National Conference with International Participation

ENGINEERING MECHANICS 2008

Svratka, Czech Republic, May 12 – 15, 2008

PRECONDITIONING OF THE CONJUGATE GRADIENT METHOD BASED ON AGGREGATION

J. Kruis[*] , P. Mayer [**]

Summary: *Efficient solvers for large systems of algebraic equations are still in the center of attention because time devoted to the solution of such systems is the main part of computational time. This contribution deals with preconditioning of the conjugate gradient method by the aggregation method. The preconditioning described is useful especially for very ill-conditioned problems which lead to thousands of iterations. Unknowns are aggregated to non-overlapped aggregates and a smoothing procedure is used for generation of the overlap. Correction operators are defined and used for computation of the new residual in the conjugate gradient method. Several plate problems are solved and results are summarized at the end of this paper.*

1. Introduction

Solution of systems of algebraic equations is a very important part of any analysis based on the finite element method. Especially if the system is large, a suitable method has to be used in order to obtain results within reasonable time. The extremely large systems are usually solved by an iterative method which is equipped with a preconditioner. Efficient preconditioners are usually based on a direct method.

The direct methods are based on the Gaussian elimination algorithm. Exploitation of system properties leads to particular methods such as the LL factorization, the LDL factorization and the LU factorization. It is possible to compute or estimate the number of needed arithmetic operations and cancellation errors occur very rarely. On the other hand, there is the so-called fill-in phenomenon which causes relatively high demands on computer memory, especially in three-dimensional problems. Details about the direct methods can be found in references (Duff et al, 2003; Golub & Van Loan, 1996; Meurant, 1999).

The iterative methods are usually based on a matrix-vector multiplication which results in significant advantage of low computer memory demands. The biggest disadvantage comes from the unknown number of iterations needed for achievement of the prescribed error. There are only estimates of the number of iterations which are usually based on the condition number of the system matrix. Unfortunately, there are many problems with the very large

[*] Doc. Ing. Jaroslav Kruis, Ph.D.: Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7; 166 29 Prague; tel.: +420 224 354 580, fax: +420 224 310 775; e-mail: jk@cml.fsv.cvut.cz

[**] Doc. Dr. RNDr. Petr Mayer: Department of Mathematics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7; 166 29 Prague; tel.: +420 224 355 465, fax: +420 233 332 732; e-mail: pmayer@mat.fsv.cvut.cz

condition number and really many iterations are necessary in order to attain prescribed error.

Preconditioning is an improvement of the iterative methods which is based on transformation of the original system to a new one. In the case of the conjugate gradient method, the residual is transformed with the help of an auxiliary system of equations which should be easily solvable. There are several types of preconditioners such as incomplete factorization but unfortunately no preconditioner is universal. More details about the iterative methods and preconditioning can be found in references (Golub & Van Loan, 1996; Greenbaum, 1997; Saad, 2003).

The most efficient algorithms for solution of large systems of equations are based on the combination of direct and iterative methods. Typical example of such algorithms are domain decomposition methods (Kruis, 2006; Toselli & Widlund, 2005; Magoules, 2006). They are efficient because they can be executed on parallel computers.

This contribution deals with a preconditioning of the conjugate gradient method based on a smoothed aggregation algorithm. The unknowns of the system solved are split in nonoverlapping aggregates and the characteristic functions of aggregates, which are piecewise constant, are defined. Then, the characteristic functions are smoothed and this step results in overlapping aggregates. Tentative and final prolongators are constructed and they are used for definition of local and coarse matrices as well as correction operators. The algorithm starts with a vector which is spread to aggregates and correction step is performed on each aggregate. Then, one coarse level correction is computed and local corrections on aggregates are performed in reverse ordering. After such algorithm, new approximation of the solution is obtained. The algorithm mentioned above is used for computation of new residual in the conjugate gradient method.

2 BOSS Algorithm

The BOSS algorithm is based on aggregation technique and it was introduced by Brezina in reference (Brezina, 1997). Aggregation methods were introduced in thirties of the twentieth century and they were devoted for computations in economics (Leontief, 1951). Later on, the aggregation methods were used for solution of diffusion equation in connection with nuclear engineering. The aggregation served as a special homogenization method. E. Fermi dealt with the aggregates in 1936 and later references (Erlich, 1954) and (Young, 1954) occurred. At this time, the aggregation methods are successfully used even in IT.

A scalar differential equation is used in order to demonstrate basic features of the BOSS method. The choice is motivated by the fact, that discretization of the differential equation by the finite element method result in one unknown at each node. Therefore, there is one-to-one mapping between unknowns and nodes. Aggregation of nodes is identical with aggregation of unknowns. In the case of vector differential equation, each node contains several unknowns and appropriate mapping among nodes and unknowns has to be taken into account. Aggregation of nodes differs from aggregation of unknowns.

First step of the BOSS method collects unknowns into nonoverlapping aggregates.

It creates a disjunct covering of the domain solved. The characteristic functions of aggregates are piecewise constant functions which are equal to one for interior unknowns (unknowns inside of an aggregate) and zero for exterior unknowns (unknowns outside of an aggregate).

Second step is based on a smoothing procedure which is applied to the characteristic functions. It means that jumps on aggregate boundaries from function value 1 to value 0 are exchanged by a smooth function. The smoothing procedure enlarges aggregates because at least the closest neighbour unknowns are attached to the aggregates. The originally nonoverlapping covering is replaced by an overlapping one. The width of overlap can be defined by the smoothing procedure. The wider overlap, the better information exchange among aggregates but also greater memory requirements.

In connection with the finite element method, an aggregate is a set of mesh nodes. Let m denote the number of aggregates. The index set $\mathcal{A}_i$ contains indices of nodes collected in the i-th aggregate. The disjunct covering means that

$$\mathcal{A}_i \cap \mathcal{A}_j = \emptyset \quad \text{for} \quad i \neq j \ . \tag{1}$$

Simultaneously, the following relation is valid

$$\cup_{i=1}^{m} \mathcal{A}_i = \{1, 2, \ldots, n_u\} \ , \tag{2}$$

where n_u denotes the number of all nodes in the mesh.

Tentative prolongator is described by a matrix $\hat{\boldsymbol{P}} \in R^{n \times m}$ and it is a mapping from space R^m to space R^n, where n denotes the number of all unknowns. It is a mapping from the space of aggregates to space of all unknowns. Let the i-th column of the matrix $\hat{\boldsymbol{P}}$ be denoted $\hat{\boldsymbol{p}}_i$. It is a vector with n components, where only components with indices from the set $\mathcal{A}_i$ are equal to one, all remaining components are equal to zero.

Let $\hat{\varrho}$ denote an upper bound of the spectral radius of the matrix $\boldsymbol{A}$ (denoted $\varrho(\boldsymbol{A})$) and let

$$\varrho(\boldsymbol{A}) \leq \hat{\varrho} \leq C_\varrho \varrho(\boldsymbol{A}) \tag{3}$$

be valid. The estimate (3) can be obtained with the help of Gershgorin theorem. The degree of smoothing is denoted by d_S and the length of recursion is defined

$$K = \lfloor \log_3(d_S + 1) \rfloor - 1 \ , \tag{4}$$

where $\lfloor . \rfloor$ is the truncation to the nearest smaller integer.

For $j > 0$ is new variable defined $\hat{\varrho}_j = \frac{\hat{\varrho}}{9^j}$ and new matrices are computed

$$\boldsymbol{A}_0 = \boldsymbol{A} \tag{5}$$

$$\boldsymbol{A}_j = (\boldsymbol{I} - \frac{4}{3}\widehat{\varrho}_{j-1}^{-1}\boldsymbol{A}_{j-1})^2 \boldsymbol{A}_{j-1} \quad \text{for } j = 0, \ldots, K \ . \tag{6}$$

Finally, the prolongator smoother has the form

$$\boldsymbol{S} = \prod_{j=0}^{K} (\boldsymbol{I} - \frac{4}{3}\widehat{\varrho}_j^{-1}\boldsymbol{A}_j) \ . \tag{7}$$

Such approach cannot be used for implementation because there is huge fill-in during matrix multiplication which lead to extremely large memory requirements. It is important to emphasize that the matrix $\boldsymbol{S}$ is a polynomial of the matrix $\boldsymbol{A}$. Construction of such a polynomial also cannot be used due to highly oscillating coefficients which result in numerical instability. The polynomial can be assembled only for very low degrees.

With respect to better behaviour, a smooth prolongator is computed from the tentative one. The final prolongator is denoted by $\boldsymbol{P}$ and it is equal to

$$\boldsymbol{P} = \boldsymbol{S}\hat{\boldsymbol{P}} \; . \tag{8}$$

Difficulties with evaluation of the matrix $\boldsymbol{S}$ can be avoided if the matrix-matrix multiplication $\boldsymbol{S}\hat{\boldsymbol{P}}$ is computed consequently. The sparsity of both matrices must be taken into account otherwise the implementation is very inefficient. Computational complexity $O(\deg(\boldsymbol{S})n)$ can be achieved if the sparsity is employed properly.

The final prolongator $\boldsymbol{P}$ defines new index sets denoted $\mathcal{N}_i$. They are defined by the relationship

$$\mathcal{N}_i = \{j : P_{ji} \neq 0\} \; . \tag{9}$$

There are m index sets $\mathcal{N}_i$. They describe covering with overlap and therefore the relationship

$$\mathcal{N}_i \cap \mathcal{N}_j = \emptyset \quad \text{for} \quad i \neq j \tag{10}$$

is not valid. The number of components of index set $\mathcal{N}_i$ is denoted n_i.

In the following text, the global vector denotes a vector from the space R^n, i.e. a vector with n components. The global vector contains all components of the problem solved. The local vector denotes a vector from the space R^{n_i}, i.e. the vector contains only components connected to the i-th aggregate. Mapping between a global and local vectors can be defined with the help of index sets $\mathcal{N}_i$. Let the global vector be denoted $\boldsymbol{g}$ and the local vector associated with the i-th aggregate be denoted $\boldsymbol{l}_i$. There exist the matrix $\boldsymbol{N}_i$ which can be used for transformation in the form

$$\boldsymbol{g} = \boldsymbol{N}_i \boldsymbol{l}_i \; . \tag{11}$$

The transposed matrix $\boldsymbol{N}_i^T$ selects components connected to the i-th aggregate from the global vector $\boldsymbol{g}$

$$\boldsymbol{l}_i = \boldsymbol{N}_i^T \boldsymbol{g} \; . \tag{12}$$

The local matrix of the i-th aggregate is defined with the help of the matrices $\boldsymbol{N}_i$ in the form

$$\tilde{\boldsymbol{A}}_i = \boldsymbol{N}_i^T \boldsymbol{A} \boldsymbol{N}_i \; . \tag{13}$$

The local correction operator is defined by the relationship

$$\boldsymbol{R}_i = \boldsymbol{N}_i (\tilde{\boldsymbol{A}}_i)^{-1} \boldsymbol{N}_i^T \; . \tag{14}$$

The inverse matrix $(\tilde{\boldsymbol{A}}_i)^{-1}$ is of course not computed. A suitable factorization to the form $\boldsymbol{L}\boldsymbol{L}^T$, $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ or $\boldsymbol{L}\boldsymbol{U}$ is performed.

Except of the local correction operators, there is also coarse correction operator

$$\boldsymbol{R}_0 = \boldsymbol{P}(\tilde{\boldsymbol{A}}_0)^{-1}\boldsymbol{P}^T \,, \tag{15}$$

where the matrix $\boldsymbol{P}$ represents the final prolongator and the matrix $\tilde{\boldsymbol{A}}_0$ is defined by the form

$$\tilde{\boldsymbol{A}}_0 = \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P} \,. \tag{16}$$

Similarly, the inverse matrix $(\tilde{\boldsymbol{A}}_0)^{-1}$ is not computed because $\boldsymbol{L}\boldsymbol{L}^T$, $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ or $\boldsymbol{L}\boldsymbol{U}$ factorization can substitute it.

Let system of algebraic equations be in the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \,. \tag{17}$$

The BOSS method for the solution of system (17) is summarized in Table 1.

Tab. 1: The BOSS method.

1. initial vector $\boldsymbol{z}_0 = \boldsymbol{x}^{(k)}$
2. local corrections on aggregates for i from 1 to m: $\boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \boldsymbol{R}_i(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}_{i-1})$
3. coarse correction $\boldsymbol{v}_m = \boldsymbol{z}_m + \boldsymbol{R}_0(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}_m)$
4. local corrections on aggregates - in reverse ordering for i from $m-1$ to 0: $\boldsymbol{v}_i = \boldsymbol{v}_{i+1} + \boldsymbol{R}_{i+1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{v}_{i+1})$
5. resulting vector $\boldsymbol{x}^{(k+1)} = \boldsymbol{v}_0$

3 Preconditioned conjugate gradient method

The number of iterations in iterative methods can be reduced by application of a preconditioner. Let the system of equations has the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \,, \tag{18}$$

where the matrix $\boldsymbol{A}$ represents e.g. a stiffness matrix and the vector $\boldsymbol{b}$ represents the right hand side, e.g. a vector of prescribed forces. The preconditioning transforms the original system (18) to a new one

$$\boldsymbol{B}\boldsymbol{A}\boldsymbol{D}\boldsymbol{y} = \boldsymbol{B}\boldsymbol{b} \,, \tag{19}$$

where the substitution

$$\boldsymbol{x} = \boldsymbol{D}\boldsymbol{y} \tag{20}$$

is used and the equation is multiplied by a matrix $\boldsymbol{B}$. If the original matrix $\boldsymbol{A}$ is symmetric, the matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ should satisfy $\boldsymbol{B} = \boldsymbol{D}^T$ in order to preserve symmetry. The ideal preconditioner has the form $\boldsymbol{B} = \boldsymbol{D} = \boldsymbol{A}^{-\frac{1}{2}}$ because it results in a system with unit matrix and the solution is equal to the vector on the right hand side. But this choice is nearly always unattainable.

In the case of symmetric positive definite matrix, the system can be solved by the conjugate gradient method. Matrix multiplication on the left hand side in Equation (19) is not performed. It is substituted by a new matrix $\boldsymbol{C}$. The preconditioned conjugate gradient method is summarized in Table 2.

Tab. 2: Preconditioned conjugate gradient method.

initial approximation $\boldsymbol{x}^{(0)}$

initial residuum $\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}$

$$\boldsymbol{h}^{(0)} = \boldsymbol{C}^{-1}\boldsymbol{r}^{(0)}$$

$$\boldsymbol{s}^{(0)} = \boldsymbol{h}^{(0)}$$

iteration $k = 0, 1, ...$

$$\tilde{\alpha}^{(k)} = \frac{(\boldsymbol{r}^{(k)})^T \boldsymbol{h}^{(k)}}{(\boldsymbol{s}^{(k)})^T \boldsymbol{A}\boldsymbol{s}^{(k)}}$$
$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \tilde{\alpha}^{(k)} \boldsymbol{A}\boldsymbol{s}^{(k)}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tilde{\alpha}^{(k)} \boldsymbol{s}^{(k)}$$

$$\boldsymbol{h}^{(k+1)} = \boldsymbol{C}^{-1}\boldsymbol{r}^{(k+1)}$$

$$\tilde{\beta}^{(k)} = \frac{(\boldsymbol{r}^{(k+1)})^T \boldsymbol{h}^{(k+1)}}{(\boldsymbol{r}^{(k)})^T \boldsymbol{h}^{(k)}}$$
$$\boldsymbol{s}^{(k+1)} = \boldsymbol{h}^{(k+1)} + \tilde{\beta}^{(k)} \boldsymbol{s}^{(k)}$$

Preconditioning of the method is described by the step in the form

$$\boldsymbol{h}^{(k+1)} = \boldsymbol{C}^{-1}\boldsymbol{r}^{(k+1)} \ , \tag{21}$$

where $\boldsymbol{r}^{(k+1)}$ denotes the residual and it is recalculated to a new vector $\boldsymbol{h}^{(k+1)}$. If the matrix $\boldsymbol{C}$ is equal to the matrix $\boldsymbol{A}$, the conjugate gradient method finds the solution after

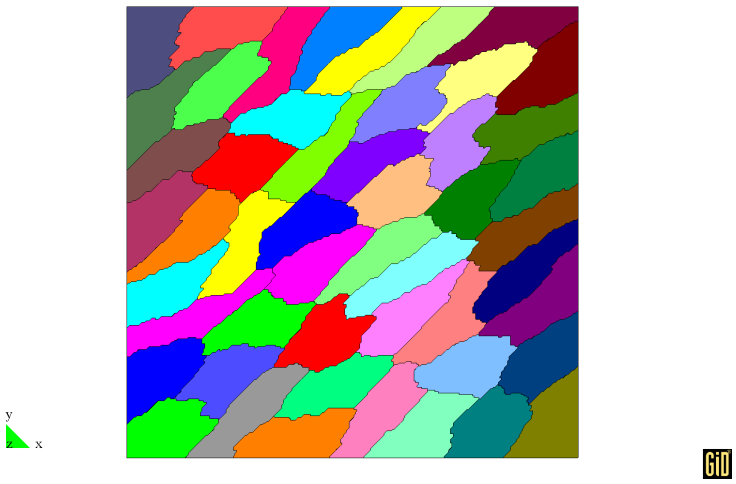

Fig. 1: Example of aggregates on a square domain.

one iteration. Computation of the inverse matrix to the matrix $\boldsymbol{A}$ is much more demanding task than solution of system of equations (18), therefore such choice is never used. The aim is to construct the matrix $\boldsymbol{C}$ as the best approximation of the matrix $\boldsymbol{A}$.

The aim of this contribution is to construct the matrix $\boldsymbol{C}$ which is very close to the matrix $\boldsymbol{A}$ by the BOSS method. Equation (21) can be rewritten to the form

$$\boldsymbol{C}\boldsymbol{h}^{(k+1)} = \boldsymbol{r}^{(k+1)} \,, \tag{22}$$

which is directly suitable for application of the BOSS method. The matrix $\boldsymbol{C}$ is substituted by the matrix $\boldsymbol{A}$ and the system of equations (22) is solved approximately by the BOSS. The vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ defined in the BOSS method are substituted by the vectors $\boldsymbol{r}^{(k+1)}$ and $\boldsymbol{h}^{(k+1)}$ used in the conjugate gradient method.

4 Numerical examples

Numerical examples deal with a plane stress problem and a plate problem described by the Mindlin-Reissner theory. The problems are selected intentionally in order to show advantages as well as disadvantages of the proposed algorithm. The plane stress problem is relatively well-conditioned and the conjugate gradient method solves it efficiently even without any preconditioner. On the other hand, the plate problems are ill-conditioned and the BOSS preconditioner reduces elapsed time. All examples are solved on a square domain and linear elastic material model is assumed. All time informations are in seconds. Figure 1 depicts an example of aggregates generated on a domain. All numerical examples are computed by an implementation which is not optimised yet. Not all operations performed in the code use fully sparsity of the matrices and vectors. Despite it, important features of the BOSS method can be seen.

Tab. 3: The number of iterations and elapsed times for plane stress problems.

NA	NI	CG [s]	TOT [s]	NI	CG [s]	TOT [s]
NP	967	6	7	967	6	7
10	9	5	6	5	6	9
20	10	5	6	5	6	9
30	10	6	7	5	7	10
NP	1924	40	2	1924	40	2
20	15	24	34	8	24	47
30	16	27	37	8	24	49
NP	2880	110	113	2880	110	113
10	17	60	130	10	55	178
20	19	67	111	10	68	165
30	21	79	121	11	78	167

4.1 Plane stress

Table 3 summarizes the behaviour of the conjugate gradient method with the BOSS preconditioner. NA denotes the number of aggregates, NI denotes the number of iterations, CG denotes the time spent in computer subroutine dealing with the conjugate gradient method and TOT denotes the total time of computation. Difference between CG and TOT is a time devoted to reading input files, writing results to output files, the stiffness matrix assembling and computation of auxiliary matrices and arrays for the BOSS method. NP denotes the conjugate gradient method without preconditioner and LDL denotes time of $\boldsymbol{LDL}^T$ factorization. Left part of the table describes results for the degree of smoothing $d_S = 2$ while the right part describes results for $d_S = 3$. Three meshes with 100x100, 200x200 and 300x300 finite elements are used. The results for particular mesh can be found in the upper, middle or lower part of Table 3.

4.2 Plate problem

A plate problem described by the Mindlin-Reissner theory is studied. Three meshes with 100x100, 200x200 and 300x300 finite elements are used. The structure of Table 4 is identical with the structure of Table 3. The results for particular mesh can be found in the upper, middle or lower part of Table 4.

5 Conclusion

The numerical examples show clearly that the BOSS method is very efficient preconditioner for the conjugate gradient method for very ill-conditioned problems. It is caused by the local aggregate matrices and local correction operators as well as the global cor-

Tab. 4: The number of iterations and elapsed times for plate problems.

NA	NI	CG [s]	TOT [s]	NI	CG [s]	TOT [s]
LDL	-	-	3	-	-	3
NP	12875	25	26	12875	25	26
10	56	7	8	13	11	23
20	67	7	9	14	12	25
30	67	7	9	14	12	25
LDL	-	-	168	-	-	168
NP	21044	185	189	21044	185	189
20	83	147	175	38	37	61
30	63	149	173	42	41	70
NP	42908	870	876	42908	870	876
10	177	226	291	59	117	217
20	226	324	366	68	159	229
30	266	453	495	76	196	266

rection operator. Assembling of the matrices and correction operators takes some time which is worthwhile in the case of ill-conditioned problems. If the condition number is not so large, this type of preconditioning is not efficient.

Acknowledgement

Financial support for this work was provided by project number 103/07/1455 of Czech Science Foundation. The financial support is gratefully acknowledged.

References

Brezina, M. (1997) Robust Iterative Methods on Unstructured Meshes. Ph.D. Thesis, University of Colorado at Denver.

Duff, I.S.,Erisman, A.M. & Reid, J.K. (2003) Direct Methods for Sparse Matrices. Oxford Science Publications, Clarendon Press Oxford.

Erlich, R. & Hurwitz, H. (1954) Multigroup Methods for Neutron Diffusion Problems. Nucleonics, vol. 12, n. 2, pp. 1-23.

Golub, G.H. & Van Loan, C.F. (1996) Matrix Computations. The Johns Hopkins University Press, Third edition, Baltimore, USA.

Greenbaum, A. (1997) Iterative Methods for Solving Linear Systems. Frontiers in Applied Mathematics, SIAM, Philadelphia, USA.

Kruis, J. (2006) Domain Decomposition Methods for Distributed Computing. Saxe-Coburg Publications, Kippen, Stirling, Scotland, UK.

Leontief, W. (1951) The Structure of the American Economy 1919-1939. Oxford University Press, New York.

Magoules, F. (2007) Mesh Partitioning Techniques and Domain Decomposition Methods. Saxe-Coburg Publications, Kippen, Stirling, Scotland, UK.

Meurant, G. (1999) Computer Solution of Large Linear Systems. Studies in Mathematics and its Applications, 28, North-Holland, Elsevier, Amsterdam, The Netherlands.

Saad, Y. (2003) Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia, USA.

Toselli, A. & Widlund, O. (2005) Domain Decomposition Methods - Algorithms and Theory. Springer Series in Computational Mathematics, vol. 34, Springer-Verlag, Berlin, Germany.

Young, D.M. (1954) Iterative Method for Solving Partial Differential Equations of Elliptic Type. Trans. Amer. Math. Soc., 43, pp. 92-111.