RATIO MATHEMATICA
ISSUE N. 30 (2016) pp. 35-43

ISSN (print): 1592-7415
(online): 2282-8214

A Recursive Variant of Schwarz Type Domain Decomposition Methods

František Bubeník, Petr Mayer
Faculty of Civil Engineering, Czech Technical University in Prague, Czech Republic
Frantisek.Bubenik@cvut.cz, Petr.Mayer@cvut.cz

Abstract

In this paper a slightly different approach to the use of the domain decomposition method of the Schwarz type is proposed. Instead of the standard coarse space construction we propose to use a recursive solution on each domain. Thus we do not need to construct a coarse space but nevertheless we are still keeping $O(1)$ convergence speed. For local problems we use the standard iterative solvers for which the amount of the work for one step is $O(N)$, where N is the number of equations. Due to the fact that the overlapping is under our control we can keep total work in $O(N^{(1+\gamma)})$ operations with arbitrary positive γ.

Keywords: Domain Decomposition, Finite Element Method, Linear Systems.

2000 AMS subject classifications: 97U99.

1 Introduction

This paper deals with some aspects of the classic Schwarz alternating method. There are analyzed ways how to arrange with the deceleration of algorithms if the stepsize of a mesh for the finite element method is decreasing. The standard two-level method is described and some alternative approaches to solve a number of local problems by the same method are proposed.

2 The Schwartz alternating method

As a model problem we will solve a Poisson problem

$$-\triangle u(x) = f(x), \quad x \in \Omega \subseteq \mathbb{R}^d.$$

We use the finite element method to solve the problem. It leads to a linear system

$$Ax = b\,. \tag{1}$$

Consider functions $\varphi_1, \ldots, \varphi_n$ as a basis and denote by $V_n = span(\varphi_1, \ldots, \varphi_n)$ the linear hull of the functions, that is the set of all linear combinations of the functions. Let us remind that $A_{ij} = a(\varphi_i, \varphi_j)$ and

$$a(u, v) = \int_\Omega uv \; \mathrm{d}\Omega.$$

The matrix A for our problem is symmetric and positive definite. Moreover, for many interesting choices of the basis of V_n, the matrix A is sparse, but large.

One of the possible strategies to solve the system (1) is to use a sparse version of LU-decomposition. In most cases, fill-in which takes place along with Gaussian elimination makes such an approach unusable. A usual choice is then to use some iterative method. Since our matrix is a symmetric positive definite, it seems to be more advantageous to employ the conjugate gradient method. But this choice is still problematic because the amount of the work is rapidly increasing with the size of the problem. Therefore there is then more convenient to use a preconditioned conjugate gradient method with an appropriate preconditioner.

As a preconditioner we can choose a slightly modified the Schwarz alternatig method, see [2]. The original description is in [1]. We can see it as a kind of some block symmetrized Gauss-Seidel method.

2.1 Formulation of the algorithm

Let us denote by n_d the number of domains. For each $i \in I = \{1, 2, \ldots, n_d\}$ we define an index set $I_i = \left\{ i_1^{(i)}, i_2^{(i)}, \ldots, i_{n_i}^{(i)} \right\}$. These index sets realize a covering of I, i. e. $I = \bigcup_{i=1}^{n_d} I_i$. This covering is not required to be disjoint. We define subspaces of V_n so that the subspace $V_n^{(i)}$ is the linear hull of a corresponding part of the basis of V_n, that is $V_n^{(i)} = \underset{j \in I_i}{span} \; \{\varphi_j\}$. Finally we define subdomains

$$\Omega^{(i)} = \bigcup_{j \in I_i} supp\,(\varphi_j)\,, \tag{2}$$

where $supp\,(f)$ denotes the support of a function f. The sizes of individual domains are $n_1, \ldots, n_{n_d}$. We can write $A^{(i)} = A(I_i, I_i)$ in terms of Matlab-like notation. Matrix interpretation of local problems is $N^{(i)} = \{n_{k,l}^{(i)}\} \in R^{n \times n_i}$, where

$$n_{k,l}^{(i)} = \begin{cases} 1 & \text{if} \quad l = i_k^{(i)}, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$A^{(i)} = N^{(i)^T} A\, N^{(i)}. \tag{3}$$

The following algorithm describes the transition from $x^{(k)}$ to $x^{(k+1)}$.

Algorithm 2.1. *One step of the symmetrized Schwarz method*

```
$x^{(k+\frac{0}{2n_d})} := x^{(k)}$
$for\ i = 1, \ldots, n_d$
        $r := b - Ax^{(k+\frac{i-1}{2n_d})}$
        $\widetilde{r} := N^{(i)^T} r$
        $A^{(i)} := N^{(i)^T} A N^{(i)}$
($\clubsuit$)   $Solve\ A^{(i)} c = \widetilde{r}, \quad i.\ e. \quad c = A^{(i)^{-1}} \widetilde{r}$
        $x^{(k+\frac{i}{2n_d})} := x^{(k+\frac{i-1}{2n_d})} + N^{(i)} c$
$end\ for$                                                        (4)
$for\ i = 1, \ldots, n_d$                                          (5)
        $r := b - Ax^{(k+\frac{1}{2}+\frac{i-1}{2n_d})}$
        $\widetilde{r} := N^{(n_d+1-i)^T} r$
        $A^{(n_d+1-i)} := N^{(n_d+1-i)^T} A N^{(n_d+1-i)}$
($\clubsuit$)   $Solve\ A^{(n_d+1-i)} c = \widetilde{r}$
        $x^{(k+\frac{1}{2}+\frac{i}{2n_d})} := x^{(k+\frac{1}{2}+\frac{i-1}{2n_d})} + N^{(n_d+1-i)} c$
 $end\ for$
$end\ algorithm$
```

The method used in the Algorithm 2.1 can also be viewed as a variant of the block Gauss-Seidel method, but with the fact that the individual blocks can overlap.

Put

$$P^{(i)} = A^{1/2} N^{(i)} (N^{(i)^T} A N^{(i)})^{-1} N^{(i)^T} A^{1/2}. \tag{6}$$

Then

$$\begin{aligned} P^{(i)^2} &= A^{1/2}N^{(i)}(N^{(i)^T}AN^{(i)})^{-1}N^{(i)^T}A^{1/2}A^{1/2}N^{(i)}(N^{(i)^T}AN^{(i)})^{-1}N^{(i)^T}A^{1/2} \\ &= A^{1/2}N^{(i)}A^{(i)^{-1}}A^{(i)}A^{(i)^{-1}}N^{(i)^T}A^{1/2} \\ &= A^{1/2}N^{(i)}A^{(i)^{-1}}N^{(i)^T}A^{1/2} = P^{(i)}. \end{aligned}$$

It follows that $P^{(i)}$ is a projection, moreover A-orthogonal. Further $P^{(i)} = P^{(i)^T}$, then the projection is symmetric.

Let us denote

$$\varepsilon^{(k)} = x^{(k)} - x^*,$$

where x^* denotes the solution of the problem. Errors are analyzed in terms of the energy norm $||x||_A = \sqrt{(x,x)_A}$, where $(x,y)_A = x^T A\, y$.

Let us note that A is a symmetric positive definite matrix and $A^{(i)}$ is a principal minor of A. Then $A^{(i)}$ is also a symmetric positive definite matrix.

We have

$$||\varepsilon^{(k)}||_A^2 = \varepsilon^{(k)^T} A\, \varepsilon^{(k)} = \varepsilon^{(k)^T} A^{1/2}\, A^{1/2}\, \varepsilon^{(k)} = ||A^{1/2}\varepsilon^{(k)}||_A^2.$$

Thus

$$\begin{aligned} A^{1/2}\varepsilon^{(k-1)} &= (I - P^{(1)})\dots(I - P^{(n_d)})(I - P^{(n_d)})\dots(I - P^{(1)})A^{1/2}\varepsilon^{(k)} \\ &= M\, A^{1/2}\varepsilon^{(k)}. \end{aligned} \tag{7}$$

Since $I - P^{(i)}$ is a symmetric $A-$orthogonal projection then M is a symmetric matrix. And moreover, M is, according to the definition, a positive semi-definite matrix. It can be proved that M is even a positive definite matrix.

2.2 Dependence on the dimension of V_n

For the following considerations we suppose that piecewise linear finite elements are used. In that case $n = O(1/h^d)$, where h is the stepsize of a mesh. If we try to keep domains with the same geometry, the amount of elements will increase as $O((H/h)^d)$, where H is the typical size of a domain. Then, the amount of iterations is the same, but the amount of work for one step will increase.

On the other hand, when we keep equal the number of elements inside a domain, then the size of the domain will decrease and then the number of domains will increase. This leads to increasing amount of iterations and slightly increasing work for one full step.

Usual solution is to use a coarse space. It typically means to replace each domain by a base function. We create the coarse space and the solution of the problem for the corrections on the coarse level is inserted between steps (4) and (5) of the Algorithm 2.1. A detailed analysis is introduced, for example, in [2]. When it is used in a right way, we get $O(1)$ convergence speed.

2.3 Basic convergence

Let us denote

$$E(x) = x^T Ax - 2x^T b. \tag{8}$$

It is known that $Ax = b$ if and only if $E(x)$ assumes its minimum at x. Each step in Algorithm 2.1 means the minimization of functional (8) on a corresponding subspace and the following inequality holds for the successive terms of the minimizing sequence

$$E(x^{(k+1)}) \leq E(x^{(k)}). \tag{9}$$

We prove the following equivalence:

Lemma 2.1. *The equality in* (9) *occurs* $\Longleftrightarrow x^{(k)}$ *is the accurate solution of* $Ax^{(k)} = b$.

Proof. It is clear that an accurate solution is equivalent to $r^{(k)} = 0$, where $r^{(k)} = b - Ax^{(k)}$.

We prove one direction of the equivalence: Suppose that $x^{(k)}$ is an accurate solution and we prove the equality required. It is easy to see that if $x^{(k)}$ is an accurate solution then $r^{(k)} = 0$ and then the equality in (9) occurs.

Now we prove the opposite direction of the equivalence. We apply the proof by contradiction: suppose that $x^{(k)}$ is not an accurate solution and suppose that the equality in (9) holds. If $x^{(k)}$ is not an accurate solution then $r^{(k)} \neq 0$. Then there exists the least index i_m such that ${N^{(i_m)}}^T r^{(k)} \neq 0$. It causes a decrease at this step of Algorithm 2.1 and then the strict inequality $E(x^{(k+1)}) < E(x^{(k)})$. This contradicts to the assumption of equality in (9) and the proof of the equivalence in Lemma 2.1 is complete. $\square$

3 Recursive approach

Another possibility comes from the idea that the local problems are conceptually identical as the original one. It opens a possibility to use the same Schwarz algorithm for solving them. It means to retain domains in the same geometry, then $\Omega^{(i)}$ in (2) remain unchanged. On the other hand it means that the number of degrees of freedom for a domain increases. In this case we recommend to use the same algorithm for each domain separately.

3.1 Two - level variant

We start from the Algorithm 2.1, and we replace both steps denoted by (♣) in the algorithm by an iterative solution for c. It means that we replace ${A^{(i)}}^{-1}\widetilde{r}$ by an approximate solution of the problem $A^{(i)}c = \widetilde{r}$. As the method we use again the algorithm 2.1 with ℓ steps.

Then the error operator has the form

$$\widetilde{M} = (I - \widetilde{P}^{(1)}) \dots (I - \widetilde{P}^{(n_d)})(I - \widetilde{P}^{(n_d)}) \dots (I - \widetilde{P}^{(1)}),$$

where

$$\widetilde{P}^{(i)} = A^{1/2} N^{(i)} \widetilde{Q}^{(i)} {N^{(i)}}^T A^{1/2}. \tag{10}$$

Expression $({N^{(i)}}^T A N^{(i)})^{-1}$ in (6) is for short denoted by $Q^{(i)}$ and it is replaced in (10) by

$$\widetilde{Q}^{(i)} = {A^{(i)}}^{-1/2} \left[I - \left(I - {A^{(i)}}^{1/2} M^{(i)} {A^{(i)}}^{1/2} \right)^{\ell} \right] {A^{(i)}}^{-1/2}, \tag{11}$$

where $A^{(i)}$ is from (3) and $M^{(i)}$ denotes the error operator to the algorithm 2.1 applied on the $i-$th domain. This replacement comes from the following:

Let us solve a problem $Ax = b$ and let us use the following iterative method

$$x^{(i+1)} = x^{(i)} + M(b - Ax^{(i)}),$$

where M is a symmetric positive definite matrix. The initial approximation is

$$x^{(0)} = 0. \tag{12}$$

Let $x^* = A^{-1}b$. Then

$$x^* - x^{(i+1)} = x^* - x^{(i)} - M(b - Ax^{(i)}) = (I - MA)(x^* - x^{(i)})$$

and then

$$A^{1/2}(x^* - x^{(i+1)}) = (I - A^{1/2} M A^{1/2}) A^{1/2} (x^* - x^{(i)}).$$

After $\ell-$iterations we get

$$A^{1/2}(x^* - x^{(\ell)}) = (I - A^{1/2} M A^{1/2})^{\ell} A^{1/2} (x^* - x^{(0)})$$

and thus

$$x^* - x^{(\ell)} = A^{-1/2} (I - A^{1/2} M A^{1/2})^{\ell} A^{1/2} (x^* - x^{(0)}).$$

Since $x^{(0)} = 0$ we get

$$x^* - x^{(\ell)} = A^{-1/2}(I - A^{1/2}MA^{1/2})^\ell A^{1/2}x^* = A^{-1/2}(I - A^{1/2}MA^{1/2})^\ell A^{-1/2}b.$$

Thus

$$\begin{aligned} x^{(\ell)} &= x^* - A^{-1/2}(I - A^{1/2}MA^{1/2})^\ell A^{-1/2}b \\ &= A^{-1/2}\left[I - \left(I - A^{1/2}MA^{1/2}\right)^\ell\right] A^{-1/2}b \end{aligned}$$

and that is why the form of $\widetilde{Q}^{(i)}$ in (11) and $\widetilde{P}^{(i)}$ in (10).

3.2 Recursive - multilevel method

When we use a two-level method we need to compute $\widetilde{P}^{(i)}$ in (10) and for it there is required to know $\widetilde{Q}^{(i)}$ from (11). For $\widetilde{Q}^{(i)}$ there is necessary to know $M^{(i)}$ and its realization comes from the solution of a local problem on subdomains of the $i-$th domain. In case when these local problems are still large then the process may be repeated again and a two-level method becomes a recursive-multilevel method.

As to convergence of a recursive variant of the method the same facts as in section 2.3 can be used. Then we can state that a recursive - multilevel method converges as well and we have proved the following theorem.

Theorem 3.1. *A recursive - multilevel method is convergent.*

4 Cost analysis

4.1 Two levels

The work required for solving a problem of size n is

$$W = K\, n^{(1+\beta)}$$

with K and β positive constants. Let α be a relative overlapping in one dimension. The number of domains is n_d. Then the size of a local problem is

$$n_{loc} = \frac{n}{n_d}(1+\alpha)^d$$

and the work needed for its solving

$$W = K\left(\frac{n}{n_d}(1+\alpha)^d\right)^{(1+\beta)}.$$

Thus the work for one iteration is

$$W = 2n_d\, K\, \left(\frac{n}{n_d}(1+\alpha)^d\right)^{(1+\beta)} = \frac{2(1+\alpha)^{d(1+\beta)}}{n_d^\beta} K\, n^{(1+\beta)}$$

and for ℓ iterations on this level

$$W = 2\ell \frac{(1+\alpha)^{d(1+\beta)}}{n_d^\beta} K\, n^{(1+\beta)}.$$

4.2 k levels

In the $k-$th level we repeat the previous considerations. We have n_d^k subdomains and the size of one subdomain is $n_{loc,k} = n\left(\frac{1+\alpha}{n_d}\right)^k$. The problem is solved on each subdomain $(2\ell)^k$ times. Total work is

$$W = (2\ell)^k n_d^k\, K\, \left(n\left(\frac{1+\alpha}{n_d}\right)^k\right)^{(1+\beta)} = K\left(2\ell\, n_d^{-\beta}(1+\alpha)^{(1+\beta)}\right)^k n^{(1+\beta)}.$$

4.3 Full recursion

We want to find such k that $n_{loc,k} = 1$. We take the greatest possible k. This is

$$k = \frac{\ln n}{\ln n_d - \ln(1+\alpha)}.$$

We obtain

$$W = K\left(2\ell\, n_d^{-\beta}(1+\alpha)^{(1+\beta)}\right)^{\frac{\ln n}{\ln n_d - \ln(1+\alpha)}} n^{(1+\beta)}$$

which is

$$W = K\exp^{(\ln 2 + \ln\ell - \beta\ln n_d + (1+\beta)\ln(1+\alpha))\frac{\ln n}{\ln n_d - \ln(1+\alpha)} + (1+\alpha)\ln n}.$$

This expression can be improved and after some manipulations we get

$$W = K\, n^{(1+\gamma)},$$

with

$$\gamma = \frac{\ln(1+\alpha) + \ln 2 + \ln\ell}{\ln n_d - \ln(1+\alpha)}. \tag{13}$$

We can see from (13) that it is possible to achieve γ arbitrary small by an appropriate choice of $\alpha,\ \ell, n_d$.

5 Conclusion

An alternative process to the classic two-level method with a coarse space is proposed in this paper. One of the significant advantages of the method presented here is the fact that we can extremely reduce the memory requirements if this is called for.

References

[1] H. A. Schwarz, *Gessamelte Mathematische Abhandlungen*, volume 2, pages 133 – 143, Springer, Berlin, (1890). First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, *Über einen Grenzübergang durch alternierendes Verfahren*, volume 15, (1870), pp. 272 – 286.

[2] A. Toselli and O. Widlund, *Domain Decomposition Methods - Algorithms and Theory.* Springer-Verlag, Berlin Heidelberg, (2005).

[3] M. Brezina and P. Vaněk, *A black-box iterative solver based on a twolevel Schwarz method.* Computing 63(3), (1999), 233 – 263.

[4] R. Varga, *Matrix Iterative Analysis.* Prentice Hall, first edition, (1962).