Thirteenth International Conference on Domain Decomposition Methods
Editors: N. Debit, M.Garbey, R. Hoppe, J. Périaux, D. Keyes, Y. Kuznetsov

40 A Hierarchical Domain Decomposition Method with Low Communication Overhead

M. Israeli[1], E. Braverman[2], A. Averbuch[3]

1 Introduction

We present a low communication, non-iterative algorithm for a high order (spectral) solution of the Poisson equation. The domain is decomposed into nonoverlapping subdomains. Particular solutions are found in subdomains and subsequently hierarchically matched, such that only the solution in the adjacent subdomains are coupled at each matching step, then these joint subdomains are matched etc. If originally we had 2^{2k} subdomains, after k steps we obtain a smooth global solution.

Implicit discretization of time dependent problems in computational physics, semiconductor device simulation, electromigration and fluid dynamics often gives rise to equations of Poisson and modified Helmholtz type. Thus, fast and accurate methods for elliptic equations are important for such applications.

We solve the Poisson equation

$$\Delta u = f(x,y) \ \text{ in } \ \Omega \tag{1}$$

or the modified Helmholtz equation

$$\Delta u - \lambda^2 u = f(x,y) \ \text{ in } \ \Omega \tag{2}$$

in the rectangular/square domain $\Omega = [0,L] \times [0,K]$ with Dirichlet

$$u = \Phi(x,y) \ \text{ on } \ \partial\Omega \tag{3}$$

boundary conditions by the Domain Decomposition (DD) methods.

An algorithm for a fast solution of the Poisson equation by decomposition of the domain into square domains and the subsequent matching of these solutions by the fast multipole method was developed in [GL96]. Previously [ABI00] we adopted a DD method where the equation was solved in each subdomain with assumed boundary conditions, resulting in jumps in function or derivative on subdomain boundaries. The solution in each rectangular domain is fast and accurate and is based on the algorithm developed in [AIV97, AIV98]. The jumps at the interfaces were removed by the introduction of singularity layers. In order to account for the global effect of these layers we had to compute the influence of each layer on each subdomain boundary. In order to alleviate this heavy computational task we took into account the decay or smoothing out of the influence as a function of the distance from the layer. To

[1]Technion, Computer Science Dept., Haifa 32000, Israel, israeli@cs.technion.ac.il. The research of the first author was supported by the VPR fund for promotion of research at the Technion.

[2]Technion, Computer Science Dept., Haifa 32000, Israel, maelena@csd.technion.ac.il; now on leave in Yale University, Dept. of Math., 10 Hillhouse Ave., New Haven, CT 06520, USA, braverm@cyndra.cs.yale.edu

[3]Tel Aviv University, School of Math. Sciences, Tel Aviv 69978,Israel, amir@math.tau.ac.il

reduce the communication load, compression in a multiwavelet basis was applied. Nevertheless, this part of the procedure can become expensive as the number of subdomains grows considerably.

The algorithm developed in [ABI00] consists of the following steps:

1. In each subdomain a particular solution $u_1^{(s)}$ of the non-homogeneous equation with arbitrary Neumann (Dirichlet) boundary conditions is found.

2. The collection of particular solutions $u_1^{(s)},\ s = 1, \ldots, l$, usually have discontinuities (or discontinuities in the derivatives) on the boundaries of the subdomains. We introduce double (single) layers on the boundaries to match the solutions from different domains to have continuous global solution. The effect of these layers on other boundaries is calculated.

3. With the boundary conditions that were computed in the previous step, the solutions $u_1^{(s)}$ are patched by adding the solutions $u_2^{(s)},\ s = 1, \ldots, l$, of the Laplace equation.

4. An additional solution of the Laplace equation is added to satisfy the boundary conditions on $\partial\Omega$. Namely, for the Dirichlet case the solution u_3 of the homogeneous equation on the boundary $\partial\Omega$ is derived by

$$u_3(x, y) = \Phi(x, y) - u_1(x, y) - u_2(x, y) \tag{4}$$

(the case with Neumann boundary conditions is treated similarly). Thus $u = u_1 + u_2 + u_3$ is the solution of the non-homogeneous equation with the initial non-homogeneous boundary conditions.

The interface jump removal can become cheaper if only adjacent boxes are matched, which is a basis of the hierarchical approach which is proposed in the present paper. The present hierarchical approach matching only two adjacent boxes at each level requires only local corrections at the boundaries of these boxes. The result is a much more efficient computation.

2 Outline of the Algorithm

In the new hierarchical approach the domain is decomposed into k^2 subdomains; first (see Fig. 1) the smallest domains 1,2,3,4 are matched, then they are matched with larger blocks 5,6,7, and, finally, the resulting box is matched with 8,9,10.

The "elementary step" of the hierarchical algorithm is the following.

1. First, in each of four subdomains some smooth boundary conditions are defined. These conditions should not contradict the given right hand side, at the junctions. The Poisson equation is solved with these boundary conditions by a fast spectral algorithm which takes $O(N^2 \log N)$ operations (N is a number of points in each direction).

2. The solutions have a discontinuity in the first derivative. We match the subdomains by adding certain discontinuous functions. In fact we only evaluate these functions at the boundaries of four adjacent subdomains and then solve homogeneous equations in each subdomain with the cumulative boundary conditions.

3. The global homogeneous equation is solved in such a way that it satisfies the assumed conditions at the "global boundaries" of the merged subdomains.

This step is repeated $\log k$ times, for a smalled number of larger subdomains each time.

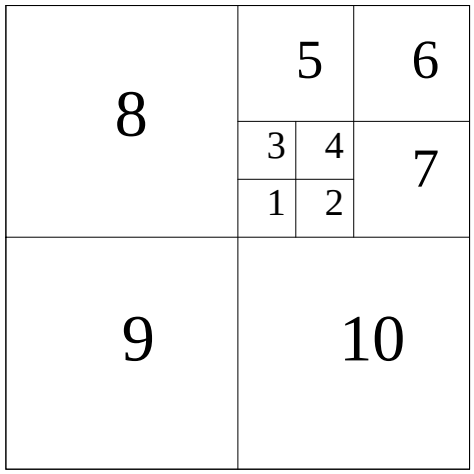

Figure 1: The domain is decomposed into k^2 subdomains; first the smallest domains 1,2,3,4 are matched, then they are matched with larger blocks 5,6,7, and, finally, the obtained box is matched with 8,9,10.

The algorithm can be also implemented on parallel multiprocessors. The parallelization of the serial algorithm is achieved by decomposition of the computational domain into smaller domains. Each domain is assigned to a processor. The information transmitted between the processors is the influences of a function/derivative jumps at the interfaces. of a function/derivative jumps at the interfaces. The low communication is achieved due to the fast decay of these influences and their efficient representation in multiwavelet bases.

For instance, when computing the influence of the derivative jump in the form of the sum of random Gaussians

$$\sum_{k=1}^{12} \exp\{-\alpha_k((x-x_k)^2+(y-y_k)^2)\}, \quad 0.2 \le \alpha_k \le 7,$$

at distance 3 we have only 4 (of 256) multiwavelet coefficients above 10^{-6}, 7 above 10^{-8}, 15 above 10^{-10}.

3 Matching step of the algorithm

The fast and efficient solution of the Poisson/modified Helmholtz and their homogeneous analogs was in detail described in [AIV97, AIV98]. Thus we focus here on the matching step of the algorithm.

Let us consider the simplest ("linear") geometry of the 2-D problem (see Fig. 2) and present the corresponding steps concerned either with the choice of the initial boundary conditions or with patching jumps between the subdomains.

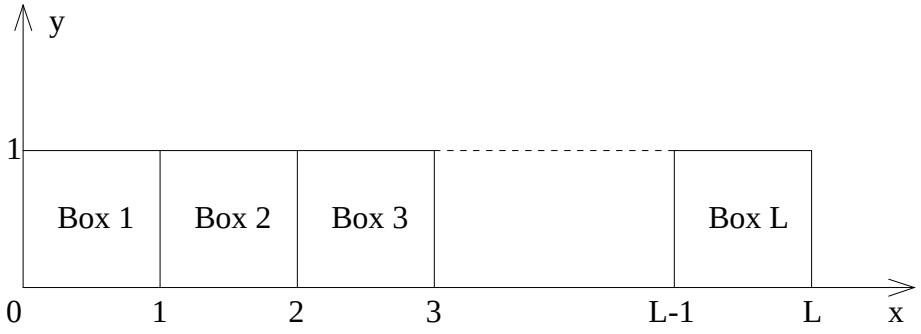

Figure 2: The domain is decomposed into L subdomains.

Step 1. At the boundary of the global domain we assume the original boundary conditions, to avoid singularities at the corners. At the interfaces $x = x_0,\ 0 < y < 1$ we assume

$$u(x_0, y) = u_1(x_0, y) + u_2(x_0, y), \tag{5}$$

where

$$u_1(x_0, y) = \Phi(x_0, 0)\frac{\sinh(\lambda(1-y))}{\sinh(\lambda)} + \Phi(x_0, 1)\frac{\sinh(\lambda y)}{\sinh(\lambda)} \tag{6}$$

matches the values at the interfaces with the value at the boundary and

$$u_2(x_0, y) = \frac{f(x_0, 0) - \Phi(x_0, 0)\lambda^2}{\lambda_1^2 - \lambda_2^2)}\left[\frac{\sinh(\lambda_1(1-y))}{\sinh(\lambda_1)} - \frac{\sinh(\lambda_2(1-y))}{\sinh(\lambda_2)}\right]$$

$$+\frac{f(x_0, 1) - \Phi(x_0, 1)\lambda^2}{\lambda_1^2 - \lambda_2^2)}\left[\frac{\sinh(\lambda_1 y)}{\sinh(\lambda_1)} - \frac{\sinh(\lambda_2 y)}{\sinh(\lambda_2)}\right] \tag{7}$$

to satisfy the Poisson equation at the corners of the interfaces. The latter function vanishes at $x = x_0, y = 0, 1$.

Step 2. We solve the Poisson equation with the prescribed (at the first step) boundary conditions. There is a jump of the first derivative at the interfaces

$$\frac{\partial u}{\partial x}(x_0+, y) - \frac{\partial u}{\partial x}(x_0-, y) = h(y). \tag{8}$$

Since the original boundary conditions are smooth, then $h(0) = h(1) = 0$. After subtracting a function

$$g(y) = \frac{h''(1)}{\lambda_1^2 - \lambda_2^2}\left[\frac{\sin(\lambda_1 y)}{\sin(\lambda_1)} - \frac{\sin(\lambda_2 y)}{\sin(\lambda_2)}\right] \tag{9}$$

$h(y)$ vanishes at $y = 1$ together with its second derivative. A similar function is subtracted for $y = 0$. Then the remaining part $\tilde{h}$ can be accurately expanded into the sine series (in fact, the fourth and higher even derivatives can be also eliminated by an analogous procedure)

$$\tilde{h}(y) = \sum a_k \sin(\pi k y) \tag{10}$$

Then, after adding to the solution to the left of $x = x_0$ the following function

$$\begin{aligned}
&\frac{h''(1)}{2(\lambda_1^2 - \lambda_2^2)} \left[\frac{\cosh(\lambda_1(x - x_0 + L))}{\lambda_1 \sinh(\lambda_1 L)} \frac{\sin(\lambda_1 y)}{\sin(\lambda_1)} - \frac{\sinh(\lambda_2(x - x_0 + L))}{\cosh(\lambda_2 L)} \frac{\sin(\lambda_1 y)}{\sin(\lambda_1)} \right] \\
&\quad + \frac{h''(0)}{2(\lambda_1^2 - \lambda_2^2)} \left[\frac{\cosh(\lambda_1(x - x_0 + L))}{\lambda_1 \sinh(\lambda_1 L)} \frac{\sin(\lambda_1(1 - y))}{\sin(\lambda_1)} \right. \\
&\quad \left. - \frac{\sinh(\lambda_2(x - x_0 + L))}{\cosh(\lambda_2 L)} \frac{\sin(\lambda_1(1 - y))}{\sin(\lambda_1)} \right] \\
&\quad + \frac{1}{2\pi k \cosh(\pi k L)} \sum a_k \sin(\pi k y) \cosh(\pi k (L - x_0 + x))
\end{aligned} \tag{11}$$

and a symmetric (with respect to axis $x = x_0$) to the right of this axis, we obtain a function which is smooth together with its first derivative. Besides, the function which we add decays exponentially with the growth of the distance from $x = x_0$.

4 Numerical Results

Assume that u is the exact solution and u' is the computed solution. In the examples we will use the following measures to estimate the errors:

$$\begin{aligned}
\varepsilon_{MAX} &= \max \|u'_i - u_i\| \\
\varepsilon_{MSQ} &= \sqrt{\frac{\sum_{i=1}^{N} (u'_i - u_i)^2}{n}} \\
\varepsilon_{\mathcal{L}^2} &= \sqrt{\frac{\sum_{i=1}^{N} (u'_i - u_i)^2}{\sum_{i=1}^{N} u_i^2}}
\end{aligned}$$

4.1 Linear geometry

We assume the geometry of Fig. 2 where the domain is decomposed in one dimension only.

Example 1. We solve the Poisson equation $\Delta u = -2 \cos x \cos y$ with the boundary conditions corresponding to the exact solution $u(x, y, z) = \cos x \cos y$ in the domain $[0, 3] \times [0, 1]$ which is divided into three equal boxes.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
32×32	4.1e-7	1.2e-7	2.1e-7
64×64	2.9e-8	8.2e-9	1.4e-8
128×128	2.0e-9	5.4e-10	9.2e-10
256×256	1.3e-10	3.5e-11	5.9e-11
512×512	8.5e-12	2.2e-12	3.8e-12

Table 1: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x, y, z) = \cos x \cos y$ for three boxes

Example 2. We solve the Poisson equation with boundary conditions corresponding to the exact solution

$$u(x,y) = \exp\left\{-\alpha\left((x-x_0)^2 + (y-y_0)^2\right)\right\},$$

with $x_0 = 1.5,\ y_0 = 0.5, \alpha = 2$ in the domain $[0,3] \times [0,1]$ which is divided into three equal boxes.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
32×32	6.8e-6	2.0e-6	4.4e-6
64×64	3.9e-7	1.1e-7	2.4e-7
128×128	2.3e-8	6.6e-9	1.4e-8
256×256	1.4e-9	4.0e-10	8.5e-10
512×512	8.7e-11	2.4e-11	5.2e-11

Table 2: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{-2(x-1.5)^2 + 2(y-0.5)^2\}$.

4.2 Hierarchical subdomains matching

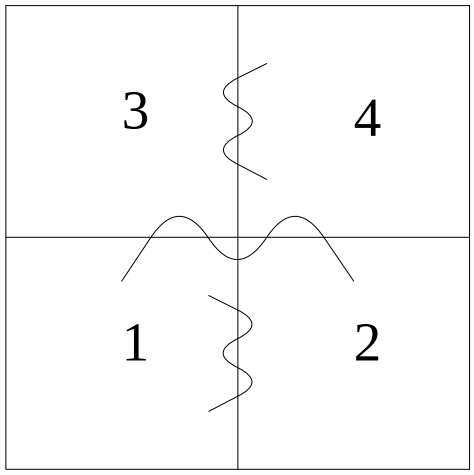

Figure 3: The domain is decomposed into four subdomains

In examples 3, 4 the global subdomain was decomposed into four subdomains (see Fig. 3). In the practical implementation first two pairs of adjacent subdomains were matched: box 1 and 2, box 3 and 4. Afterwards the two resulting boxes were patched. This is also valid for examples 5,6 with 16 subdomains, where each four subdomains were matched in the same way.

Example 3. We solve the Poisson equation with boundary conditions corresponding to the exact solution

$$u(x,y) = \exp\left\{-\alpha\left((x-x_0)^2 + (y-y_0)^2\right)\right\},$$

with $x_0 = 0.5,\ y_0 = 0.5, \alpha = 2$ in the domain $[0,1] \times [0,1]$ divided into four equal boxes.

$N_x \times N_y$ in each subdomain	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	1.7e-4	5.1e-5	7.1e-5
16×16	1.3e-5	3.6e-6	4.9e-6
32×32	1.2e-6	3.5e-7	4.7e-7
64×64	1.0e-7	2.8e-8	3.7e-8
128×128	8.0e-9	2.0e-9	2.7e-9
256×256	6.1e-10	1.4e-10	1.9e-10

Table 3: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{2((x-0.5)^2 + 2(y-0.5)^2)\}$. in the domain $[0,1] \times [0,1]$

Table 4 presents the results for the same exact solution when the Dirichlet problem is solved in the square $[0,2] \times [0,2]$.

$N_x \times N_y$ in each subdomain	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	3.5e-3	1.2e-3	3.1e-3
16×16	1.7e-4	5.7e-5	1.4e-4
32×32	8.3e-6	2.5e-6	6.1e-6
64×64	4.4e-7	1.2e-7	2.9e-7
128×128	2.5e-8	6.5e-9	1.6e-8
256×256	1.5e-9	3.8e-10	9.4e-10

Table 4: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{-2((x-0.5)^2 + 2(y-0.5)^2)\}$ in the domain $[0,2] \times [0,2]$

Example 4. The exact solution is a steep Gaussian

$$u(x,y) = \exp\left\{-10\left((x-0.2)^2 + (y-0.3)^2\right)\right\}.$$

Table 5 presents the numerical errors.

$N_x \times N_y$ in each subdomain	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	1.9e-2	4.4e-3	2.4e-2
16×16	4.9e-4	9.1e-5	4.9e-4
32×32	1.1e-5	2.1e-6	1.1e-5
64×64	3.0e-7	6.0e-8	3.2e-7
128×128	1.1e-8	1.9e-9	1.0e-8
256×256	4.7e-10	6.4e-11	3.5e-10

Table 5: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{-10((x-0.2)^2 + 2(y-0.3)^2)\}$. in the domain $[0,2] \times [0,2]$

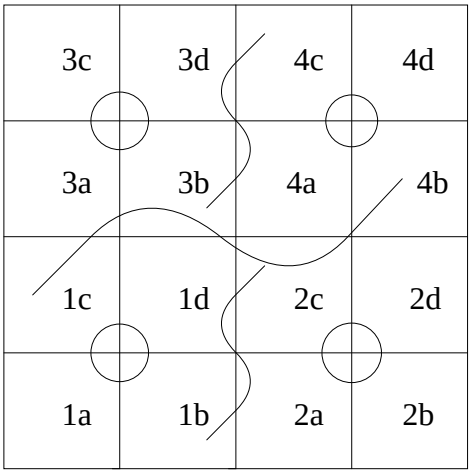

Figure 4: The domain is decomposed into sixteen subdomains

The domain is decomposed into sixteen subdomains which are hierarchically matched (see Fig. 4): first each four small boxes and then the resulting four "big" (joint) boxes. Such domain decomposition was implemented in Example 5.

Example 5. We solve the Poisson equation with the boundary conditions corresponding to the exact solution

$$u(x,y) = \exp\left\{-2\left((x-0.5)^2 + (y-0.5)^2\right)\right\}$$

in the domain $[0,1] \times [0,1]$ divided into sixteen equal boxes (Table 6).

$N_x \times N_y$ in each subdomain	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
4×4	3.6e-4	1.6e-4	2.2e-4
8×8	2.4e-5	1.0e-5	1.4e-5
16×16	2.3e-6	9.6e-7	1.3e-6
32×32	1.9e-7	8.0e-8	1.1e-7
64×64	1.5e-8	6.1e-9	8.2e-9
128×128	1.1e-9	4.5e-10	6.0e-10
256×256	8.3e-11	3.3e-11	4.4e-11

Table 6: MAX, MSQ and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{-2((x-0.5)^2 + 2(y-0.5)^2)\}$ in the domain $[0,1] \times [0,1]$

5 Summary

1. The procedure developed here reduces drastically (by a factor of $O(k^2/\log k)$ times) the number of computations as compared to our previous algorithm where influences of the layers at interfaces are evaluated.

2. The algorithm has an adaptive DD version and also achieves high accuracy (10^{-7} – 10^{-8} for 64×64 points in the smallest subdomains).

3. The algorithm is applicable for parallel implementation as its previous version developed in [ABI00].

4. This algorithm can be used as a preconditioner for the solution of elliptic equations with nonconstant coefficients by solving local constant coefficient problems in subdomains.

5. The present algorithm is close to a multigrid strategy, where the discretization points are replaced by the small boxes in where the equation is satisfied with spectral accuracy which is preserved in the final solution.

References

[ABI00]Amir Averbuch, Elena Braverman, and Moshe Israeli. Parallel adaptive solution of a Poisson equation with multiwavelets. *SIAM J. Sci. Comput.*, 22(3):1053–1086, 2000.

[AIV97]Amir Averbuch, Moshe Israeli, and Lev Vozovoi. On fast direct elliptic solver by modified Fourier method. *Numer. Algorithms*, 15:287–313, 1997.

[AIV98]Amir Averbuch, Moshe Israeli, and Lev Vozovoi. A fast Poisson solver of arbitrary order accuracy in rectangular regions. *SIAM J. Sci. Comput.*, 19:933–952, 1998.

[GL96]Leslie Greengard and June-Yub Lee. A direct adaptive Poisson solver of arbitrary order accuracy. *J. Comput. Phys.*, 125:415–424, 1996.