53

Remarks on the implementation of the Generalized Neumann-Neumann algorithm

Marina Vidrascu [1]

INTRODUCTION

In the past decade significant research efforts have been applied to develop robust algorithms using domain decomposition methods. From both a theoretical and a numerical point of view these methods can now be considered mature. For widespread use, these methods must be accessible to the non-specialist and thus must be available as black boxes. This paper addresses the issues faced when trying to maintain an open software platform suitable for ongoing research while simultaneously meeting commercial code requirements as black boxes. The characteristics of the generalized Neumann-Neumann algorithm, a good black box domain decomposition algorithm candidate, are defined and illustrated with specific examples.

First the general methodology used to practically solve a given problem is described. The next sections review the main characteristics of the Neumann-Neumann preconditioner and discusses the key points of its practical implementation and use. This algorithm is implemented whitin the framework of the general purpose finite element library Modulef. For message passing the PVM library is used. Thus the software is portable and can be run on different platforms including standard parallel computers and clusters of workstations.

1 INRIA-Rocquencourt, B.P 105, F78153 Le Chesnay Cedex, e-mail : Marina.Vidrascu@inria.fr

Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund ©1999 DDM.org

GENERAL METHODOLOGY

To achieve efficiency and reliability modern software must be open and modular. An additional important feature is that it should be easlly reused in different situations. On the other hand, in order to promote the use of numerical methods for non-specialists, it is important to develop black boxes. A black box is a specialized software which is easy to use and minimizes the user's decisions. In addition, it has to be robust and efficient. The question is how to fulfill the apparent conflicting requirements of both modern software and black boxes.

For numerical analysts who use an open software, the solution of a problem is built using a selection of appropriate modules. In general several solution methods are available and it is the user's responsibility to choose the best ones depending, for instance, on the problem type or size. Modules for direct or iterative solutions of linear systems are typical examples of software used in all such constructions. On the contrary, a black box contains all steps involved in the modeling process and corresponds to a particular choice at each step.

The approach proposed to derive a black box from an open software for a given problem is the following:

- Select the optimal solution method, i.e. one whith serious theoretical background and proved to be robust and reliable. In the context of black boxes the best choice is not always the most efficient one. It is interesting to note that, in commercial codes, a direct solution method is often preferred to an iterative one even if in the research community it is well established that iterative methods are more efficient than direct ones especially for large three dimensional problems.
- Design specific pre- and post-processing tools well adapted to the problem to be solved. In particular, check the validity of the model as far as possible and use the specific vocabulary of the actual application.

The advantage of this approach is that the solution of a new problem requires only a few software developments which increase both productivity and robustness.

An overview of existing methods available to solve linear systems shows why it is interesting to consider domain decomposition algorithms. On the one hand direct methods are robust but too expensive for large three dimensional problems. On the other hand, iterative methods are efficient but it is difficult to have robust preconditioners in particular when considering highly non-homogeneous materials. Domain decomposition methods use direct solvers on each subdomain and iterative one on the interface. The choice of the preconditioner is a key point to achieve robustness and performance. The objective of the following sections is to show that the balanced Neumann Neumann algorithm is a good candidate to build a black box for linear and non linear structural mechanics problems.

GENERALIZED NEUMANN-NEUMANN PRECONDITIONER

Definition

The generalized Neumann-Neumann preconditioner described in [LV98], [LV97] is a particular case of the additive Schwarz method applied to the primal Schur complement.

To define the interface problem, first split the original domain of calculation Ω into non-overlapping sub-domains $\Omega = \bigcup_{i=1}^{N} \Omega_i$ with interfaces $\Gamma_i = \partial\Omega_i \backslash \partial\Omega \quad \Gamma = \cup_i \Gamma_i$.

Then consider a decomposition of each local stiffness matrix in internal and interface nodes contributions :

$$\mathbf{K}^i = \begin{bmatrix} \mathring{\mathbf{K}}^{\mathbf{i}} & \mathbf{B}^i \\ \mathbf{B}^{i^t} & \bar{\mathbf{K}}^i \end{bmatrix},$$

Denoting by $\mathbf{R}^i \bar{X}$ the restriction of the set of interface nodal values $\bar{X}$ to the interface Γ_i, the entire interface problem takes the form of the classical Schur complement system :

$$\mathbf{S}\bar{X} := \sum_i \mathbf{R}^{i^t} \left(\bar{\mathbf{K}}^i - \mathbf{B}^{i^t} (\mathring{\mathbf{K}}^i)^{-1} \mathbf{B}^i \right) \mathbf{R}^i \bar{X} = \bar{F} - \sum_i \mathbf{R}^{i^t} \mathbf{B}^{i^t} (\mathring{\mathbf{K}}^i)^{-1} \bar{F}^i. \qquad (1)$$

To define the generalized Neumann Neumann preconditioner choose :

1. a partition of unity $\mathbf{D}^i : \mathbf{V}_{|\mathbf{\Gamma_i}} \to \mathbf{V}$ satisfying $(\mathbf{V} = \{\bar{\mathbf{v}} = \mathbf{Tr}\ \mathbf{v}|_{\mathbf{\Gamma}}\})$

$$\sum_{i=1}^{N} \mathbf{D}^i \mathbf{R}^i = \mathbf{Id}|_{\mathbf{V}}. \qquad (2)$$

2. an approximate local operator $\tilde{\mathbf{S}}^i$ such that

$$\tilde{\mathbf{S}} = \sum \mathbf{R}^{i^t} \tilde{\mathbf{S}}^i \mathbf{R}^i \approx \mathbf{S}. \qquad (3)$$

   In linear elasticity this is generally the exact local Schur complement, in non linear elasticity it may be the local Schur complement of the linearized elasticity operator.

3. an $\tilde{\mathbf{S}}^i$-orthogonal decomposition of each local space $\mathbf{V}_{|\mathbf{\Gamma_i}}, \mathbf{i} = \mathbf{1}, \mathbf{N}$, into

$$\mathbf{V}_{|\mathbf{\Gamma_i}} = \mathbf{V_i} \oplus \mathbf{Z_i}. \qquad (4)$$

   The local coarse space $\mathbf{Z}_i$ contains all potential local singularities.

The generalized Neumann-Neumann domain decomposition technique is then the additive Schwarz algorithm solving $\mathbf{S}$ on the space $\mathbf{V}$ of restrictions with

1. coarse space $\mathbf{V_0} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{D^i Z_i} \subset \mathbf{V}$, endowed with the scalar product $\tilde{\mathbf{S}}$,
2. local spaces $\mathbf{V_i}, \mathbf{i} = \mathbf{1}, \mathbf{N}$ endowed with the scalar product $\mathbf{B_i} = \tilde{\mathbf{S}}^i$,
3. extensions $I_i = (I - \mathbf{P})\mathbf{D}^i$, with $\mathbf{P}$ the $\tilde{\mathbf{S}}$ orthogonal projection of $\mathbf{V} \to \mathbf{V_0}$.

This algorithm corresponds to the following preconditioning operator

$$\mathbf{M}^{-1} = \tilde{\mathbf{S}}_0^{-1} + \sum_i (I - \mathbf{P}) \mathbf{D}^i \tilde{\mathbf{S}}_i^{-1} \mathbf{D}^{i^t} (I - \mathbf{P})^t. \qquad (5)$$

General properties

The purpose of this section is to explain why the Neumann-Neumann preconditioner is a good candidate for a black box system. The relevant features in this context include the fact that one can use arbitrary unstructured subdomains and meshes and various different finite elements (2d, 3d, shells).

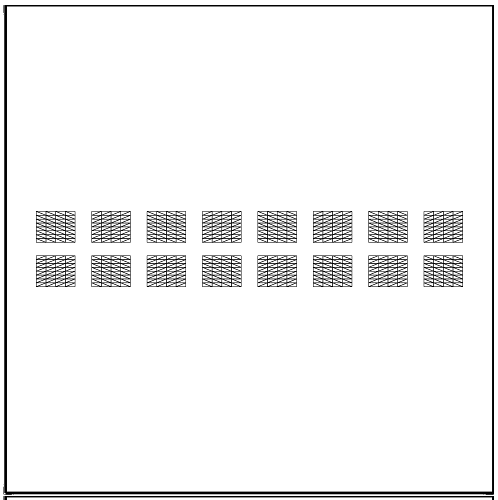
This preconditioner has been extensively studied both from a theoretical and numerical point of view ([Man93], [Le 94], [LV97]). It is robust and can handle non-homogeneous materials and jumps in coefficients. The algorithmic scalability of the preconditioner was proved, i.e. the number of iterations to reach convergence is asymptotically independent on the mesh size and on the number of subdomains. In early versions of the preconditioner (Original Neumann-Neumann in Table 1) the independence on the number of subdomains was not achieved. This improvement, due to J. Mandel [Man93], obtained by the introduction of the coarse space is mandatory for the robustness (Generalized Neumann Neumann in Table 1). There still is a drawback, the aspect ratio of the subdomains has an important influence in the convergence. In practice this means that the way the domain is split into subdomains is very important.

Both the importance of algorithmic scalability and aspect ratio of the subdomains are illustrated with a very simple example. We consider a two dimensional elasticity problem. The domain is clamped on the left side and a uniform traction is imposed on the right side. The domain is decomposed into 2, 4, 8, 16 subdomains. In this last case several decompositions are considered, see Figures 1, 2, 3. The convergence behavior is presented in Figure 4 and clearly shows that the original Neumann-Neumann is not a good candidate for a black-box as the number of iterations increases with the number of subdomains. In addition, the generalized Neumann-Neumann algorithm is stable with respect to the number of subdomains. The number of iterations is given in Table 1. In Figure 4 the best result for the decomposition in 16 subdomains is used for the graphic representation. This example is not used to prove the performance of the method (for this purpose a small 2d problem is irrelevant), but to illustrate that the algorithmic scalability is an issue. The comparison between the three decompositions used for the case of 16 subdomains shows the importance of the aspect ratio of the subdomains and of the regularity of the boundary. This is even worth in 3d real life problems where the algorithm may not converge for an inappropriate decomposition.

In addition this preconditioner is easy to implement and several extensions, to non-matching grids, to non-linear elasticity or to low frequency vibration problems are possible in a straightforward manner.

PARALLEL IMPLEMENTATION FOR THE SOLUTION OF LARGE SIGNIFICANT PROBLEMS

One motivation for the design of domain decomposition algorithms was that they are suitable for coarse-grained parallel computers with distributed memory or for clusters of workstations. The actual implementation is done within the finite element library Modulef with no restrictions on the shapes of the domains and on the choice of finite elements. To preserve the portability, the PVM library is used to exchange messages



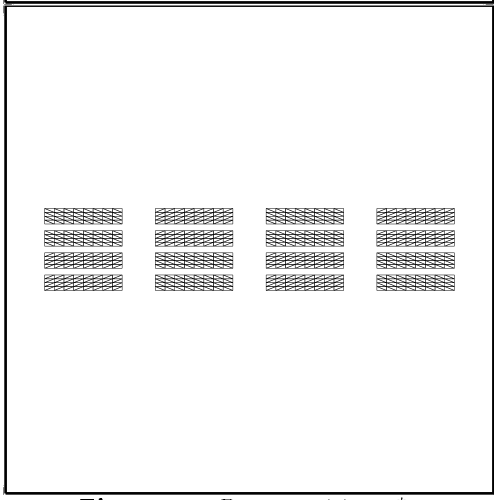
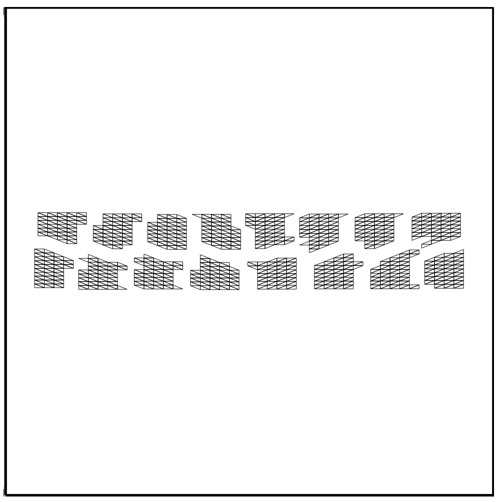


Figure 1 *Decomposition 8*2*

Figure 2 *Decomposition 4*4*



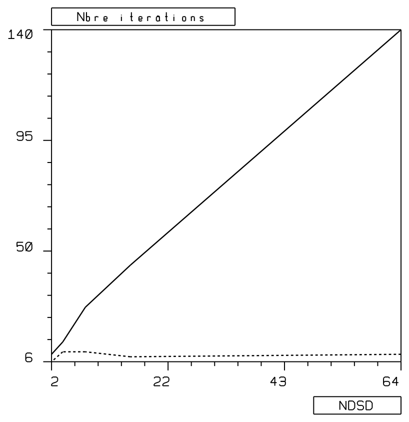


Figure 3 *Irregular decomposition*

Figure 4 *Number of iteration vs number of subdomains*

Table 1 *Number of iterations for the two versions of the preconditioner*

# subdomains	Orig. Neumann Neumann	Generalized Neumann Neumann
2	9	6
4	14	10
8	28	10
16 (4*4) (Fig: 2)	65	21
16 (8*2) (Fig: 1)	45	8
16 (irregular) (Fig: 3)	70	17
64	140	9

and data between processors.

There are two different ways to tackle the practical implementation of this domain decomposition algorithm : either consider the entire domain or a given decomposition into non-overlapping subdomains as an input.

The choice used here is the second one because:

- It is more amenable to a modular approach. Indeed, from an algorithmic point of view the only link between the decomposition of the domain into subdomains and the solution is that regular domains with nice aspect ratio are needed. In addition several tools can be used for mesh partitioning (see section 53) and the re-usability of algorithms is increased.
- It allows the solutions of very large systems where the memory of one processor is not sufficient for the entire mesh.
- Each subdomain is considered as a standard mesh data structure. Thus, the data locality, an important problem in parallel processing, is obtained with no additional work and this also ensures a very low degree of data migration.

There is, of course a drawback: a particular data structure to describe the interface between subdomains is needed. We will concentrate here on the solution algorithm, the mesh partitioning being a completely different problem.

To summarize section 53, the problem is to solve $\mathbf{S}\bar{X} = F$ where $\mathbf{S}$ is defined by (1), using a preconditioned conjugate gradient algorithm where the preconditioner M is defined by (5).

The software is parallel and uses the message passing paradigm and a **master-slave** approach. Here, the **master** pilots the PCG algorithm for the interface problem and each **slave** is in charge of the local computations by subdomain. As for classical problems the boundary conditions, material characteristics and loads are described globally, in a standard way.

The user starts the master process which will start as many slave processes as subdomains. A modular approach is used which make it easy to program the extensions described in section 53.

The **master** process has the following tasks:

i) spread data to all processors, collect data to compute the data structure describing the interface,
ii) pilot construction of the coarse problem,
iii) pilot the conjugate gradient iterations. The only non-standard step is the solution of the coarse problem in the preconditioning step. As this is a problem of small size (less then $6 \times$ *subdomains*) a direct method is convenient.

The main programming effort is contained in step ii). Step iii) is standard and uses numerical kernels such as BLAS. The specific steps are the matrix vector product and the preconditioning step. Notice that $\mathbf{S}$ is never computed but, building the matrix-vector product $\mathbf{S}\bar{X}$ involves local solutions of Dirichlet problems on each subdomain. This will be done by the **slave** processes. Similarly, at the preconditioning step, computation of $M^{-1}\bar{r}$, involves solution of local Neumann problems. In both situations the **master** process send to the **slave** processes a vector containing data on the boundary, the **slave** processes perform the local computations and send back a vector containing local solutions and, finally, the **master** process add up all contributions. This description shows that the communications are very simple, so the choice of the message passing software is not an issue.

The **slave** processes have the following tasks:

i) receive data from master and use it to compute matrix $\mathring{\mathbf{K}}^i$ in (1) and $\tilde{\mathbf{S}}$ in (5). These matrices will be used to solve local Dirichlet problems (for the computation of $\mathbf{Sx}$), respectively Neumann problems (for the preconditioner). They are computed and factorised once for all as direct methods are used to solve local problems. Also compute matrix $\mathbf{B}^i$ in (1).
ii) construction of the coarse problem. The key point is the choice of the local coarse space $\mathbf{Z}_i$ in (4). For elasticity problems, $\mathbf{Z}_i$ is the space of local rigid-body motions. To compute it, introduce the independent degrees of freedom characterizing the rigid body motions (six or less of them). The simplest way to identify these modes is to stop the factorization process when a "null pivot" is found. This procedure is not very robust as the value of the "null pivot" depends on the conditionning of the matrix which is unknown. A reliable procedure, such as that described in [FG96], is requiered for a black box.
iii) conjugate gradient iteration. The local contribution of vector $\bar{X}$ is $X_i = \mathbf{R}^i\bar{X}$ and contribution to solution of $\mathbf{S}\bar{X}$ implies the solution of the Dirichlet problem $(\mathring{\mathbf{K}}^i)^{-1}(\mathbf{B}^i X_i)$. Similar computations for the preconditioning step, once the $\mathbf{D}^i$ in (5) are choosed as indicated in (2). A good choice which allows to consider jumps in coefficients is to define $\mathbf{D}^i$ at each interface point P_l as the relative value of the average stiffness ρ_i of subdomain Ω_i to the sum $\sum_{P_l \in \Omega_j} \rho_j$ of stiffness values of all the subdomains Ω_j containing P_l.

Notice that the computations performed by the **slaves** are done in parallel.

REQUIREMENTS FOR CORRECT USE OF DOMAIN DECOMPOSITION

To use a domain decomposition algorithm as a black box it is essential to use an automatic mesh partitioning tool. The Modulef library has a partitioner based on K-means techniques. Other popular partitioning tools well-adapted to this algorithm are described in ([SF95], [Sim91]). As noted in section 53 the quality of the partition is very important from an algorithmic point of view, which is true for an entire class of domain decomposition methods [FMB94]. For this method a good aspect ratio of the domains is required, as already mentioned in the example of section 53 (compare results obtained with the decompositions in figures 1 and 2). It is almost impossible, for general meshes to obtain a regular boundary with an automatic mesh partitioner. This has an impact on convergence behavior (again, see in section 53 results obtained with meshes of figures 1 and 3). Nevertheless, in realistic computations it is mandatory to use an automatic tool partitioning.

To obtain parallel performance the decomposition must also achieve a good load balancing. How to achieve a good decomposition is still an active research area. This is an essential requirement for a correct use of domain decomposition methods.

CONCLUSIONS

Domain decomposition methods when used on parallel computers provide a very powerful tool to solve large real-life elliptic problems. The generalized Neumann-Neumann algorithm is robust and can easily be implemented using existing software. In order to use it as a black box it is necessary to combine the use ot this algorithm with an automatic mesh partitioning tool, further research should address the issue of the impact of the size of subdomains to achieve parallel performance as well as the choice of the best strategy to partition the mesh.

REFERENCES

[FG96] Farhat C. and Gradin M. (may 1996) On the computation of the null space and generalized inverse of a large matrix, and the zero energy modes of a structure. Technical Report CU-CAS-96-13, Center for aerospace structures, University of Colorado, Boulder.

[FMB94] Farhat C., Maman N., and Brown G. (january 1994) Mesh partitioning for implicit computations via domain decomposition: Impact and optimization of the subdomain aspect ratio. Technical Report CU-CAS-94-02, Center for aerospace structures.

[Le 94] Le Tallec P. (1994) Domain decomposition methods in computational mechanics. In *Computational Mechanics Advances*, number 1 in Computational Mechanics Advances, pages 121–220. North-Holland.

[LV97] Le Tallec P. and Vidrascu M. (1997) *Solving Large Scale Structural Problems on Parallel Computers using Domain Decomposition*, chapter 2, pages 49–82. John Wiley & Sons.

[LV98] Le Tallec P. and Vidrascu M. (1998) Generalized neumann-neumann preconditioners for iterative substructuring. In Petter Bjørstad M. E. and

Keyes D. (eds) *Proceedings of the ninth international symposium on Domain Decomposition Methods for Partial Differential Equations, Bergen, June 96*, pages 413–425.

[Man93] Mandel J. (1993) Balancing domain decomposition. *Communications in numerical methods in engineering* 9: 233–241.

[SF95] Sharp M. and Farhat C. (1995) TOP/DOMDEC A Totally Object Oriented Program for Visualisation, Domain Decomposition and Parallel Processing. User's manual, PGSoft and The University of Colorado, Boulder.

[Sim91] Simon H. (1991) Partitioning of unstructured problems for parallel processing. *Comput. Sys. Engrg.* 2: 135–148.