Fast solution methods

Tetsuya Sakuma, Stefan Schneider, Yosuke Yasuda

Fast Solution Methods

Tetsuya Sakuma[1], Stefan Schneider[2], and Yosuke Yasuda[3]

[1] Graduate School of Frontier Sciences, University of Tokyo, 5–1–5 Kashiwanoha, Kashiwa, Chiba 277–8563, Japan
sakuma@k.u-tokyo.ac.jp
[2] CNRS/LMA, 31 chemin Joseph–Aiguier, 13402 Marseille Cedex 20, France
schneider@lma.cnrs-mrs.fr
[3] Graduate School of Frontier Sciences, University of Tokyo, 5–1–5 Kashiwanoha, Kashiwa, Chiba 277–8563, Japan
yyasuda@k.u-tokyo.ac.jp

Summary. The standard boundary element method applied to the time harmonic Helmholtz equation yields a numerical method with $\mathcal{O}(N^3)$ complexity when using a direct solution of the fully populated system of linear equations. Strategies to reduce this complexity are discussed in this paper. The $\mathcal{O}(N^3)$ complexity issuing from the direct solution is first reduced to $\mathcal{O}(N^2)$ by using iterative solvers. Krylov subspace methods as well as strategies of preconditioning are reviewed. Based on numerical examples the influence of different parameters on the convergence behavior of the iterative solvers is investigated. It is shown that preconditioned Krylov subspace methods yields a boundary element method of $\mathcal{O}(N^2)$ complexity. A further advantage of these iterative solvers is that they do not require the dense matrix to be set up. Only matrix–vector products need to be evaluated which can be done efficiently using a multilevel fast multipole method. Based on real life problems it is shown that the computational complexity of the boundary element method can be reduced to $\mathcal{O}(N \log^2 N)$ for a problem with N unknowns.

12.1 Introduction

The standard boundary element method, that is the calculation and storage of the dense matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ and the direct solution of the system of linear equations, requires $\mathcal{O}(N^2)$ memory locations and $\mathcal{O}(N^3)$ arithmetic operations. This makes the method hardly applicable to the solution of problems of practical interest. However, the fact that the BEM requires only a discretization of the boundary Γ of the fluid domain Ω is advantageous when solving radiation or scattering problems involving complex shaped radiators or scatterers.

The following chapter discusses possibilities to increase the efficiency of the BEM such that it will be applicable to problems of practical interest. Two directions are issued. Firstly, the iterative solution of the system of linear equations and secondly the efficient evaluation of the product of the matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ with a vector as needed within the iterative solution process.

12.2 Iterative Solution

Iterative methods have been widely used for solving linear systems in various fields. For the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, an iterative method gives successive approximations $\boldsymbol{x}_i$ at each step i from an initial approximation $\boldsymbol{x}_0$ in a systematic way, until the residual norm $||\boldsymbol{r}_i|| = ||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i||$ sufficiently decreases. If the iterative process rapidly converges, iterative methods are much faster than direct methods, such as Gaussian elimination.

Iterative methods are classified into two groups: stationary methods and nonstationary methods. In the former methods, the coefficients for renewal of the approximation $\boldsymbol{x}_i$ do not depend on the iteration count. Jacobi, Gauss–Seidel, Successive Over–Relaxation (SOR), and Symmetric Successive Over–Relaxation (SSOR) methods, etc. belong to this group. The latter methods use iteration–dependent coefficients so that the computations can involve renewal information obtained at each iteration. The Krylov subspace methods are most popular in this group.

The Krylov subspace methods, which were selected as one of the top 10 algorithms of the 20th century in *SIAM News* [19], are very effective for large systems of linear equations. A Krylov method successively gives the approximations $\boldsymbol{x}_i$ in the process of increasing the dimensionality of the Krylov subspace, which is spanned by the Krylov sequence generated by the initial residual $\boldsymbol{r}_0$ and the system matrix $\boldsymbol{A}$. The Krylov subspace methods become more powerful when they are used with preconditioning technique for improvement of iterative convergence. One special feature of the methods is that the system matrix $\boldsymbol{A}$ is not explicitly necessary, and only the products of the matrix $\boldsymbol{A}$ with vectors (matrix–vector products) are required. Therefore, the Krylov methods are applicable with fast evaluation of the products without generating $\boldsymbol{A}$.

In the field of computational acoustics, there exist many examples using the Krylov methods: FE [45, 46] and BE [43, 55] analyses for interior problems, BE analyses for exterior problems [6, 54, 57, 60], structure–acoustic problems [17], etc.

This chapter describes the outline of the Krylov subspace iterative methods and its application to the BEM. For the details about iterative methods, see References [8, 10, 53, 66], for example.

12.2.1 The Krylov Subspace Methods

Consider that the linear system of equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{12.1}$$

is solved with an iterative method. The initial residual $\boldsymbol{r}_0$ is expressed as:

$$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0, \tag{12.2}$$

where $\boldsymbol{x}_0$ denotes the initial approximate solution of $\boldsymbol{x}$. The Krylov subspace methods are iterative methods in which the approximate solution $\boldsymbol{x}_n$ is taken as satisfying the next condition,

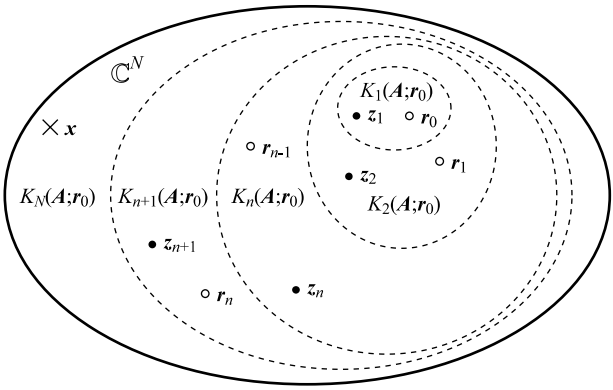

Fig. 12.1 Relationship among subspaces.

$$\boldsymbol{x}_n = \boldsymbol{x}_0 + \boldsymbol{z}_n, \quad \boldsymbol{z}_n \in K_n(\boldsymbol{A}; \boldsymbol{r}_0), \tag{12.3}$$

where $K_n(\boldsymbol{A}; \boldsymbol{r}_0)$ denotes the n–dimensional Krylov subspace spanned by a Krylov sequence $\{\boldsymbol{A}^i \boldsymbol{r}_0\}_{i \geq 0}$:

$$K_n(\boldsymbol{A}; \boldsymbol{r}_0) = \mathrm{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r}_0\}. \tag{12.4}$$

The iteration residual $\boldsymbol{r}_n$ is expressed as follows:

$$\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{z}_n \in K_{n+1}(\boldsymbol{A}; \boldsymbol{r}_0). \tag{12.5}$$

Figure 12.1 illustrates relationship among subspaces. Extension of the subspace with each iteration enables one to express better approximations of $\boldsymbol{x}_n$.

Krylov subspace methods require an approach for identification of the approximate solution $\boldsymbol{x}_n$ as follows.

- The minimum norm residual approach: identify $\boldsymbol{r}_n$ for which the norm $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}||$ is minimal over $K_n(\boldsymbol{A}; \boldsymbol{r}_0)$.

$$||\boldsymbol{r}_n|| = \min_{\boldsymbol{x} \in \boldsymbol{x}_0 + K_n(\boldsymbol{A}; \boldsymbol{r}_0)} ||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}||. \tag{12.6}$$

- The Ritz–Galerkin approach: identify $\boldsymbol{r}_n$ that $\boldsymbol{r}_n$ is orthogonal to the current subspace.

$$\boldsymbol{w}^H \boldsymbol{r}_n = 0, \quad \boldsymbol{w} \in K_n(\boldsymbol{A}; \boldsymbol{r}_0), \tag{12.7}$$

  where $\boldsymbol{w}^H$ denotes the conjugate transpose of $\boldsymbol{w}$.

Classification

Lanczos, Arnoldi and Bi–Lanczos Types

In the Krylov subspace methods, an orthogonal basis for the Krylov subspace is required for stability of computation. The Krylov subspace methods are classified

by the algorithms for production of an orthogonal basis and by the approaches for identification of the approximate solution $\boldsymbol{x}_n$ mentioned above.

If $\boldsymbol{A}$ is a Hermitian matrix, one can compute an orthogonal basis for the Krylov subspace $K_n(\boldsymbol{A}; \boldsymbol{r}_0)$ based on only three–term reccurence relations. This process is called the Lanczos process. The methods based on the Lanczos process have the advantage that the operation count and the required memory per iteration step do not increase with the number of iteration.

If $\boldsymbol{A}$ is a non–Hermitian matrix such as that obtained with the BEM, one cannot compute an orthogonal basis with short reccurence relations. One have to keep all elements of the orthogonal basis computed in the iterative process, to obtain the next element. This process is called the Arnoldi process. The methods based on the Arnoldi process can be applied any regular matrices, whereas they have a major disadvantage that the operation count and required memory per iteration step rise linearly with the number of iteration.

There is an inexpensive process for computation of bases for non–Hermitian matrices, where bi–orthogonal bases are computed instead of an orthogonal basis. In addition to the Krylov subspace $K_n(\boldsymbol{A}; \boldsymbol{r}_0)$, one can consider another Krylov subspace $K_n(\boldsymbol{A}^H; \boldsymbol{r}_0^*)$, and construct bi–orthogonal bases $\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{u}_j\}$ for the two subspaces, satisfying the following relation

$$\boldsymbol{v}_i^H \boldsymbol{u}_j = 0, \quad i \neq j, \tag{12.8}$$

where $\boldsymbol{r}_n^*$ denotes the shadow residual, satisfying $\boldsymbol{r}_0^{*H} \boldsymbol{r}_0 \neq 0$. This process is called the Bi–Lanczos process, where one can compute the bi–orthogonal bases with only three–term reccurence relations, like the Lanczos process. The main drawback of the methods based on this process is the possibility of break down in the iterative process, which does not occur in methods based on the Lanczos process. Since the above approaches Equations (12.6) and (12.7) for identification of $\boldsymbol{x}_n$ cannot be directly used with the Bi–Lanczos algorithm, the following approach is used instead.

- The Petrov–Galerkin approach: find $\boldsymbol{r}_n$ so that $\boldsymbol{r}_n$ is orthogonal to the other subspace $K_n(\boldsymbol{A}^H; \boldsymbol{r}_0^*)$

$$\boldsymbol{w}^H \boldsymbol{r}_n = 0, \quad \boldsymbol{w} \in K_n(\boldsymbol{A}^H; \boldsymbol{r}_0^*). \tag{12.9}$$

Table 12.1 shows popular Krylov subspace methods classified by the process for computation of the basis and the approach for identification of the approximate solution.
There is a short description of the representative methods below.

Conjugate Gradient (CG) [38] : The most basic method based on the Lanczos process. This method is used for linear systems with positive–definite Hermitian matrices, and extremely effective. This cannot be directly used for the BEM.

Generalized Minimal Residual (GMRes) [52] : This method is based on the Arnoldi process and known as one of the robust methods. This is applicable to the linear systems with non–Hermitian matrices and often used for sound field analyses using the

Table 12.1 Classification of Krylov subspace methods.

	Lanczos	Arnoldi	Bi–Lanczos
Ritz–Galerkin	CG	FOM	–
Minimum norm residual	MinRes	GMRes	–
Petrov–Galerkin	–	–	BiCG

BEM. The major drawback is that the amount of operation and memory required per iteration step rise linearly with the number of iteration. Thus, one needs to restart the process every l iterations to overcome this limitation (GMRes(l)). It is quite difficult to choose an appropriate value for l, since GMRes may converge slowly, or completely fail to converge if l is too small.

Bi–Conjugate Gradient (BiCG) [26, 41] : This method is based on the Bi–Lanczos process and can be applied to the systems with non–Hermitian matrices. Unlike the methods based on the Arnoldi process, the amount of operation and required memory per iteration step do not rise with the number of iteration. However, BiCG requires not only the original system matrix but also its conjugate transpose. Another drawback is that the method might be break down, or converge very irregular.

BiCG–based Methods

BiCG is an alternative of the methods based on the Arnoldi process in efficiency, whereas it has some disadvantages mentioned above. A lot of variants of BiCG, here we call the BiCG–based methods, have been developed to improve these disadvantages. All of the methods shortly described below do not need the conjugate transpose of the system matrix. The convergence behavior of the methods of this group including BiCG is hardly known.

Conjugate Gradient Squared (CGS) [63] : CGS often converges or diverges much faster than BiCG, because the residual polynomial of CGS is the square of that of BiCG. The convergence behavior may be irregular for large problems, due to round–off errors.

Biconjugate Gradient Stabilized (BiCGStab) [65] : This method has smoother convergence behavior than CGS, keeping its rapid convergence. This is regarded as the product of BiCG and GMRes(1).

Generalized Product Type of Biconjugate Gradient (GPBiCG) [75] : This method is derived from generalization of BiCG–based methods. This is effective when the eigenvalues of the system matrix are complex.

BiCGStab2 [33], BiCGStab(l) [61] and TFQMR [28], etc. are also well known.

Methods Based on Normal Equations

There is another class of Krylov subspace methods for non–Hermitian matrices. These methods are based on the application of the methods for Hermitian matrices to the normal equations. Popular methods of this class are CGNE and CGNR, in

which CG is applied to the systems $(\boldsymbol{A}^H\boldsymbol{A})\boldsymbol{x} = \boldsymbol{A}^H\boldsymbol{b}$ and $(\boldsymbol{A}\boldsymbol{A}^H)(\boldsymbol{A}^{-H}\boldsymbol{x}) = \boldsymbol{b}$, respectively. These methods have a disadvantage that the matrices of normal equations are more ill–conditioned than the original matrices. The convergence may be slow.

12.2.2 Preconditioning

Preconditioning is a well known technique for improvement of convergence when one attempts to solve linear systems using iterative methods. A matrix $\boldsymbol{M} = \boldsymbol{M}_1\boldsymbol{M}_2$, which approximates the system matrix $\boldsymbol{A}$ in some way, transforms the original system into the following system with more favorable properties for iterative solution:

$$\boldsymbol{M}_1^{-1}\boldsymbol{A}\boldsymbol{M}_2^{-1}(\boldsymbol{M}_2\boldsymbol{x}) = \boldsymbol{M}_1^{-1}\boldsymbol{b}. \tag{12.10}$$

In general, a good preconditioner $\boldsymbol{M}$ should meet the following requirements: the preconditioned system should be easy to solve, and the preconditioner should be cheap to construct and apply. For the detail review on preconditioning techniques, an intensive review paper [10] can be available. We describe below some preconditioning techniques, which can be used for the BEM.

Diagonal Preconditioning

The preconditioner $\boldsymbol{M}$ consists of just the diagonal of the matrix $\boldsymbol{A}$. This preconditioning is applicable without extra memory and time, and easy to apply to almost all kinds of the BEM. This preconditioning was reported to be effective for the BEM for some fields, for example, the thermal and elastic BEM [47], whereas reported not so effective for the acoustic BEM [43, 72].

Incomplete Factorizations

Preconditioners based on incomplete factorizations of the system matrix, such as incomplete LU (ILU) and incomplete Cholesky (IC) factorizations, have been widely used. In ILU preconditioning, the system matrix $\boldsymbol{A}$ is incompletely factorized as a preconditioner $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{U} \approx \boldsymbol{A}$, where $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular matrices, respectively. If the system matrix is Hermitian and positive definite, IC factorization can be applied, which is a special case of ILU factorizations.

It is better to make ILU factorization near to complete factorization for good convergence, but resulting in high cost. Actually, complete LU factorization is equivalent to Gaussian elimination, and $\boldsymbol{L}$ and $\boldsymbol{U}$ obtained by complete factorization generally have many fill–in (nonzero) entries. There are some dropping strategies for discarding fill–ins to make ILU preconditioners. ILU(k) allows fill–ins only to a certain level k. ILU(0) corresponds to allowance of fill–ins only at positions for which the corresponding entries of $\boldsymbol{A}$ are nonzero. ILU(k) has some disadvantages; for example, the computational cost rapidly increases with k, and the storage and computational cost

cannot be predicted in advance. ILUT(τ, p) [51] is a threshold–based ILU factorization. This employs a dual dropping strategy and more powerful than ILU(k); the threshold τ for dropping elements having small magnitude and the maximum number p of fill–in elements in each row of $\boldsymbol{L}$ and $\boldsymbol{U}$ are used to control the computational costs for factorization and application of the preconditioner.

ILU preconditioners are usually applied to sparse linear systems. They are very expensive when this is directly applied to the system with a dense system matrix, typically obtained with the BEM. Some techniques have been proposed to overcome this limitation. Please see 12.2.3.

Sparse Approximate Inverses (SPAI)

The basic idea of this class is that $\boldsymbol{M}^{-1} \approx \boldsymbol{A}^{-1}$ is explicitly computed and used as a preconditioner. The computational cost for naive construction and application of $\boldsymbol{M}^{-1}$ is very high, since $\boldsymbol{M}^{-1}$ is usually dense. In SPAI preconditioning, a *sparse* matrix $\boldsymbol{M}^{-1} \approx \boldsymbol{A}^{-1}$ is computed as the solution of the constrained minimization problem

$$\min_{\boldsymbol{M}^{-1} \in \mathcal{S}} ||\boldsymbol{A}\boldsymbol{M}^{-1} - \boldsymbol{I}||_F, \tag{12.11}$$

where $\mathcal{S}$ is a set of sparse matrices and $||\cdot||_F$ denotes the Frobenius norm of a matrix. This results in n independent least–squares problems for each columns of $\boldsymbol{M}^{-1}$. The computational cost mainly depends on how to give the sparsity pattern $\mathcal{S}$. To use this for dense linear systems, $\mathcal{S}$ must be sufficiently sparse.

12.2.3 Application to the BEM

The Krylov subspace methods require repeated calculation of matrix–vector products from the system matrix $\boldsymbol{A}$ and vectors, which occupies quite a large amount of operations in the iterative process. If $\boldsymbol{A}$ is a $N \times N$ dense matrix as generally obtained with the BEM, the operation count for a matrix–vector product is $\mathcal{O}(N^2)$. Thus, if the number of iteration is sufficiently smaller than N, the Krylov methods remarkably reduce the total operation count for solving the system compared with direct methods, which require $\mathcal{O}(N^3)$ operations. In addition, more reduction of the operation count can be achieved by using methods for efficient evaluation of the matrix–vector products, such as the fast multipole method (FMM). The required memory is also reduced sharply by using such methods, because it is not necessary to store the entire system matrix $\boldsymbol{A}$. For more details on the efficient evaluation of matrix–vector products, please see Section 12.3.

Iterative Methods for the BEM

The BEM gives a linear system with a non–Hermitian matrix, to which the iterative methods based on the Arnoldi or Bi–Lanczos process and based on normal equations are applicable. For an advanced BEM that does not store the entire system matrix, it is difficult to apply the methods that require the conjugate transpose of the system matrix, such as BiCG, QMR and the methods for the normal equations.

Preconditioners for the BEM

There are not many preconditioners effective for non–Hermitian dense matrices, because preconditioners usually have been proposed for the sparse matrices. For the preconditioners for dense linear systems, see References [14,15,67,70] and the references therein. Moreover, fewer preconditioners are applicable to the advanced BEM that does not store the entire system matrix.

One class of preconditioners for the BEM is based on the splitting of the boundary integral operators [5,14]. The system matrix $\boldsymbol{A}$ is splitted into two matrices $\boldsymbol{A}_{\text{near}}$ and $\boldsymbol{A}_{\text{far}}$, the former of which is sparse and composed of the influence parts between near elements. ILU factorization is done for the matrix $\boldsymbol{A}_{\text{near}}$, and the results are used for the preconditioning to the original system with $\boldsymbol{A}$. This technique is very suitable for the advanced BEM that does not store the entire $\boldsymbol{A}$, because this type of the BEM requires to generate $\boldsymbol{A}_{\text{near}}$ in the same manner as the standard BEM. This technique has been proved to be very effective through some investigations, where practical acoustic problems were calculated with the advanced BEM using the regular grid method (RGM) [59] and the FMM [59, 72]. For the standard BEM, a simple way of making a substitute of $\boldsymbol{A}_{\text{near}}$ is to take the tri–diagonal band of $\boldsymbol{A}$ together with the anti–diagonal corner elements a_{1n} and a_{n1} [5].

12.2.4 Convergence Behavior for the BEM

The rate at which an iterative method converges relates directly to its computational time. Regarding non–Hermitian matrices, the convergence of iterative methods is not clear, and depends greatly on the properties of the matrices [8].

There are some studies on the convergence for the acoustic BEM [2–5,14–17,37, 43,44,59,72,74]. Many of them are especially for exterior problems with Burton–Miller formulation [3,16,37,59,74], because iterative methods do not converge well in this case. Preconditioning techniques were mainly discussed in References [5,14–16,37]. Detailed comparison through numerical experiments has been done for a variety of practical problems [43,59,72].

The following refers to general tendencies on the convergence, giving the examples using the collocation BEM with constant elements. The multilevel fast multipole algorithm (MLFMA) is used for efficient evaluation of matrix–vector products. For more details on MLFMA, see Section 12.3. The following equation is used as a stopping criterion for the linear system,

$$\frac{|\boldsymbol{r}_i|}{|\boldsymbol{b}|} = \frac{|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i|}{|\boldsymbol{b}|} \leq \varepsilon, \tag{12.12}$$

where $|\cdot|$ is the Euclidean norm. $\varepsilon = 10^{-6}$ is used, unless noted otherwise. Instead of the number of iteration, the number of matrix–vector products is counted. This is because the operation of matrix–vector products accounts for the greater part of the iterative process, and iterative methods have different numbers of matrix–vector products per iteration. For example, CGS requires two matrix–vector multiplications per iteration, whereas GMRes requires one.

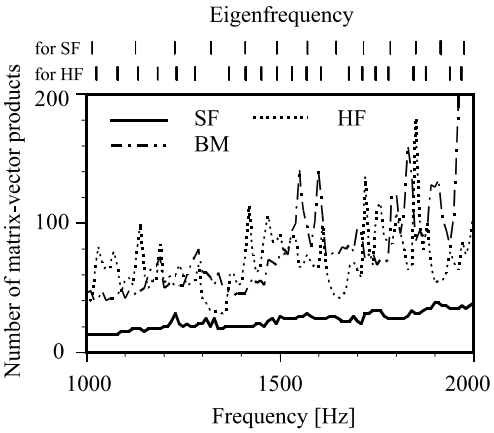

Fig. 12.2 Effect of different kinds of formulation on the number of matrix–vector products for an exterior problem with a vibrating cube. The iterative method is unpreconditioned BiCGStab, DOF is 6,144, and $\varepsilon = 10^{-3}$.

Effect of Formulation

The convergence behavior is affected by the formulation of the BEM: basic formulation (singular formulation: SF), normal derivative formulation (hypersingular formulation: HF), and formulation by Burton and Miller (BM) [13] to avoid fictitious eigenfrequency difficulties. Figure 12.2 shows the history of the residual norms for an exterior problem with a vibrating cube. The fictitious eigenfrequencies for SF and for HF are also shown in the upper part of the figure. The convergence with HF and with BM is slower than that with SF at all frequencies. This is because the matrices obtained with HF generally have eigenvalues which are not clustered compared to those with SF (i.e., HF matrices are ill–conditioned), and BM matrices inherit the ill condition from HF matrices.

It has been reported that the hypersingular equation can be reduced to weakly singular one by using the high–order Galerkin BEM, resulting in improvement of the matrix condition and the convergence behavior with BM [37, 74].

Effect of Boundary Shapes

The shape of boundary can greatly affect the convergence behavior. There is a tendency that the convergence is more rapid with smoother surface [43, 72]. Figure 12.4 shows the history of the residual norms for two interior models (Figure 12.3), which have nearly the same DOF and ratio of the element width to the wavelength. Slower convergence is seen for the problem with a complex shape (auditorium) than with a simple shape (cube), when both the problems are under the same boundary condition.

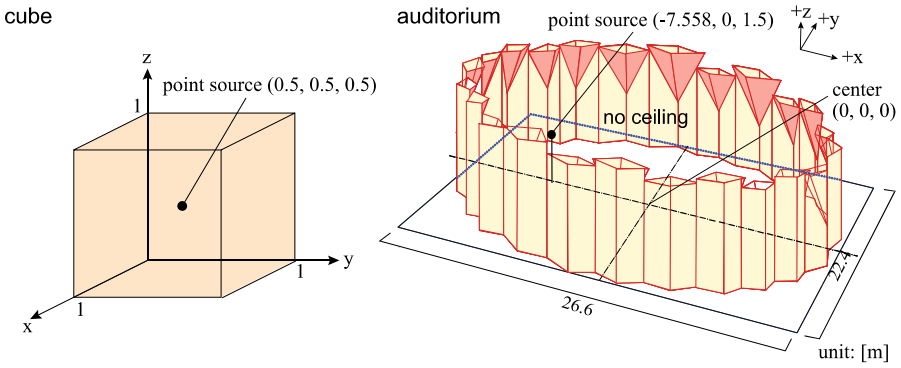

Fig. 12.3 Analysis models for interior problems: a cube (simple shape) and an auditorium (complex shape).

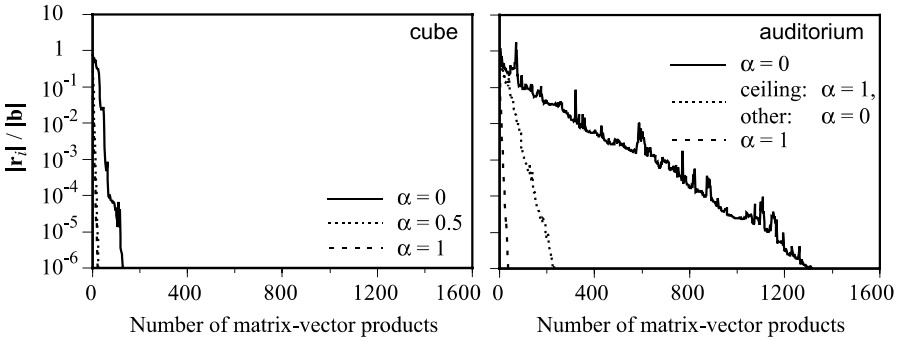

Fig. 12.4 Effect of shapes and boundary conditions on the iteration residual. α is the absorption coefficient on the surfaces. The formulation is SF, the iterative method is unpreconditioned GPBiCG, DOF is 24,576 (cube) and 24,514 (auditorium).

Effect of Boundary Conditions

The larger the absorption coefficient α is, the faster the convergence is [43, 72], as is seen in Figure 12.4. This is the general tendency, independent of calculated problems and iterative methods. As shown in Figure 12.5, a little absorption greatly improves the convergence, compared to the case of perfectly rigid boundaries.

Effect of Preconditioning

The following refers to the effect of ILUT(τ, p) [51], which is one of the powerful preconditioners, on the convergence behavior with Burton–Miller formulation. In particular, it focuses on the effect of the parameter p, which greatly affects memory

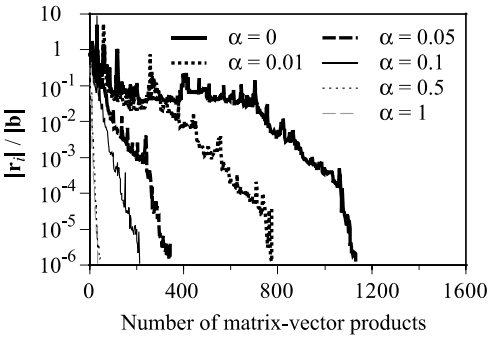

Fig. 12.5 Effect of absorption coefficients α on the iteration residual for an auditorium at 63 Hz. The formulation is SF, the iterative method is unpreconditioned BiCGStab, and DOF is 6,110.

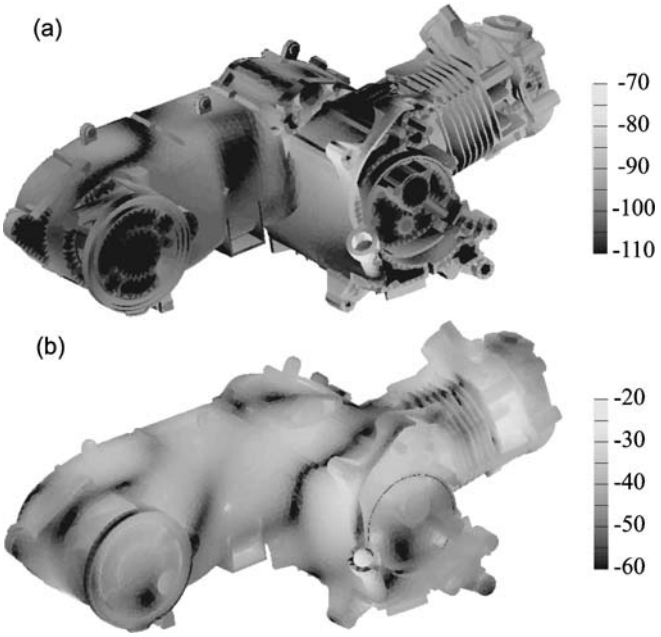

Fig. 12.6 An engine model: (a) relative vibration velocity level distribution, and (b) relative SPL distribution at 1,977 Hz. The formulation is BM, and DOF is 42,152.

requirements. This preconditioner is referred to as ILUT(p) with $\tau = 10^{-5}$ in the below.

Figure 12.7 shows the history of the residual norms with three kinds of iterative methods, for an exterior problem with a vibrating engine, Figure 12.6. In this case,

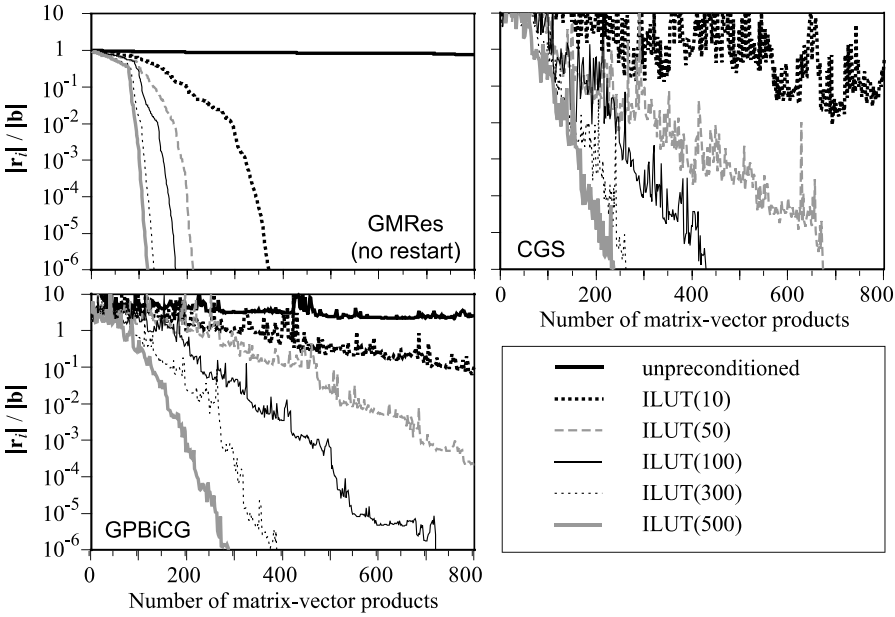

Fig. 12.7 Effect of the number of fill–in elements p for ILUT(p) on the iteration residual for the engine problem at 3 kHz. The formulation is BM, and DOF is 42,152.

all the unpreconditioned methods do not converge at all, while ILUT preconditioning remarkably improves the convergence of all the methods. In addition, the effect of ILUT(p) increases with p. As for GMRes(∞) (no restarting), even small p greatly improves the convergence, making GMRes(∞) the fastest among the methods. From the viewpoint of convergence behavior, GMRes(∞) with ILUT preconditioning is generally recommended for calculation in Burton–Miller formulation.

Figure 12.8 shows the effect of the restart number l for GMRes(l) on the iteration residual. The residual norms stagnate from every restart point [59, 72], resulting in slower convergence of GMRes(l) than CGS and GPBiCG, depending on ILUT(p), see Figure 12.7. It is stated that one had better avoid restarting in GMRes(l) for reducing computational time. If the restarting cannot be avoided due to the restriction of memory storage, preconditioned BiCG–based methods possibly converge faster than GMRes(l). For fast convergence of GMRes(l) with ILUT(p), one should consider the trade–off between the two parameters l and p in the restriction of memory storage.

12.3 Efficient Evaluation of the Matrix–Vector Product

Within the iterative solution of the dense linear system arising from a solution of the Helmholtz equation using Boundary Element Method, namely, when calculating

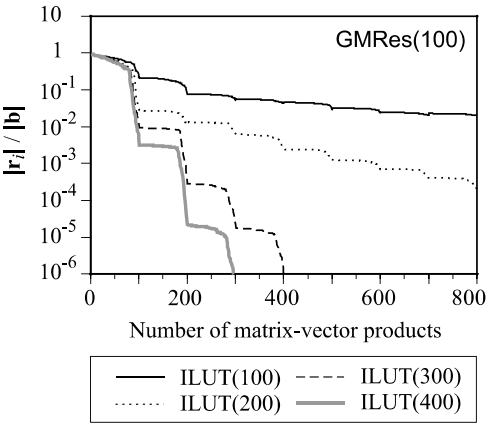

Fig. 12.8 Effect of the restart number l for GMRes(l) on the iteration residual for the engine problem at 3 kHz. The formulation is BM, $l = 100$, and DOF is 42,152.

the matrix–vector product $\boldsymbol{z} = (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{u}$ one of the following quantities has to be evaluated

$$z_j = \int_\Gamma \varphi_j(\mathbf{x})\varphi(\mathbf{x})\, d\Gamma \cdot \boldsymbol{u} - \int_\Gamma \varphi_j(\mathbf{y}) \int_\Gamma k(\mathbf{x}-\mathbf{y})\varphi(\mathbf{x})\, d\Gamma\, d\Gamma \cdot \boldsymbol{u} \quad (12.13)$$

when using a Galerkin method,

$$z_j = u_j - \int_\Gamma k(\mathbf{x}-\mathbf{y}_j)\varphi(\mathbf{x})\, d\Gamma \cdot \boldsymbol{u} \quad (12.14)$$

when using a Collocation method or

$$z_j = u_j - w_i k(\mathbf{x}_i - \mathbf{y}_j)\varphi(\mathbf{x}_i) \cdot \boldsymbol{u} \quad (12.15)$$

when using a Nyström method for all $j = 1, \ldots, N$ with N being the number of unknowns of the given problem. Due to the fact that the kernel

$$k(\mathbf{x}-\mathbf{y}) = \alpha G(\mathbf{x},\mathbf{y}) + \beta \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) \cdot \nu_{\mathbf{x}} \quad (12.16)$$

with the fundamental solution $G(\mathbf{x},\mathbf{y})$ has non–local support this operation costs $\mathcal{O}(N^2)$ arithmetic operations.

Several possibilities to achieve a reduction of the complexity have been proposed in the literature.

One approach consists in using a suitable wavelet basis together with a drooping strategy to sparsify the dense matrix $\boldsymbol{A}$ [12, 21, 27, 39]. The most challenging part of this approach is the direct evaluation of the matrix $\boldsymbol{A}$ in the wavelet basis.

A second approach consists in a low rank approximation of $\boldsymbol{A}$ together with efficient $\mathcal{H}$–matrix techniques [9, 35]. Such an approach is based only on smoothness properties of the kernel function without necessarily knowing it explicitly. This

is a very important point when considering numerical implementation. A possible change of the kernel function, as long as the smoothness properties are satisfied, does not require a complete recoding of the numerical algorithms.

A third approach is based on a suitable approximation of a specific kernel function in order to separate the $\mathbf{x}$ and $\mathbf{y}$ dependency of the kernel function. For two points $\mathbf{x}$ and $\mathbf{y}$ with $|\mathbf{x}-\mathbf{y}| > \varrho > 0$ the kernel Equation (12.16) is a smooth function, thus an approximation $k(\mathbf{x},\mathbf{y}) \approx h(\mathbf{y}-\mathbf{y}_c)\mu(\mathbf{y}_c-\mathbf{x}_c)g(\mathbf{x}-\mathbf{x}_c)$ can be used to replace the original kernel function. Such an approximation is in general linked to a specific kernel $k(\cdot)$. Hence, changing the kernel function in general requires a complete recoding of the numerical algorithms. The use of such a kernel approximation forms the basis of methods like regular grid method [11], panel clustering [36,56] or the fast multipole method [18, 31]. The latter method, especially its multilevel variant, seems to be the most widely accepted method as it covers the very low to high frequency range. An overview of the state–of–the–art of the fast multipole method can be found in the textbook [32].

12.3.1 Basic Concept of the Fast Multipole Method

The usage of the Fast Multipole Method for the solution of boundary value problems for the Laplace equation goes back to Greengard and Rohklin [31]. The application of this method for two–dimensional scattering problems can be found in [49]. It was again Rohklin [50] who extended it to the three–dimensional case. The Fast Multipole Method for the three–dimensional Helmholtz equation will be derived related to the work done by Anderson [7] and Rahola [48] which is often referred to as the "fast multipole method without multipoles".

The Fast Multipole Method was first used to calculate particle interactions. From this point of view (integration replaced by summation) Equation (12.13) to Equation (12.15) can be regarded as the communication of every point $\mathbf{y}_j$ on the surface with all other points $\mathbf{x}_i$ on the surface, hence the direct evaluation of these equations has $\mathcal{O}(N^2)$ complexity. Avoiding the evaluation of the interaction of all $(\mathbf{y}_j, \mathbf{x}_i)$ pairs will reduce the complexity of the algorithm. This can be achieved by splitting the direct path $\mathbf{y}_j - \mathbf{x}_i$. Therefore sets of points $\mathbf{z}_i$ are introduced where the information of assigned points $\mathbf{x}_i$ is aggregated. Thereafter, only the interaction between aggregation points is evaluated. Information is redistributed from the points $\mathbf{z}_j$ to the associated points $\mathbf{y}_j$ on the surface Γ after all interactions have been calculated. This situation is depict in Figure 12.9.

The idea behind the Fast Multipole Method is to replace (approximate) the kernel function $k(\mathbf{x}-\mathbf{y})$ for all well separated point $\mathbf{x}$ and $\mathbf{y}$ with $|\mathbf{x}-\mathbf{y}| > \varrho$ by

$$\begin{aligned} k(\mathbf{x}-\mathbf{y}) &\approx g(\mathbf{y}-\mathbf{z}_2)\mu(\mathbf{z}_2-\mathbf{z}_1)h(\mathbf{z}_1-\mathbf{x}) \\ &\approx g(\mathbf{c}_2)\mu(\mathbf{a})h(\mathbf{c}_1)\,. \end{aligned} \tag{12.17}$$

Such a factorization of the kernel can be obtained using the truncation of the series expansion of the fundamental solution of the Helmholtz equation. The use of a kernel Equation (12.17) instead of the original kernel Equation (12.16) leads to an approximate factorization of the matrix $\boldsymbol{A}$ in Equation (12.13) to Equation (12.15) such that

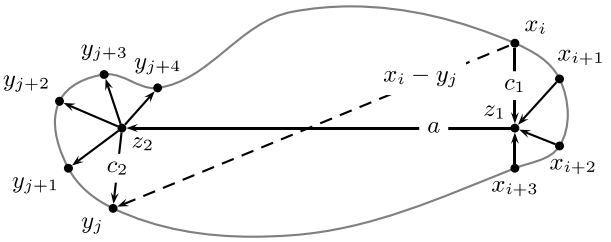

Fig. 12.9 Splitting of the $\mathbf{y}_j - \mathbf{x}_i$ path such that $\mathbf{y}_j - \mathbf{x}_i = (\mathbf{y}_j - \mathbf{z}_2) + (\mathbf{z}_2 - \mathbf{z}_1) + (\mathbf{z}_1 - \mathbf{x}_i)$.

$\boldsymbol{A} \approx (\boldsymbol{I} - \boldsymbol{A}_{\text{near}} - \boldsymbol{V}\boldsymbol{B}\boldsymbol{W})$ where $\boldsymbol{A}_{\text{near}}$, $\boldsymbol{V}$, $\boldsymbol{B}$ and $\boldsymbol{W}$ are sparse matrices. In the following the application of the Fast Multipole Method to Equation (12.14) will be considered. Its application to the Galerkin–Method can be found in [25].

12.3.2 Series Expansion of the Fundamental Solution

The fundamental solution $G(\mathbf{x}, \mathbf{y})$

$$G(\mathbf{x}, \mathbf{y}) = \frac{e^{\mathrm{i}k|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|} = \frac{e^{\mathrm{i}kr}}{4\pi r}, \quad \mathbf{r} = |\mathbf{x}-\mathbf{y}|, \quad \mathbf{x} \neq \mathbf{y} \tag{12.18}$$

of the Helmholtz equation in three dimensions can be expressed in terms of spherical Hankel and Bessel functions, Legendre polynomials and spherical harmonics. For a detailed description we refer to [1].

The derivation of the desired formula Equation (12.17) is mainly based on the truncation of the series expansion of the fundamental solution Equation (12.18). For $|\mathbf{a}| > |\mathbf{c}|$ the fundamental solution can be written as, see [1, 10.1.45],

$$\frac{e^{\mathrm{i}k|\mathbf{a}+\mathbf{c}|}}{4\pi|\mathbf{a}+\mathbf{c}|} = \frac{\mathrm{i}k}{4\pi} \sum_{l=0}^{\infty} (2l+1)(-1)^l h_l^{(1)}(k|\mathbf{a}|) j_l(k|\mathbf{c}|) P_l(\hat{a}\cdot\hat{c}) \tag{12.19}$$

with the spherical Hankel function $h_l^{(1)}$, the spherical Bessel function j_l and the Legendre polynomial P_l of order l. The notation $\hat{x} = \mathbf{x}/|\mathbf{x}|$, $|\mathbf{x}| \neq 0$ has been used for vectors on the unit sphere. Using the partial wave expansion of the plane wave (see [1, 10.1.47])

$$e^{\mathrm{i}k\hat{x}\cdot\mathbf{z}} = \sum_{l=0}^{\infty} \mathrm{i}^l (2l+1) j_l(k|\mathbf{z}|) P_l(\hat{x}\cdot\hat{z}) \tag{12.20}$$

and making use of the orthogonality of the Legendre polynomials yields the following identity

$$\begin{aligned} \int_{\mathbb{S}^2} e^{\mathrm{i}k\mathbf{y}\cdot\hat{s}} P_m(\hat{s}\cdot\hat{x})\, do(\hat{s}) &= \sum_{l=0}^{\infty} \mathrm{i}^l (2l+1) j_l(k|\mathbf{y}|) \int_{\mathbb{S}^2} P_m(\hat{y}\cdot\hat{s}) P_l(\hat{s}\cdot\hat{x})\, do(\hat{s}) \\ &= 4\pi \mathrm{i}^m j_m(k|\mathbf{y}|) P_m(\hat{x}\cdot\hat{y})\,. \end{aligned} \tag{12.21}$$

Equation (12.21) enables to replace the function $j_l(k|\mathbf{c}|)P_l(\hat{a}\cdot\hat{c})$ in Equation (12.19) which depends on the product of $\hat{a}\cdot\hat{c}$ by a product of two functions each depending on $\hat{a}$ or $\hat{c}$. Combining Equations (12.21) and (12.19) results in

$$\frac{e^{\mathrm{i}k|\mathbf{a}+\mathbf{c}|}}{4\pi|\mathbf{a}+\mathbf{c}|} = \frac{\mathrm{i}k}{4\pi}\sum_{l=0}^{\infty}(2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{a}|)\frac{1}{4\pi}\int_{\mathbb{S}^2} e^{\mathrm{i}k\mathbf{c}\cdot\hat{s}}P_l(\hat{s}\cdot\hat{a})\,do(\hat{s})\,. \tag{12.22}$$

By setting $\mathbf{a} = \mathbf{z}_1 - \mathbf{z}_2$ and $\mathbf{c} = (\mathbf{z}_2 - \mathbf{y}) + (\mathbf{x} - \mathbf{z}_1)$ the fundamental solution Equation (12.18) can be expressed as follows

$$\begin{aligned}\frac{e^{\mathrm{i}k|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|} &= \frac{\mathrm{i}k}{4\pi}\sum_{l=0}^{\infty}(2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{z}_2-\mathbf{z}_1|)\\ &\quad\times\frac{1}{4\pi}\int_{\mathbb{S}^2} e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_2)\cdot\hat{s}}e^{\mathrm{i}k\,(\mathbf{z}_1-\mathbf{x})\cdot\hat{s}}P_l(\hat{s}\cdot\widehat{(z_2-z_1)})\,do(\hat{s})\end{aligned} \tag{12.23}$$

for all point $\mathbf{x}$ and $\mathbf{y}$ satisfying the admissibility condition

$$|(\mathbf{z}_2-\mathbf{y})+(\mathbf{x}-\mathbf{z}_1)| < |\mathbf{z}_2-\mathbf{z}_1|\,. \tag{12.24}$$

In the same manner the series expansion for the gradient of the fundamental solution

$$\begin{aligned}\nabla_{\mathbf{x}}G(\mathbf{x},\mathbf{y}) &= \frac{\mathrm{i}k}{4\pi}\sum_{l=0}^{\infty}(2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{z}_2-\mathbf{z}_1|)\\ &\quad\times\frac{1}{4\pi}\int_{\mathbb{S}^2} e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_2)\cdot\hat{s}}(-\mathrm{i}k\hat{s})e^{\mathrm{i}k\,(\mathbf{z}_1-\mathbf{x})\cdot\hat{s}}P_l(\hat{s}\cdot\widehat{(z_2-z_1)})\,do(\hat{s})\end{aligned} \tag{12.25}$$

is obtained. Thus, for all pairs $(\mathbf{x},\mathbf{y})$ with $|(\mathbf{z}_2-\mathbf{y})+(\mathbf{x}-\mathbf{z}_1)| < |\mathbf{z}_1-\mathbf{z}_2|$ the kernel function $k(\mathbf{x}-\mathbf{y})$ can be replaced by

$$\begin{aligned}k(\mathbf{x}-\mathbf{y}) &= \alpha G(\mathbf{x},\mathbf{y}) + \beta\nabla_{\mathbf{x}}G(\mathbf{x},\mathbf{y})\cdot\boldsymbol{\nu}_x =\\ &= \frac{\mathrm{i}k}{4\pi}\sum_{l=0}^{\infty}(2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{z}_2-\mathbf{z}_1|)\frac{1}{4\pi}\int_{\mathbb{S}^2} e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_2)\cdot\hat{s}}(\alpha\mathrm{i}k+\beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}})\\ &\qquad\times P_l(\hat{s}\cdot\widehat{(z_2-z_1)})e^{\mathrm{i}k\,(\mathbf{z}_1-\mathbf{x})\cdot\hat{s}}\,do(\hat{s})\end{aligned} \tag{12.26}$$

This rather technical representation of the kernel function is not yet in the desired form of Equation (12.17) as the infinite summation and the integration can not be interchanged. But after truncation of the infinite summation Equation (12.26) becomes

$$\begin{aligned}k(\mathbf{x}-\mathbf{y}) &= \alpha G(\mathbf{x},\mathbf{y}) + \beta\nabla_{\mathbf{x}}G(\mathbf{x},\mathbf{y})\cdot\boldsymbol{\nu}_{\mathbf{x}}\\ &\approx \int_{\mathbb{S}^2}\underbrace{\frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_2)\cdot\hat{s}}}{4\pi}}_{g(\mathbf{y}-\mathbf{z}_2,\hat{s})}\underbrace{\mu^M(\mathbf{z}_2-\mathbf{z}_1,\hat{s})}_{\mu^M(\mathbf{z}_2-\mathbf{z}_1,\hat{s})}\underbrace{(\alpha\mathrm{i}k+\beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}})e^{\mathrm{i}k\,(\mathbf{z}_1-\mathbf{x})\cdot\hat{s}}}_{h(\mathbf{z}_1-\mathbf{x},\hat{s})}\,do(\hat{s})\\ &\approx \int_{\mathbb{S}^2} g(\mathbf{y}-\mathbf{z}_2,\hat{s})\mu^M(\mathbf{z}_2-\mathbf{z}_1,\hat{s})h(\mathbf{z}_1-\mathbf{x},\hat{s})\,do(\hat{s})\end{aligned}$$

with

$$\mu^M(\mathbf{z}_2 - \mathbf{z}_1, \hat{s}) \;=\; \frac{1}{4\pi}\sum_{l=0}^{M}(2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{z}_2 - \mathbf{z}_1|)P_l(\hat{s}\cdot\widehat{(z_2 - z_1)})\,. \quad (12.27)$$

To apply this expansion to the fast evaluation of Equation (12.14) for all points $\mathbf{y}_j$ on Γ, the surface Γ has to be splitted according to the admissibility condition Equation (12.24). Therefore, the elements of the surface triangulation $\mathcal{T}$ are grouped to clusters τ_j with the radius ρ_j and the center $\mathbf{z}_j$. Thereafter, the far and near field of a cluster τ_j are defined as follows

$$\mathcal{F}(\tau_j) := \{\tau_i |\, \rho_i + \rho_j < \eta|\mathbf{z}_j - \mathbf{z}_i|\} \quad \text{far field of } \tau_j$$

$$\mathcal{N}(\tau_j) := \{\tau_i |\, \tau_i \notin \mathcal{F}(\tau_j)\} \quad \text{near field of } \tau_j\,.$$

The parameter $\eta \in (0,1)$ defines the number of buffered clusters. The series expansion of the kernel $k(\mathbf{x} - \mathbf{y}_j)$ for a point $\mathbf{y}_j$ with $\mathbf{y}_j \in \tau_j$ can now be used for all $\mathbf{x} \in \tau_i$ with $\tau_i \in \mathcal{F}(\tau_j)$. The interaction of the cluster τ_j with the remaining clusters $\mathcal{N}(\tau_j)$ – the near field of τ_j – has to be calculated directly using standard boundary element techniques.

Using the splitting of the elements of the surface triangulation with respect to $\mathbf{y}_j \in \tau_j$ Equation (12.14) becomes

$$\boldsymbol{z}_j = u_j - \underbrace{\int_{\Gamma\cap\mathcal{N}(\tau_j)} k(\mathbf{x} - \mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma}_{=:\,\boldsymbol{A}_{\mathrm{near}<j,:>}}\cdot\boldsymbol{u} - \underbrace{\int_{\Gamma\cap\mathcal{F}(\tau_j)} k(\mathbf{x} - \mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma}_{=:\,\boldsymbol{A}_{\mathrm{far}<j,:>}}\cdot\boldsymbol{u}$$

The second term forms the j–th row of the sparse matrix $\boldsymbol{A}_{\mathrm{near}}$. The contribution of all clusters τ_i belonging to the far field of τ_j can be approximated using Equation (12.26)

$$\begin{aligned}
&\int_{\Gamma\cap\mathcal{F}(\tau_j)} k(\mathbf{x} - \mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma\cdot\boldsymbol{u} = \\
&\qquad = \sum_{\tau_i\in\mathcal{F}(\tau_j)}\int_{\tau_i} k(\mathbf{x} - \mathbf{y}_j)\varphi(\mathbf{x})\,do(\mathbf{x})\cdot\boldsymbol{u} \\
&\qquad \approx \sum_{\tau_i\in\mathcal{F}(\tau_j)}\int_{\tau_i}\int_{\mathbb{S}^2}\frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi}\mu^M(\mathbf{z}_j - \mathbf{z}_i, \hat{s})(\alpha\mathrm{i}k + \beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}}) \\
&\qquad\qquad\qquad \times e^{\mathrm{i}k\,(\mathbf{z}_i-\mathbf{x})\cdot\hat{s}}\,do(\hat{s})\varphi(\mathbf{x})\,do(\mathbf{x})\cdot\boldsymbol{u}\,.
\end{aligned}$$

Changing the order of integration and summation yields

$$\int_{\Gamma\cap\mathcal{F}(\tau_j)} k(\mathbf{x}-\mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma\cdot\boldsymbol{u} =$$

$$\approx \int_{\mathbb{S}^2} \frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi} \sum_{\tau_i\in\mathcal{F}(\tau_j)} \mu^M(\mathbf{z}_j-\mathbf{z}_i,\hat{s})$$

$$\times \underbrace{\int_{\tau_i} (\alpha\mathrm{i}k+\beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}})e^{\mathrm{i}k\,(\mathbf{z}_i-\mathbf{x})\cdot\hat{s}}\varphi(\mathbf{x})\,do(\mathbf{x})\cdot\boldsymbol{u}}_{=:\,\Psi^{\tau_i}(\mathbf{z}_i,\hat{s})}\,do(\hat{s})\,.$$

The function $\Psi^{\tau_i}(\mathbf{z}_i,\hat{s})$ is often referred to as far field pattern of the cluster τ_i. Using this notation gives

$$\int_{\Gamma\cap\mathcal{F}(\tau_j)} k(\mathbf{x}-\mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma\cdot\boldsymbol{u} \approx \int_{\mathbb{S}^2} \frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi}$$

$$\times \underbrace{\sum_{\tau_i\in\mathcal{F}(\tau_j)} \mu^M(\mathbf{z}_j-\mathbf{z}_i,\hat{s})\Psi^{\tau_i}(\mathbf{z}_i,\hat{s})}_{=:\,\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})}\,do(\hat{s})\,.$$

All far field pattern $\Psi^{\tau_i}(\mathbf{z}_i,\hat{s})$ with $\tau_i\in\mathcal{F}(\tau_j)$ have been converted to the near field pattern $\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})$ of the cluster τ_j. Thus, the contribution of the far field of the cluster τ_j to the j–th row of the matrix–vector–product can be approximated via

$$\int_{\Gamma\cap\mathcal{F}(\tau_j)} k(\mathbf{x}-\mathbf{y}_j)\varphi(\mathbf{x})\,d\Gamma\cdot\boldsymbol{p} \;\approx\; \int_{\mathbb{S}^2} \frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi}\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})\,do(\hat{s}) \qquad (12.28)$$

using the near field pattern $\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})$ of the cluster τ_j. Finally Equation (12.14), j–th row of $\boldsymbol{z}=(\boldsymbol{I}-\boldsymbol{A})\boldsymbol{u}$, becomes

$$z_j \approx u_j - \boldsymbol{A}_{\mathrm{near}\,j}\cdot\boldsymbol{u} - \int_{\mathbb{S}^2} \frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi}\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})\,do(\hat{s})\,. \qquad (12.29)$$

To use Equation (12.29) to evaluate a matrix–vector product in a first step the far field pattern $\Psi^{\tau_i}(\mathbf{z}_i,\hat{s})$

$$\Psi^{\tau_i}(\mathbf{z}_i,\hat{s}) = \int_{\tau_i} (\alpha\mathrm{i}k+\beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}})e^{\mathrm{i}k\,(\mathbf{z}_i-\mathbf{x})\cdot\hat{s}}\varphi(\mathbf{x})\,do(\mathbf{x})\cdot\boldsymbol{p} \qquad (12.30)$$

$$= \sum_{i=1}^{N} p_i \int_{\tau_i\cap\mathrm{supp}\,\varphi_i} (\alpha\mathrm{i}k+\beta k^2\hat{s}\cdot\boldsymbol{\nu}_{\mathbf{x}})e^{\mathrm{i}k\,(\mathbf{z}_i-\mathbf{x})\cdot\hat{s}}\varphi_i(\mathbf{x})\,do(\mathbf{x})$$

must be calculated for all $\tau\in\mathcal{T}$. This corresponds to the evaluation of a surface integral over a smooth function. Hence, standard Gaussian quadrature can be applied. In a second step the near field pattern $\Upsilon^{\tau_j}(\mathbf{z}_j,\hat{s})$ of a cluster τ_j

$$\Upsilon^{\tau_j}(\mathbf{z}_j, \hat{s}) = \sum_{\tau_i \in \mathcal{F}(\tau_j)} \mu^M(\mathbf{z}_j - \mathbf{z}_i, \hat{s}) \Psi^{\tau_i}(\mathbf{z}_i, \hat{s}) \tag{12.31}$$

is the translation of the previously calculated far field pattern of all η–admissible clusters of τ_j. Therefore, the operator $\mu^M(\mathbf{z}_j - \mathbf{z}_i, \hat{s})$ is referred to as translation operator. In a third step the sound pressure at the point $\mathbf{y}_j$ on the surface Γ is the results of the integration over the unit sphere

$$p(\mathbf{y}_j) = \int_{\mathbb{S}^2} \frac{e^{\mathrm{i}k\,(\mathbf{y}-\mathbf{z}_j)\cdot\hat{s}}}{4\pi} \Upsilon^{\tau_j}(\mathbf{z}_j, \hat{s})\, do(\hat{s})\,. \tag{12.32}$$

For a numerical implementation of the above equations a suitable truncation of the infinite sum in Equation (12.23) and numerical integration over the unit sphere as well as the error introduced by doing so requires some attentions.

12.3.3 Truncation of the Series Expansion and Integration on the Unit Sphere

For the numerical implementation of the Fast Multipole Method the infinite sum over l in Equation (12.23) has to be truncated for a certain integer M

$$\mu(\mathbf{a}, \hat{s}) \approx \mu^M(\mathbf{a}, \hat{s}) = \frac{1}{4\pi} \sum_{l=0}^{M} (2l+1)\mathrm{i}^l h_l^{(1)}(k|\mathbf{a}|) P_l(\hat{s}\cdot\hat{a})\,. \tag{12.33}$$

One would expect that for increasing M the Fast Multipole Method gives greater accuracy. This holds true only in exact arithmetic as the sum over l diverges as M tends to infinity. In exact arithmetic the growth of $h_l^{(1)}(k|\mathbf{a}|)$ is implicitly balanced by the decay of $j_l(k|\mathbf{c}|)P_l(\hat{c}\cdot\hat{a})$ but which has been replaced by

$$j_l(k|\mathbf{c}|)P_l(\hat{c}\cdot\hat{a}) = \frac{1}{4\pi\mathrm{i}^l} \int_{\mathbb{S}^2} e^{\mathrm{i}k\mathbf{c}\cdot\hat{s}} P_l(\hat{s}\cdot\hat{a})\, do(\hat{s})\,.$$

As the integration over the unit sphere has to be undertaken numerically

$$\int_{\mathbb{S}^2} f(\hat{s})\, do(\hat{s}) \approx \sum_{i=1}^{K_\mathrm{I}} w_i f(\hat{\xi}_i)$$

using a quadrature rule with the nodes $\hat{\xi}_i$ and the weights w_i ($i = 1, \ldots, K_\mathrm{I}$) the values of $j_l(k|\mathbf{c}|)P_l(\hat{c}\cdot\hat{a})$ appear only to a finite precision and the index of truncation M has to be chosen according to the accuracy of the numerical integration. As the value of K_I determines the efficiency of the algorithm the smallest possible value should be used. A detailed analysis of the truncation error was presented by Darve [22]. The author shows (Proposition 1 and 2) that for $\eta \leq 2/\sqrt{5}$ there exist constants C_1, C_2, C_3 and D_1, D_2, D_3 such that if $M \geq C_1 + C_2 k|\mathbf{c}| + C_2 \log(k|\mathbf{c}|) + C_3 \log \varepsilon^{-1}$ terms are kept in the sum in Equation (12.33) and the quadrature rule integrates exactly the first $K \geq D_1 + D_2 k|\mathbf{c}| + D_2 \log(k|\mathbf{c}|) + D_3 \log \varepsilon^{-1}$ spherical harmonics on $\mathbb{S}^2$,

then the error introduced by truncation and numerical integration when replacing the fundamental solution by the truncated series expansion Equation (12.22) is bounded by ε. Darve further states "*Practically a good choice for K is $K \geq 2M$*". Out of the numerical tests [20, 34, 62] the authors empirically gave formulas for the value of M of the following type

$$M = k|\mathbf{c}| + \frac{d}{1.6}\log(k|\mathbf{c}| + \pi) \tag{12.34}$$

where the value of d controls the number of significant digits in the approximation. For numerical implementation the value of K is often fixed to $K = 2M$. Consequently, the quadrature rule has to be exact up to the first K spherical harmonics. Using a Gaussian quadrature rule $\widehat{Q}_M$ which integrates exactly the first $2M$ spherical polynomials – see [64, Theorem 2.7-1] for details – $K_{\mathrm{I}} = (M+1)(2M+1)$ sample points of each far field and near field pattern as well for each translation operator need to be stored.

12.3.4 Aspects of the Numerical Implementation

To evaluate

$$z_j = (\boldsymbol{I} - \boldsymbol{A})_j \cdot \boldsymbol{u} = u_j - \sum_{i=1}^{N} u_i \int_{\Gamma \cap \mathrm{supp}\varphi_i} k(\mathbf{x} - \mathbf{y}_j)\varphi_i(\mathbf{x})\, d\Gamma \tag{12.35}$$

at each collocation point $\mathbf{y}_j, j = 1, \ldots, N$ with less than $\mathcal{O}(N^2)$ arithmetic complexity using the Fast Multipole Method the elements of the surface triangulation must be grouped into clusters. It is assumed that each element Δ_i of the surface triangulation $\mathcal{T}$ can be uniquely assigned to a cluster τ_i with the radius ρ_i and the center $\mathbf{z}_i$. The maximal cluster radius is defined as

$$\rho := \max_{i=1,\ldots,N_{\mathrm{C}}} \rho_i$$

with the number of clusters N_{C}. Thereafter the order of expansion of the fundamental solution M in Equation (12.33) is set to

$$M = \max(M_0, 2k\rho + \frac{d}{1.6}\log(2k\rho + \pi))\,. \tag{12.36}$$

The constant M_0 ensures that a sufficient number of coefficients is kept even at small wavenumbers k. Commonly $M_0 = 4$ is a good choice. Further details on the treatment of the low frequency range using the Fast Multipole Method we refer to [23, 24, 30, 40, 69]. Integration on the unit sphere is performed using a quadrature rule $\widehat{Q}_M$ that integrates exactly the first $2M$ spherical polynomials. The partition of the surface Γ is based on Equation (12.24). Two clusters τ_i and τ_j are called η–admissible with respect to the parameter $\eta \in (0, 1)$ if

$$\rho_j + \rho_i < \eta|\mathbf{z}_i - \mathbf{z}_j| \tag{12.37}$$

holds true, i.e. the sum of their radii is smaller than the distance between their cluster centers. Thereafter, the interaction of all η–admissible clusters can be calculated using Equation (12.26) and for the collocation point $\mathbf{y}_j \in \tau_j$ Equation (12.35) becomes

$$\begin{aligned} z_j &= (\boldsymbol{I} - \boldsymbol{A})_j \cdot \boldsymbol{u} \\ &\approx u_j - \sum_{q=1}^{N} u_q \underbrace{\sum_{\tau_i \in \mathcal{N}(\tau_j)} \int_{\tau_i \cap \operatorname{supp}\varphi_q} k(\mathbf{x} - \mathbf{y}_j)\varphi_q \, do(\mathbf{x})}_{=: \boldsymbol{A}_{\mathrm{near}\, j,q}} \\ &\quad - \sum_{\alpha} \underbrace{\frac{1}{4\pi} w_\alpha e^{\mathrm{i}k(\mathbf{y}_j - \mathbf{z}_j)\cdot\hat{\xi}_\alpha}}_{=: \boldsymbol{V}_{j,j_\alpha}} \sum_{\tau_i \in \mathcal{F}(\tau_j)} \underbrace{\mu^M(\mathbf{z}_j - \mathbf{z}_i, \hat{\xi}_\alpha)}_{=: \boldsymbol{B}_{j_\alpha, i_\alpha}} \Psi^{\tau_i}(\hat{\xi}_\alpha) \end{aligned} \tag{12.38}$$

with

$$\Psi^{\tau_i}(\hat{\xi}_\alpha) = \sum_{q=1}^{N} u_q \underbrace{\int_{\operatorname{supp}\varphi_q \cap \tau_i} \frac{1}{4\pi} e^{\mathrm{i}k(\mathbf{z}_i - \mathbf{x})\cdot\hat{\xi}_\alpha} \left(\alpha \mathrm{i}k + \beta k^2 \hat{\xi}_\alpha \cdot \boldsymbol{\nu}_{\mathbf{x}}\right) \varphi_q(\mathbf{x}) \, do(\mathbf{x})}_{=: \boldsymbol{W}_{i_\alpha, q}}$$

Using the notations introduced above the matrix–vector–product now reads as

$$\boldsymbol{z} = (\boldsymbol{I} - \boldsymbol{A})u \approx (\boldsymbol{I} - \boldsymbol{A}_{\mathrm{near}} - \boldsymbol{V}\boldsymbol{B}\boldsymbol{W})\boldsymbol{u}$$

where $\boldsymbol{A}_{\mathrm{near}}$, $\boldsymbol{V}$, $\boldsymbol{B}$, and $\boldsymbol{W}$ are sparse matrices. The matrix $\boldsymbol{A}_{\mathrm{near}}$ contains all nearby interactions that must still be calculated using standard boundary element techniques and the original kernel function Equation (12.16). The remaining interactions are evaluated using the Fast Multipole Method.

It can be shown by counting the non–zero elements in $\boldsymbol{A}_{\mathrm{near}}$, $\boldsymbol{V}$, $\boldsymbol{B}$ and $\boldsymbol{W}$ that the effort for evaluating one matrix–vector product is $\mathcal{O}(N^{3/2})$, cf. [22]. Therefore, a further reduction of the complexity of the algorithm is highly desirable. Obviously still $\mathcal{O}(N_{\mathrm{C}}^2)$ interactions of clusters have to be calculated. However, one will find that many of the η–admissible clusters would stay η–admissible if their radii were larger. In other words the interaction of larger parts of the surface could have been calculated using the series expansion. Hence, to increase the efficiency of the Fast Multipole Method the size of the clusters has to be enlarged as long as they stay η–admissible and evaluating of their interactions must be performed when they have reached the largest possible radius. This leads to the so called Multilevel Fast Multipole Algorithm (MLFMA).

12.3.5 The Multilevel Fast Multipole Algorithm

When looking at the Fast Multipole Method described in Section 12.3.1 it turns out that there are pairs of clusters τ_i and τ_j which would satisfy the admissible condition

Equation (12.37) even with larger radii ρ_i and ρ_j or in other words more interactions of points on the surface could have been evaluated at once using the Fast Multipole Method. The main idea of the Multilevel Fast Multipole Algorithm is to enlarge the radii of the clusters adaptively using a hierarchy of clusters, a so–called cluster tree. The admissibility condition Equation (12.37) is then applied at every level l of the tree. Only these clusters interact on level l which are η–admissible and their fathers on the next higher level are not η–admissible. The interaction of the remaining clusters will be calculated at a higher level. The shift of the far and near field pattern Ψ^τ and Υ^τ through the different levels of the cluster tree is undertaken using the following relation

$$\Psi^{\tau_1}_{\mathbf{z}_2}(\hat{z}) = \Psi^{\tau_1}_{\mathbf{z}_1}(\hat{z}) e^{\mathrm{i}k(\mathbf{z}_2-\mathbf{z}_1)\cdot\hat{z}} \tag{12.39}$$

$$\Upsilon^{\tau_1}_{\mathbf{z}_2}(\hat{z}) = \Upsilon^{\tau_1}_{\mathbf{z}_1}(\hat{z}) e^{\mathrm{i}k(\mathbf{z}_2-\mathbf{z}_1)\cdot\hat{z}}\,, \tag{12.40}$$

following directly from Equation (12.23). Analyzing the complexity of the multilevel version of the Fast Multipole Method shows that the application of the Fast Multipole Method on $\mathcal{O}(\log N)$ levels of a cluster tree leads to an algorithm of $\mathcal{O}(N \log^2 N)$ complexity [18, 22, 62]. Two important differences of the multilevel and the single–level algorithm must be addressed. Firstly, the splitting of the elements of the surface triangulation must be replaced by a hierarchy of such splittings based on a cluster tree. Secondly, as cluster radii differ on different levels the expansion length M in Equation (12.36) must be adapted and in addition to the shift of the near and far field pattern interpolation and filtering of the data will be necessary when passing information between different levels. Using the notation $\boldsymbol{B}^l$ for the translation operator on level l, $\boldsymbol{I}_l^{l+1}$ for the shift and interpolation operator from level l to level $l+1$ and $\boldsymbol{F}_l^{l+1}$ for the shift and filter operator from level $l+1$ to level l the approximation of the product of a vector $\boldsymbol{u}$ with the system matrix $\boldsymbol{A}$ can be written in the following form

$$\boldsymbol{A}\boldsymbol{u} \approx \boldsymbol{A}_{\mathrm{near}}\boldsymbol{u} + \boldsymbol{V}(\boldsymbol{F}_0^1(\boldsymbol{F}_1^2(\boldsymbol{F}_2^3(\ldots)\boldsymbol{I}_2^3 + \boldsymbol{B}^2)\boldsymbol{I}_1^2 + \boldsymbol{B}^1)\boldsymbol{I}_0^1 + \boldsymbol{B}^0)\boldsymbol{W}\boldsymbol{u}\,.$$

Similar to the single–level version first the far field pattern of the vector $\boldsymbol{u}$ is evaluated. But in the multilevel version the translation operator $\boldsymbol{B}^l$ translates far field pattern to the near field pattern on level l only for admissible clusters on level l whose fathers on level $l+1$ are not admissible. Then the far field pattern from level l is shifted to level $l+1$. Once the highest level of the cluster tree is reached the near field pattern from level l is shifted to level $l-1$ and accumulated to the pattern of level $l-1$. When the lowest level of the cluster tree is reached the accumulated near field pattern is converted to a sound pressure on the surface Γ.

12.3.6 Construction of the Cluster Tree

A cluster tree is a hierarchy of sets containing the elements of the surface triangulation. Starting on the coarsest level where the elements of the surface triangulation

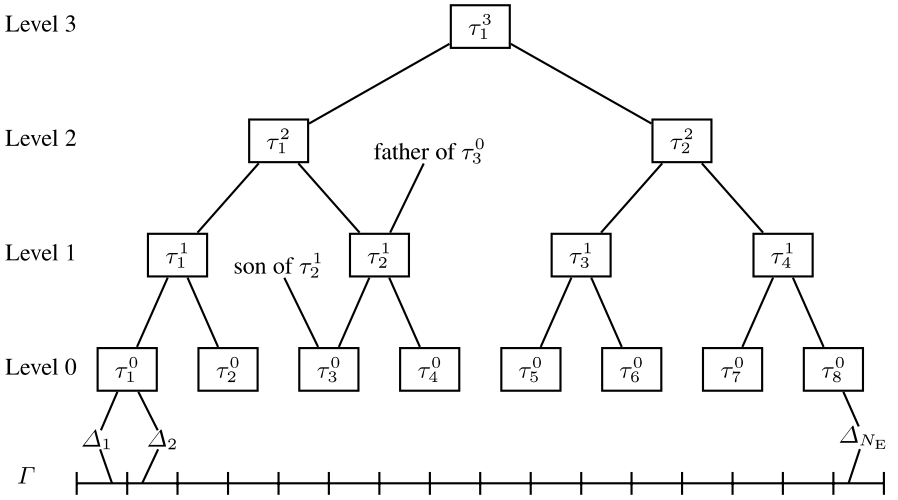

Fig. 12.10 Binary cluster tree based on the elements Δ_i on the triangulation $\mathcal{T}$ of the boundary Γ.

form a single set, sets of clusters on the next finer level are obtained by a successive subdivision of the previous sets. The newly obtained sets are referred to as sons and will become the father sets for the corresponding sets of the next finer level. As each set is subdivided into two subsets, see Figure 12.10, a binary cluster tree is obtained [29]. The process is stopped when the number of elements in a set is smaller than a given value. A different strategy where the subdivision is obtained by a successive subdivision of a cube in $\mathbb{R}^3$ yielding an oct–tree [22, 55] will not be considered here.

Once the cluster tree is build up, for each of its levels the radius ρ and the center z of a cluster is defined as the radius and center, respectively, of the smallest open ball $B(\rho, \mathbf{z})$ containing τ entirely. Further, the clusters have to satisfy $\tau_i \cap \tau_j = \emptyset$ if $i \neq j$ and $\Gamma = \cup_i \tau_i$. It is pointed out that the elements of $\mathcal{T}$ within a cluster do not necessarily have to be adjacent elements.

12.3.7 Interpolation and Filtering on the Sphere

The proper handling of the shifting of near and far field patterns through various levels of the cluster tree is of crucial importance for the efficiency of the multilevel fast multipole algorithm. As the radius of the clusters changes from level to level of the cluster tree, the number of coefficients to be kept according to Equation (12.34) is also different for each level of the cluster tree. To obtain the complexity estimates of the algorithm given above the number of coefficients M in Equation (12.33) must

be adapted according to the radius of the clusters on each level. Hence, on lower levels fewer coefficients are needed than on higher levels. Consequently, the number of sample points on the unit sphere needed for numerical integration also changes. Thus, the values at the new sample points needed when moving up and down the tree must be interpolated of the values from the previous level. As a consequence of [22, Proposition 1 and 2] this operation must not influence the number of spherical harmonics, which are required to have, so that the interpolation error is in the same order as the truncation error. Hence, interpolation must not introduce higher order harmonics. This can be guaranteed by using spherical harmonic analysis when interpolating sample points.

The spherical harmonic analysis consists of expanding a function f given at a set of points (φ_i, θ_j), $i = 0, \ldots, N$ and $j = 0, \ldots, 2N$ on the unit sphere in terms of spherical harmonics Y_n^m, that is calculating a set of coefficients α_{nm} such that

$$f(\varphi_i, \theta_j) = \sum_{n=0}^{N} \sum_{m=-n}^{n} \alpha_{nm} Y_n^m(\varphi_i, \theta_j) \tag{12.41}$$

holds true. The required function values at a new set of points $(\tilde{\varphi}_i, \tilde{\theta}_j)$, $i = 0, \ldots, \tilde{N}$ and $j = 0, \ldots, 2\tilde{N}$ are now obtained using the above expansion

$$f(\tilde{\varphi}_i, \tilde{\theta}_j) = \sum_{n=0}^{\min(N,\tilde{N})} \sum_{m=-n}^{n} \alpha_{nm} Y_n^m(\tilde{\varphi}_i, \tilde{\theta}_j)\,. \tag{12.42}$$

The values of N and $\tilde{N}$ in Equation (12.42) determine interpolation or filtering. In the case where $\tilde{N}$ is larger than N Equation (12.42) is referred to as interpolation otherwise it is referred to as filtering. Using the orthogonality properties of the spherical polynomials the coefficients α_{nm} are given by

$$\alpha_{nm} = \int_{\mathbb{S}^2} f(\varphi, \theta) \overline{Y}_n^m(\varphi, \theta)\, do(\hat{s}) = \int_{-1}^{1} \int_{0}^{2\pi} f(\varphi, t) \overline{Y}_n^m(\varphi, t)\, dt\, d\varphi \tag{12.43}$$

with the substitution $t = \cos(\theta)$. The numerical implementation of the interpolation and filtering consist of several steps. First, the integration over the φ–direction can be seen as a Fourier–transformation

$$\beta_{im} = \frac{2\pi}{2M+1} \sum_{j=0}^{2M} f(\varphi_j, \theta_i) e^{-\mathrm{i}\frac{2\pi}{2M+1} jm}\,.$$

Integration with respect to t in Equation (12.43) and summation over n in Equation (12.42) yield the new Fourier coefficients $\tilde{\beta}_{im}$

$$\tilde{\beta}_{im} = \sum_{n=0}^{M'} C_n^m C_n^m \sum_{j=0}^{M} w_j P_n^{|m|}(t_j) \beta_{jm} P_n^{|m|}(\tilde{t}_i)\,.$$

An algorithm for large M, $M^{'}$ that is more efficient than the direct evaluation of the double summation above was presented in [71]. It is base on the Christoffel–Darboux formula [1] and the application of a one dimensional Fast Multipole Method to evaluate the arising matrix–vector product in an efficient manner.

Data at the new sample points $(\tilde{\varphi}_i, \tilde{\theta}_j)$ is obtained by an inverse Fourier–transformation of $\beta^{'}_{im}$

$$f(\tilde{\varphi}_j, \tilde{\theta}_i) = \sum_{m=-M'}^{M^{'}} \tilde{\beta}_{im} e^{\mathrm{i}\frac{2\pi}{2M^{'}+1}jm} . \tag{12.44}$$

With a proper choice of M and $M^{'}$ the above required Fourier–transformations can be implemented efficiently using fast algorithms.

12.3.8 Implementation of the Algorithm

The application of the Multilevel Fast Multipole Algorithm for the Helmholtz equation is divided into two parts. In the first part commonly referred to as *setup*–step the following operations have to be performed

- definition of the number of elements of the surface triangulation n_{elem} contained in a leave of the cluster tree
- construction of the cluster tree and definition of number of levels $n_{\mathrm{level}} = c\log(N)$ that will be used,
- definition of expansion order on each level, $M^l = \max(M^0, 2k\rho^l + \frac{d}{1.6}\log(2k\rho^l + \pi))$,
- adjustment of M^l such that fast Fourier–transformation can be applied in the filter and interpolation step,
- definition of the parameter η and construction of an admissible–list of each cluster at each level according to Equation (12.37),
- calculation of the matrix $\boldsymbol{A}_{\mathrm{near}}$ of the interactions of non–η–admissible clusters on level zero and
- calculation of the sparse matrices $\boldsymbol{W}_{i_\alpha,q}$ and $\boldsymbol{B}_{j_\alpha,i_\alpha}$.

Details on the choice of the values for n_{elem}, n_{level} and M^l can be found in [55, 73]. Often the matrix $\boldsymbol{V}$ in Equation (12.38) is not explicitly calculated and the numerical integration over the unit sphere, see Equation (12.28) is performed directly at each matrix–vector–product.

The evaluation of a matrix–vector–product $\boldsymbol{v} = (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{u}$ using the Multilevel Fast Multipole Algorithm, often referred to as *apply*–step, is realized using the following algorithm:

Algorithm *apply*:

input: $\boldsymbol{u}$
output: $\boldsymbol{z} = (\boldsymbol{I} - \boldsymbol{A}_{\rm near})\boldsymbol{u} - \boldsymbol{A}_{\rm far}\boldsymbol{u}$

1. $\Psi = \boldsymbol{W}\boldsymbol{u}$! far field pattern of sound pressure distribution $\boldsymbol{u}$ on Γ
2. `do` $l = 0, n_{\rm level}$! up tree
3. `for` all clusters i on level l `do`
4. `for` all clusters j on level l `do`
5. `if` (i,j) η–admissible $\wedge$ $(f(i), f(j))$ `not` η–admissible
6. $\Upsilon^{\tau_i}(\Xi_{M^l}) = \mu^{M^l}(\mathbf{z}_i - \mathbf{z}_j, \Xi_{M^l})\Psi^{\tau_j}(\Xi_{M^l})$
7. `end`
8. `end`
9. $\Psi^{\tau_i}(\Xi_{M^{l+1}})$=`interpol`$(\Psi^{\tau_i}, M^l, M^{l+1})$
10. $\Psi^{\tau_{f(i)}}(\Xi_{M^{l+1}}) = \Psi^{\tau_{f(i)}}(\Xi_{M^{l+1}}) + \Psi^{\tau_i}(\Xi_{M^{l+1}})e^{{\rm i}k(\mathbf{z}_{f(i)} - \mathbf{z}_i)\cdot\Xi_{M^{l+1}}}$
11. `end`
12. `end`
13. `do` $l = n_{\rm level} - 1, 0$! down tree
14. `for` all boxes i on level l `do`
15. $\Upsilon^{\tau_i}(\Xi_{M^{l+1}}) = \Upsilon^{\tau_{f(i)}}(\Xi_{M^{l+1}})e^{{\rm i}k(\mathbf{z}_i - \mathbf{z}_{f(i)})\cdot\Xi_{M^{l+1}}}$
16. $\Upsilon^{\tau_i}(\Xi_{M^l}) = \Upsilon^{\tau_i}(\Xi_{M^l}) +$ `filter`$(\Upsilon^{\tau_i}(\Xi_{M^{l+1}}), M^{l+1}, M^l)$
17. `end`
18. `end`
19. `for` all clusters i on level 0 `do`
20. $z_{2_j} = \sum_\alpha \frac{1}{4\pi} w_\alpha e^{{\rm i}k(\mathbf{y}_j - \mathbf{z}_j)\cdot\hat{\xi}_\alpha}\Upsilon^{\tau_i}(\hat{\xi}_\alpha)$
21. `end`
22. $\boldsymbol{z}_1 = \boldsymbol{A}_{\rm near}\boldsymbol{u}$
23. $\boldsymbol{z} = \boldsymbol{u} - \boldsymbol{z}_1 - \boldsymbol{z}_2$

The functions `interpol` and `filter` transfer data given on a set of points Ξ_{M^l} to a set of points $\Xi_{M^{l+1}}$ and vice versa as discussed in Section 12.3.7.

The implementation of the algorithm above shows that a large part of the memory required for the multipole part is needed to store the data $\mu^M(\mathbf{z}_j - \mathbf{z}_i, \Xi_{M^l}) = \mu^M(\mathbf{z}, \Xi_{M^l})$ with $\mathbf{z} = [z^1, z^2, z^3]$ which is the translation operator that translates far field pattern to near field pattern. This is a drawback of the use of a binary cluster tree over an oct–tree. Taking a closer look at the structure of the data shows the dependence on the absolute values of the components[4] of z only. Hence, using a suitable permutation P the values $\mu^M([\pm|z^{1,2}|, \pm|z^{2,1}|, \pm|z^3|], \Xi_{M^l})$ can be calculated out of the values of $\mu^M([|z^1|, z^2|, |z^3|], \Xi_{M^l})$ yielding to the situation that the memory requirement is now negligible. To increase the probability that the difference of two cluster centers $\mathbf{z} = \mathbf{z}_j - \mathbf{z}_i$ differs only by the signs of the components the position of the cluster centers can be restricted to points on a regular grid $[ih, jh, kh]$. This yields a situation quite similar to that of an oct–tree obtained by a successive subdivision of a cube. If however necessary, a further compression of the data μ^M can be achieved by the recalculation of $\mu^M(\mathbf{z}, \Xi_{M^l})$ as it is needed. An efficient algorithm

[4]The components 1 and 2 of z may be interchanged.

for this task is given in [68]. There the authors use a one–dimensional Fast Multipole Method to evaluate

$$\mu^M(\mathbf{z},\hat{s}) = \sum_{i=0}^{M} \alpha_i P_i(\hat{z}\cdot\hat{s}) \tag{12.45}$$

with $\alpha_l = (2l+1)\mathrm{i}^l h_l^{(1)}(k|z|)$ in an efficient manner. The evaluation of μ^M at $\mathcal{O}(M^2)$ locations needs only $\mathcal{O}(M^2\log(1/\epsilon))$ arithmetic operations where ϵ determines the accuracy of the approximation. Thereafter, only the coefficients α_i which depend on the modulus of $\mathbf{z}$ need be stored.

12.3.9 An Example

The Multilevel Fast Multipole Algorithm described above will be used to demonstrate that the boundary element method can be applied efficiently to large scale problems.

The objective of the following numerical example is the prediction of the quality of an anechoic chamber in the low frequency range. The fact that the acoustic lining of such a chamber is not sufficiently absorbing at low frequencies creates several inconveniences. First, from a practical point of view, remaining reflections of the walls perturb experimental results. Secondly, from a numerical point of view, neither the geometry of the lining nor sound propagation within the absorbing material can be neglected. This excludes the use of the model of a rectangular cavity where walls are equipped with a local admittance condition to account for the acoustic lining. Therefore the numerical model must respect the real geometry of the lining and modelling of the absorbing material requires special attention. Here an admittance matrix – instead of a scalar value –, taking into account for sound propagation within a specific part of the lining and its vicinity, was used to represent the behaviour of the acoustical treatment. Hence, on the air lining interface the sound pressure p at a specific part of the surface is coupled with the surface velocity $v_{\boldsymbol{\nu}}$ on that part and its vicinity via

$$v_{\boldsymbol{\nu}} = \boldsymbol{Y}\boldsymbol{p} \tag{12.46}$$

with the dense and frequency dependent matrix $\boldsymbol{Y}$. The approximation, that only the surface velocity at the vicinity is considered, can been seen as a localisation of the non–local behavior of an absorbing material.

Numerical results will be compared with experimental data obtained from measurements carried out in the large anechoic chamber of the LMA, see Figure 12.11. The 1.5 dB region in the 20 to 200 Hz frequency range was obtained by measuring the sound pressure radiated by a bass–reflex box. For a detailed description of the experiment we refer to [58]. The anechoic chamber at the LMA has inner dimensions of $5.4 \times 6.3 \times 11.4$ m^3 measured from wedge tip to wedge tip. Each of the 3720 wedges consists of a rectangular parallelepiped measuring $.3 \times .3 \times .4$ m^3 forming the base of the wedge and a tapering section of .7 m in length. Wedges are made of melamine foam. To calculate the admittance matrix $\boldsymbol{Y}$ a central wedge and its 8 adjacent wedges have been used. As all of the wedges are identical only a single

Fig. 12.11 Interior of the large anechoic chmaber of the LMA (left sub–figure) and a photo of a sample of 3×3 elements of the acoustic lining (right sub–figure).

admittance matrix $\boldsymbol{Y}(\omega) \in \mathbb{C}^{324\times 36}$ is needed. However, to take into account for the anisotropy of the melamine foam three different matrices $\boldsymbol{Y}^i$, representing the three different material orientations, have actually been used. These matrices were precalculated in the frequency range of 20 to 200 Hz with a frequency resolution of 1 Hz using a finite element method. The geometry of the wedges has been modelled using 9 elements resulting in a mesh size of ≈.3 m. Using linear discontinuous basis functions [42] yields a linear system with N=138 280 unknowns and standard boundary element methods, requiring ≈ 300 Gb of memory to hold the system matrix, are hardly applicable. Therefore a six–level fast multipole method was applied. The leaves of the cluster tree contained up to $n_{\mathrm{elem}} = 4$ surface elements. The parameter η in Equation (12.37) was set to $\eta = .7$. The memory requirement of the algorithm at 200 Hz is given in the left sub–table of Table 12.2. It can be seen that the matrices of the near field interactions occupy almost all of the required memory. The matrix named "$work$" represents the working space needed for the multipole algorithm to hold the far and near field patterns on the different levels. The expansion length M in Equation (12.36) varied from 5 to 16 depending on the level. The memory required to hold the translation operator μ^{M}, matrix $\boldsymbol{B}$ in Table 12.2, is indeed negligible, as stated in Section 12.3.8, due to the performed compression. Each stored operator has be reused ≈ 1500 times. Computational resources, the time and the number of floating point operations required to perform a matrix–vector–product, are given in the right sub–table of Table 12.2. It is pointed out that due to the admittance boundary condition the product $\boldsymbol{A}_{\mathrm{near}}\boldsymbol{u}$ reads as $\boldsymbol{A}_{\mathrm{near}}\boldsymbol{u} = \boldsymbol{H}_{\mathrm{near}}\boldsymbol{u} - \boldsymbol{G}_{\mathrm{near}}\boldsymbol{Y}\boldsymbol{u}$. Therefore the near field part of the matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ must be stored seperately. Performing a

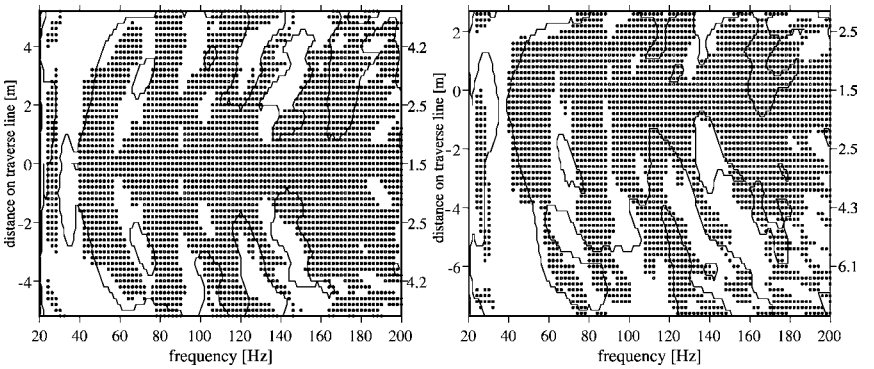

Fig. 12.12 1.5 dB regions for two source positions in the chamber obtained using numerical simulations.

single product $(\boldsymbol{I}-\boldsymbol{A})\boldsymbol{u}$ took 72.0 s on a SGI Origin3800 and required 6.4e9 floating point operations. In contrast the standard BEM would require at least $N^2 \approx$19.1e9 floating point operations. The above given computational resources have been measured using a *performance counter library*[5].

The linear system was solved using the GMRes [52] solver. Depending on frequency 80...120 iterations were necessary to obtain a residual of $\varepsilon = 10^{-6}$. The total solution time was 2.5...4 hours per frequency on a SGI Origin3800 of the Center for Information Services and High Performance Computing at the Technische Universität Dresden, Germany. Numerical results are compared with experimental data in Figure 12.12 for two different source positions. A solid dot represents the numerical result. The bounds of the 1.5 dB region obtained from experimental data are represented by a solid line. For both source positions numerical and experimental results agree well for frequencies higher than 100 Hz.

The example shows that the BEM can be applied to large scale problems. Especially when the boundary of the fluid domain has a complex shape the boundary element method is competitive to the finite element method as meshing the complex geometry of the fluid domain can be avoided.

12.4 Conclusion

Fast solution methods have been discussed to overcome the $\mathcal{O}(N^3)$ complexity of the standard boundary element method when using a direct solution of the system of linear equations. By the use of an iterative solver the complexity can be reduced to $\mathcal{O}(N^2)$, if the number of required iterations is much less than the number of unknowns N. Krylov subspace methods are the most suitable class of iterative solvers in the context here. The convergence of the iterative solvers is fairly accelerated

[5] http://www.fz-juelich.de/zam/PCL

Table 12.2 Memory requirement (left table) and computational resources needed to perform one matrix–vector product (right table) when using a six–level fast multipole algorithm. The first line, labeled "Equation (12.46)", represents the application of the boundary condition. The last column of the right table gives the performance of the algorithm with respect to the peak–performance of the computer's processor.

Matrix	Memory [Mb]
$\boldsymbol{H}_{\text{near}}$	2148
$\boldsymbol{G}_{\text{near}}$	2148
$\boldsymbol{W}$	111
$\boldsymbol{B}$	5
work	107
total	4519

	CPU–time		Flops		Flop rate
line of *apply*	[s]	[%]	[Mflops]	[%]	[%]
Eq. (12.46)	5.3	7.4	1569.5	24.3	73.5
1	0.6	0.8	34.6	0.5	14.3
22	1.3	1.8	16.2	0.3	3.2
20	1.1	1.5	214.6	3.3	48.1
6	23.5	32.7	2807.5	43.5	29.8
9–10	7.2	10.0	371.6	5.8	12.8
15–16	7.0	9.8	365.9	5.7	13.0
22	25.9	35.9	1074.1	16.6	10.4
total	72.0	100.0	6454.0	100.0	22.4

by strategies of preconditioning. Preconditioners based on the splitting of the system matrix are effective for the BEM that generates the dense system matrix. The sparse matrix corresponding to the near field interaction is factorized using an incomplete LU decomposition. The use of the hyper singular formulation leads to ill–conditioned matrices that cause slow convergence. Furthermore the properties of the boundary of the domain have a significant influence on the required number of iterations. In general it can be stated that complex shapes and low absorption will cause slow convergence.

A further reduction can be achieved through the use of fast BEMs which avoid the explicit set–up of the dense system matrix. Especially the multilevel fast multipole method seems to be the most widely accepted as such fast method. Based on the truncation of the series expansion of the fundamental solution, the fast multipole method yields an approximate factorization of the system matrix. The application of this method on multiple levels of a cluster tree enables the evaluation of a matrix–vector product with $\mathcal{O}(N \log^2 N)$ complexity. Therefore, the iterative solution of the linear system together with the multilevel fast multipole method yields a very efficient numerical method. The obtained higher efficiency of the accelerated BEM makes this numerical method applicable to real life problems.

References

1. Abramowitz M, Stegun IA (1965) Handbook of mathematical functions. Dover Publications, New York
2. Amini S (1987) An iterative method for the boundary element solution of the exterior acoustic problem. Journal of Computational and Applied Mathematics 20:109–117

3. Amini S (1999) On boundary integral operators for the Laplace and the Helmholtz equations and their discretisations. Engineering Analyis with Boundary Elements 23:327–337
4. Amini S, Chen K (1989) Conjugate gradient method for second kind integral equations–applications to the exterior acoustic problem. Engineering Analyis with Boundary Elements 6:72–77
5. Amini S, Maines ND (1998) Preconditioned Krylov subspace methods for boundary element solution of the Helmholtz equation. International Journal for Numerical Methods in Engineering 41:875–898
6. Amini S, Profit ATJ (2003) Multi–level fast multipole solution of the scattering problem. Engineering Analysis with Boundary Elements 27:547–564
7. Anderson CR (July 1992). An implementation of the fast multipole method without multipoles. SIAM Journal on Scientific and Statistical Computing 13:923–947
8. Barret R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, Van der Vorst H (1994) Templates for the solution of linear systems: building blocks for iterative methods. Society for Industrial and Applied Mathematics, Philadelphia, USA
9. Bebendorf M, Rjasanov S (2003) Adaptive low–rank approximation of collocation matrices. Computing 70:1–24
10. Benzi M (2002) Preconditioning techniques for large linear systems: A survey. Journal of Computational Physics 182:418–477
11. Bespalov A (2000) On the usage of a regular grid for implementation of boundary integral methods for wave problems. Russian Journal of Numerical Analysis and Mathematical Modelling 15:469–488
12. Beylkin G, Coifman R, Rokhlin V (1991) Fast wavelet transforms and numerical algorithms. Communications on Pure and Applied Mathematics 44:141–183
13. Burton AJ, Miller GF (1971) The application of integral equation methods to the numerical solution of some exterior boundary value problems. Proceedings of the Royal Society of London A 323:201–210
14. Chen K (1998) On a class of preconditioning methods for dense linear systems from boundary elements. SIAM Journal of Scientific Computing 20:684–698
15. Chen K (2001) An analysis of sparse approximate inverse preconditioners for boundary integral equations. SIAM Journal on Matrix Analysis and Applications 22:1058–1078
16. Chen K, Harris PJ (2001) Efficient preconditioners for iterative solution of the boundary element equations for the three–dimensional Helmholtz equation. Applied Numerical Mathematics 36:475–489
17. Chen S, Liu Y (2000) A unified boundary element method for the analysis of sound and shell–like structure interactions. II. Efficient solution techniques. Journal of the Acoustical Society of America 108:2738–2745
18. Chew WC, Jin JM, Lu CC, Michelssen E, Song JM (1997) Fast solution methods in electromagnetics. IEEE Transaction on Antennas Propagation 45:533–543
19. Cipra BA (2000) The best of the 20th century: editors name top 10 algorithms. SIAM News 33:1–2
20. Coifman R, Rokhlin V, Wandzura S (June 1993) The fast multipole method for the wave equation: a pedestrian prescription. IEEE Antennas and Propagation Magazine 35:7–12
21. Dahmen W, Kleemann B, Prössdorf S, Schneider R (1997) Multiscale methods for the solution of the Helmholtz and Laplace equations. In: Wendland W (ed) Boundary Element Methods. Reports from the Final Conference of the Priority Research Program 1989–1995 of the German Research Foundation, Stuttgart, Springer–Verlag
22. Darve E (Feb. 2001) The fast multipole method I: Error analysis and asymptotic complexity. SIAM Journal on Numerical Analysis 38:98–128

23. Darve E, Havé P (2004) Efficient fast multipole method for low–frequency scattering. Journal on Computational Physics 197:341–363
24. Darve E, Havé P (2004) A fast multipole method for maxwell equations stable at all frequencies. Philosophical Transaction of the Royal Society of London A 362:1–27
25. Fischer M, Gauger U, Gaul L (2004) A multipole Galerkin boundary element method for acoustics. Engineering Analysis with Boundary Elements 28:155–162
26. Fletcher R (1976) Conjugate gradient methods for indefinite systems. In: Watson GA (ed) Dundee Conference on Numerical Analysis. 1975, Lecture Notes in Mathematics 506:73–89, Springer–Verlag, Berlin
27. Freeden W, Schneider F (1999) Runge–Walsh–wavelet approximation for the Helmholtz equation. Journal of Mathematical Analysis and Application 235:533–566
28. Freund RW (1993) A transpose–free quasi–minimum residual algorithm for non–Hermitian linear systems. SIAM Journal on Scientific Computing 14:470–482
29. Giebermann K (2001) Multilevel approximation of boundary integral operators. Computing 67:183–207
30. Greengard L, Huang J, Rokhlin V, Wandzura S (1998) Accelerating fast multipole methods for the Helmholtz equation at low frequencies. IEEE Computational Science and Engineering 5:32–38
31. Greengard L, Rokhlin V (1987) A fast algorithm for particle simulations. Journal of Computational Physics 73:325–348
32. Gumerov N, Duraiswami R (2005) Fast multipole methods for the Helmholtz equation in three dimensions. Elsevier, Oxford
33. Gutknecht MH (1993) Variations of BiCGSTAB for matrices with complex spectrum. SIAM Journal on Scientific Computing 14:1020–1033
34. Gyure MF, Stalzer MA (1998) A prescription for the multilevel Helmholtz FMM. IEEE Computanional Science & Engineering 5:39–47
35. Hackbusch W (1999) A sparse matrix arithmetic based on H–matrices. Part I: Introduction to H–matrices. Computing 62:89–108
36. Hackbusch W, Nowak Z (1989) On the fast matrix multiplication in the boundary element method by panel clustering. Numerische Mathmatik 54:463–491
37. Harris PJ, Chen K (2003) On efficient preconditioners for iterative solution of a Galerkin boundary element equation for the three–dimensional exterior Helmholtz problem. Journal of Computation and Applied Mathematics 156:303–318
38. Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49:409–436
39. Huybrechs D, Simoens J, Vandewalle S (2004) A note on wave number dependence of wavelet matrix compression for integral equations with oscillatory kernel. Journal of Computation and Applied Mathematics 172:233–246
40. Zhao J–S, Chew WC (2000) Three-dimensional multilevel fast multipole algorithm from static to electrodynamic. Microwave and Optical Technology Letters 26:43–48
41. Lanczos C (1952) Solution of systems of linear equations by minimized iterations. Journal of Research of the National Bureau of Standards 49:33–53
42. Marburg S, Schneider S (2003) Influence of element types on numeric error for acoustic boundary elements. Journal of Computational Acoustics 11:363–386
43. Marburg S, Schneider S (2003) Performance of iterative solvers for acoustic problems. Part I. Solvers and effect of diagonal preconditioning. Engineering Analysis with Boundary Element 27:727–750
44. Ochmann M, Homm A, Makarov S, Semenov S (2003) An iterative GMRES–based boundary element solver for acoustic scattering. Engineering Analysis with Boundary Element 27:717–725

45. Okamoto N, Otsuru T, Tomiku R, Yasuda Y (2007) Numerical analysis of large–scale sound fields using iterative methods. Part II: Application of Krylov subspace methods to finite element analysis. Journal of Computational Acoustics 15, accepted for publication
46. Otsuru T, Uchinoura Y, Tomiku R, Okamoto N, Takahashi Y (2004) Basic concept, accuracy and application of large–scale finite element sound field analysis of rooms. In: Proceedings of the 18th International Congress on Acoustics (ICA), Kyoto I 479–482
47. Prasad KG, Kane JH, Keyes DE, Balakrishna C (1994) Preconditioned Krylov solvers for BEA. International Journal for Numerical Methods in Engineering 37:1651–1672
48. Rahola J (1996) Diagonal forms of the translation operators in the fast multipole algorithm for scattering problems. BIT Numerical Mathematics 36:333–358
49. Rokhlin V (Feb. 1990) Rapid solution of integral equations of scattering theory in two dimensions. Journal of Computational Physics 86:414–439
50. Rokhlin V (1993) Diagonal forms of the translation operators for the Helmholtz equation in three dimensions. Applied and Computational Harmonic Analysis 1:82–93
51. Saad Y (1994) ILUT: a dual threshold incomplete LU factorization. Numer. Linear Algebra 1:387–402
52. Saad Y, Schultz MH (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing 7:856–869
53. Saad Y, van der Vorst HA (2000) Iterative solution of linear systems in the 20th century. Journal of Computational and Applied Mathematics 123:1–33
54. Sakuma T, Takamura N, Yasuda Y, Sakamoto S (2005) Numerical analysis of the additional attenuation due to the tops of edge modified barriers. In: Proceedings of the Inter–Noise 2005, Rio de Janeiro, No. 1956 (CD)
55. Sakuma T, Yasuda Y (2002) Fast multipole boundary element method for large–scale steady–state sound field analysis, Part I: Setup and validation. Acta Acustica united with Acustica 88:513–525
56. Sauter S (2000) Variable order panel clustering. Computing 64:223–261
57. Schneider S (2003) Application of fast methods for acoustic scattering and radiation problems. Journal of Computational Acoustics 11:387–401
58. Schneider S, Kern C (2007) Acoustical behavior of the large anechoic chamber at the Laboratoire de Mécanique et d'Acoustique in the low frequency range. Acta Acustica united with Acustica, submitted for publication.
59. Schneider S, Marburg S (2003) Performance of iterative solvers for acoustic problems. Part II. Acceleration by ILU–type preconditioner. Engineering Analysis with Boundary Elements 27:751–757
60. Shen L, Liu YJ (2006) An adaptive fast multipole boundary element method for solving large–scale three–dimensional acoustic wave problems based on the Burton–Miller formulation. Computational Mechanics, accepted for publication
61. Sleijpen GLC, Fokkema DR (1993) BiCGStab(l) for linear matrices involving unsymmetric matrices with complex spectrum. Electronic Transactions on Numerical Analysis 1:11–32.
62. Song J, Lu CC, Chew WC (Okt. 1997) Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. IEEE Transactions on Antennas and Propagation 45:1488–1493
63. Sonneveld P (1989) CGS, a fast Lanczos–type solver for nonsymmetric linear systems. SIAM Journal on Scientific Statistical Computing 10:36–52
64. Stroud AH (1971) Approximate calculation of multiple integrals. Prentice–Hall, New York

65. van der Vorst HA (1992) Bi–CGSTAB: a fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific Statistical Computing 13:631–644
66. van der Vorst HA (2003) Iterative Krylov methods for large linear systems. Cambridge University Press, Cambridge
67. Vavasis SA (1992) Preconditioning for boundary integral equations. SIAM Journal on Matrix Analysis and Applications 13:905–925
68. Velamparambil S, Chew WC (2001) A fast polynomial representation for the translation operators of an MLFMA. Microwave and Optical Technology Letters 28:298–303
69. Wallén H, Sarvas J(2005) Translation procedures for broadband MLFMA. Progress in Electromagnetics Research (PIER) 55:47–78
70. Yan Y (1994) Sparse preconditioned iterative methods for dense linear systems. SIAM Journal on Scientific Computing 15:1190–1200
71. Yarvin N, Rokhlin V (1999) An improved fast multipole algorithm for potential fields on the line. SIAM Journal on Numerical Analysis 36:629–666
72. Yasuda Y, Sakamoto S, Kosaka Y, Sakuma T, Okamoto N, Oshima T (2007) Numerical analysis of large–scale sound fields using iterative methods. Part I: Application of Krylov subspace methods to boundary element analysis. Journal of Computational Acoustics 15, accepted for publication
73. Yasuda Y, Sakuma T (2003) Fast multipole boundary element method for large–scale steady–state sound field analysis. Part II: Examination of numerical items. Acta Acustica united with Acustica 89:28–38
74. Ylä–Oijala P, Järvenpää S (2006) Iterative solution of high–order boundary element method for acoustic impedance boundary value problems. Journal of Sound and Vibration 291:824–843
75. Zhang SL (1997) GPBi–CG: Generalized product–type methods based on Bi–CG for solving nonsymmetric linear systems. SIAM Journal on Scientific Computing 18:537–551