BB} - \underline{A}_{B1}\underline{A}_{11}^{-1}\underline{A}_{1B} - \underline{A}_{B2}\underline{A}_{22}^{-1}\underline{A}_{2B}\,]\,\underline{u}_B = \underline{f}_B - \underline{A}_{B1}\underline{A}_{11}^{-1}\underline{f}_1 - \underline{A}_{B2}\underline{A}_{22}^{-1}\underline{f}_2.$$

This system can also be expressed as

$$\boxed{\hat{\underline{A}}\,\underline{u}_B = \hat{\underline{f}}} \tag{6}$$

where

$$\hat{\underline{A}} = \underline{A}_{BB} - \underline{A}_{B1}\underline{A}_{11}^{-1}\underline{A}_{1B} - \underline{A}_{B2}\underline{A}_{22}^{-1}\underline{A}_{2B}, \tag{7}$$

and

$$\hat{\underline{f}} = \underline{f}_B - \underline{A}_{B1}\underline{A}_{11}^{-1}\underline{f}_1 - \underline{A}_{B2}\underline{A}_{22}^{-1}\underline{f}_2. \tag{8}$$

The matrix $\hat{\underline{A}}$ is called the *Schur complement*. Similar to $\underline{A}$, the matrix $\hat{A}$ is also symmetric and positive definite.

Note that (6) is a system for the unknowns along the subdomain interface, and thus $\dim(\hat{\underline{A}}) \ll \dim(\underline{A})$.

Assume that we can solve (6) for $\underline{u}_B$. We can then immediately compute $\underline{u}_1$ and $\underline{u}_2$ completely decoupled (and concurrently) from (3) and (4).

If Ω_1 and Ω_2 are simple, rectangular subdomains, we can even solve these subsystems using our fast $\mathcal{O}(n^2 \log n)$ FFT algorithm for $\mathcal{O}(n^2)$ unknowns. However, in principle, we can use any solver we wish to compute $\underline{u}_1$ and $\underline{u}_2$.

The natural question now is how we should solve the system (6). Since $\hat{\underline{A}}$ is symmetric and postive definite, we can also here use the conjugate gradient

method. A key difference compared to applying the conjugate gradient method directly to (1) is that the matrix-vector product that we need to perform at each conjugate gradient iteration will now be of the form $\underline{\hat{y}} = \underline{\hat{A}}\,\underline{\hat{x}}$ instead of $\underline{y} = \underline{A}\,\underline{x}$. Hence, each matrix-vector product will now imply the solution of two subdomain problems; see (7). Another difference compared to our earlier discussion of the conjugate gradient method is that the condition number of $\underline{\hat{A}}$ is smaller than the condition number of $\underline{A}$, and thus the number of iterations for the interface system (6) can be expected to be lower than for the full system (1).

The above approach can readily be generalized to more than two subdomains. Consider for example the problem depicted in Figure 1 and Figure 2.

We can now group the unknowns as

$$\underline{u} = \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \underline{u}_3 \\ \underline{u}_B \end{bmatrix}$$

Note that the interface degrees-of-freedom, $\underline{u}_B$, can here be split into two parts

$$\underline{u}_B = \begin{bmatrix} \underline{u}_B^{(12)} \\ \underline{u}_B^{(23)} \end{bmatrix} \tag{9}$$

where $\underline{u}_B^{(12)}$ represents the interface degrees-of-freedom between Ω_1 and Ω_2, and $\underline{u}_B^{(23)}$ represents the interface degrees-of-freedom between Ω_2 and Ω_3.

Again, we can derive a system for the unknowns $\underline{u}_B$ along the interface on the form

$$\underline{\hat{A}}\,\underline{u}_B = \underline{\hat{f}}, \tag{10}$$

but now with

$$\underline{\hat{A}} = \underline{A}_{BB} - \sum_{i=1}^{3} \underline{A}_{Bi}\underline{A}_{ii}^{-1}\underline{A}_{iB}, \tag{11}$$

$$\underline{\hat{f}} = \underline{f}_B - \sum_{i=1}^{3} \underline{A}_{Bi}\underline{A}_{ii}^{-1}\underline{f}_i. \tag{12}$$

The approach can also be generalized to unstructured grids, for example using the finite element method; see Figure 7.

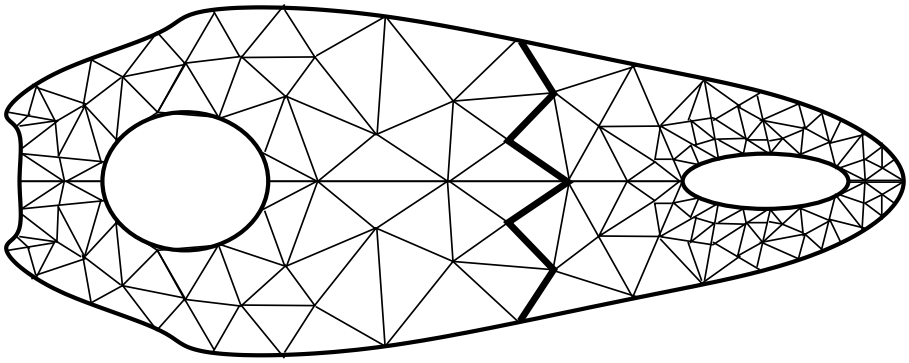

Figure 7: A finite element triangulation and a partitioning into two subdomains (the bold line represents the subdomain interface).

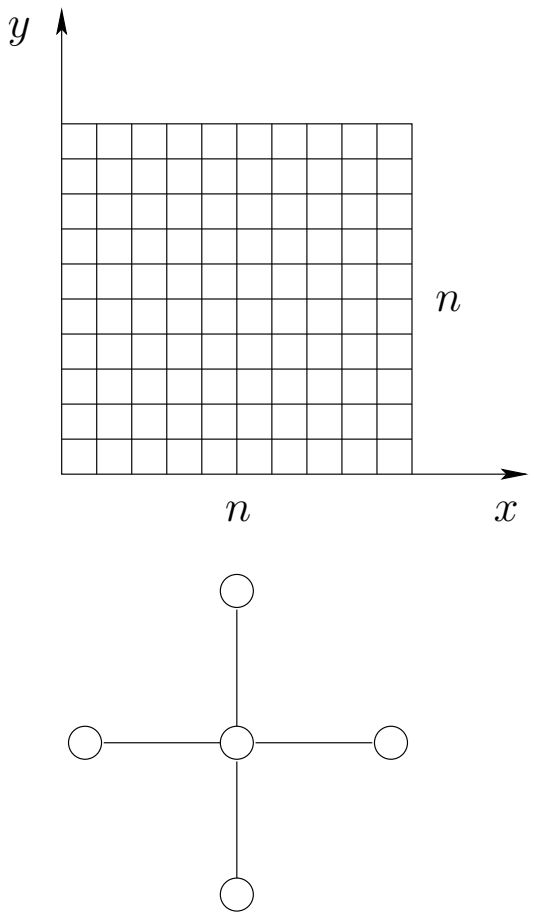

Figure 8: Computational domain and discretization using finite differences.

2 Solution methods: a comparison

Consider the Poisson model problem

- in a simple two-dimensional domain, Ω (see Figure 8),
- with homogeneous boundary conditions $u = 0$ on $\partial\Omega$,
- spatial discretization using a structured grid (see Figure 8),
- a five-point finite difference stencil (see Figure 8),
- various solution methods for the resulting system of algebraic equations.

In this section, we compare the computational complexity and the memory requirement for a few solution methods when solving the linear system of algebraic equations

$$\underline{A}\,\underline{u} = \underline{f}, \tag{13}$$

where $\underline{A}$ represents the discrete Laplace operator and $\underline{f}$ is a known right hand side. Here,

- $\dim(\underline{A}) = N = (n-1)^2 \sim \mathcal{O}(n^2)$
- $\underline{A}$ is symmetric ($\underline{A} = \underline{A}^T$)
- $\underline{A}$ is positive definite ($\underline{v}^T \underline{A}\underline{v} > 0$ for all $\underline{v} \in I\!R^N$, $\underline{v} \neq \underline{0}$)
- $\underline{A}$ is non-singular, i.e., invertible ($\det(\underline{A}) \neq 0$)

Notation:

$$\begin{aligned} N &= \text{the number of unknowns} = (n-1)^2 \\ \mathcal{N}_{op} &= \text{the number of floating point operations} \\ \mathcal{M} &= \text{the memory requirement in bytes} \end{aligned}$$

Below are a few alternative solution methods for (13):

2.1 Gaussian elimination

This is a direct method meaning that the solution $\underline{u}$ in (13) can be found in a fixed, predictable number of floating point operations. It is a general and robust method, but costly for large systems. Assuming the whole matrix is stored, we have

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(N^3) \sim \mathcal{O}(n^6), \\ \mathcal{M} &\sim \mathcal{O}(N^2) \sim \mathcal{O}(n^4). \end{aligned}$$

2.2 LU-factorization using a banded approach

Here we decompose the matrix $\underline{A}$ as

$$\underline{A} = \underline{L}\,\underline{U},$$

where L is a lower triangular matrix and $\underline{U}$ is an upper triangular matrix. We also use a natural ordering of the unknowns in order to exploit the resulting band structure; see Figure 2. The band width b will be approximately equal to $2n$ (why?). For this direct solution method, we have

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(N\,b^2) \sim \mathcal{O}(n^4), \\ \mathcal{M} &\sim \mathcal{O}(N\,b\,) \sim \mathcal{O}(n^3). \end{aligned}$$

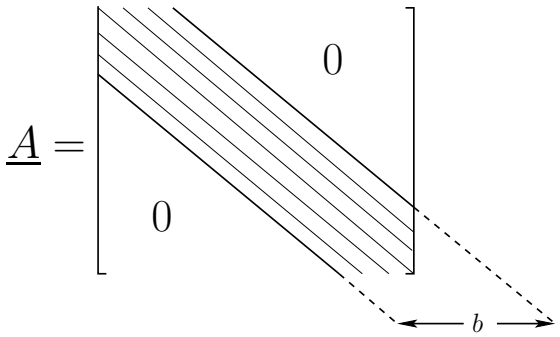

Figure 9: Band structure due to a natural ordering of the unknowns.

2.3 Jacobi iteration

Jacobi iteration is one of the simplest iterative methods that can be used to solve (13). Each iteration is based on an update of the current iterate $\underline{u}^k$ according to the simple update scheme

$$u_i^{k+1} = \frac{1}{a_{ii}}(f_i - \sum_{j \neq i} a_{ij}\, u_j^k), \qquad \forall i = 1, \ldots, N.$$

Here, a_{ij} denotes a matrix element in $\underline{A}$, and u_i^k denotes the i-th element in the vector $\underline{u}^k$ at the k-th iteration.

Assume that we iterate until the residual (the difference between the left hand side and the right hand side of (13)) is ε. Assume that

$$\varepsilon \sim \mathcal{O}(h^2) \sim \mathcal{O}(1/n^2).$$

This means that we iterate to the level of the discretization error. Then it can be shown that the number of iterations, $\mathcal{N}_{iter}$, scales as $\mathcal{N}_{iter} \sim \mathcal{O}(n^2 \log n)$. The work per iteration is $\mathcal{O}(n^2)$ (why?). We assume here that we only store the nonzero matrix elements in $\underline{A}$. For this iterative solution method, we thus have

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(n^4 \log n), \\ \mathcal{M} &\sim \mathcal{O}(n^2). \end{aligned}$$

Note that the memory requirement for this method is optimal (why?).

2.4 Diagonalization based on matrix-matrix products

We refer to the earlier handouts on this method; in summary

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(n^3), \\ \mathcal{M} &\sim \mathcal{O}(n^2). \end{aligned}$$

Again, the memory requirement is optimal.

2.5 Diagonalization based on DST and FFT

We refer to the earlier handouts on this direct solution method; in summary

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(n^2 \log n), \\ \mathcal{M} &\sim \mathcal{O}(n^2). \end{aligned}$$

The memory requirement is optimal and the computational complexity is approximately optimal since, per degree-of-freedom,

$$\begin{aligned} \mathcal{N}_{op}/N &\sim \mathcal{O}(\log n), \\ \mathcal{M}/N &\sim \mathcal{O}(1). \end{aligned}$$

This method is very fast, but has limited applicability.

2.6 The conjugate gradient method

We refer to the earlier handout on this iterative solution method; in summary

$$\begin{aligned} \mathcal{N}_{op} &\sim \mathcal{O}(n^3), \\ \mathcal{M} &\sim \mathcal{O}(n^2). \end{aligned}$$

The memory requirement is optimal, while the computational complexity is not quite scalable. However, a key advantage of this method is that it is applicable to Poisson problems in very general domains and various types of discretization methods (e.g., unstructured grids).

2.7 Summary

The table below gives a comparison of various solution methods for (13) in terms of computational complexity and memory requirement (for $(n-1)^2$ unknowns).

Solution method	Classification	$\mathcal{M}$	$\mathcal{N}_{op}$
Gaussian elimination (full LU)	Direct	$\mathcal{O}(n^4)$	$\mathcal{O}(n^6)$
Banded LU	Direct	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$
Jacobi iteration	Iterative	$\mathcal{O}(n^2)$	$\mathcal{O}(n^4 \log n)$
The conjugate gradient method	Iterative	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Diagonalization (MxM)	Direct	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Diagonalization (FST)	Direct	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \log n)$

Note that the fastest method is the most restrictive in terms of applicability, while the slowest methods (Gaussian elimination) are the most general and the most robust methods. Also note that there exist other (more advanced) solution methods which are general, robust, and with an approximately scalable complexity.