17th International Conference on the Application of Computer
Science and Mathematics in Architecture and Civil Engineering
K. Gürlebeck and C. Könke (eds.)
Weimar, Germany, 12–14 July 2006

MULTIGRID PRECONDITIONED CONJUGATE GRADIENT METHOD IN THE MECHANICAL ANALYSIS OF HETEROGENEOUS SOLIDS

S. Häfner* **and C. Könke**

**Institute of Structural Mechanics, Bauhaus-University Weimar,*
Marienstrasse 15, D-99423 Weimar, Germany
E-mail: stefan.haefner@bauing.uni-weimar.de

Keywords: multigrid method, conjugate gradient method, finite element, heterogeneous solid.

Abstract. *A fast solver method called the multigrid preconditioned conjugate gradient method is proposed for the mechanical analysis of heterogenous materials on the mesoscale. Even small samples of a heterogeneous material such as concrete show a complex geometry of different phases. These materials can be modeled by projection onto a uniform, orthogonal grid of elements. As one major problem the possible resolution of the concrete specimen is generally restricted due to (a) computation times and even more critical (b) memory demand. It is well known that this type of problem, especially for volumetric models, quickly tap today's computational potential. The possibility to increase the model size by resolution in comparison to standard procedures would be valuable. Direct solver methods require the storage of a global stiffness matrix and the corresponding memory demand limits the problem size. Iterative solvers can be based on a local element-based formulation while orthogonal grids consist of geometrical identical elements. The element-based formulation is short and transparent, and therefore efficient in implementation. A variation of the material properties in elements or integration points is possible. The multigrid method is a fast iterative solver method, where ideally the computational effort only increases linear with problem size. This is an optimal property which is almost reached in the implementation presented here. In fact no other method is known which scales better than linear. Therefore the multigrid method gains in importance the larger the problem becomes. But for heterogeneous models with very large ratios of Young's moduli the multigrid method considerably slows down by a constant factor. Such large ratios occur in certain heterogeneous solids, as well as in the damage analysis of solids. As solution to this problem the multigrid preconditioned conjugate gradient method is proposed. A benchmark highlights the multigrid preconditioned conjugate gradient method as the method of choice for very large ratio's of Young's modulus. A proposed modified multigrid cycle shows good results, in the application as stand-alone solver or as preconditioner.*

1 INTRODUCTION

The mesoscale analysis of concrete had been the primary interest of the project in which this article developed. First, there was the idea to create a realistic geometrical model of concrete on the mesoscale. Some corresponding methods were published in [7] and mechanical finite element analysis on small samples was performed. Thereby it quickly turned out that the next major problem is the computational performance. The large number of inclusions with variable shape and size requires computation of high resolution in order to reflect the mechanical behavior of the real material. In other words, many degrees of freedom are required and large equation systems have to be solved. In the complete finite element process the time of the solver is by far most relevant in comparison to all other required operations. Furthermore, the problem size is limited by the memory demand in which especially the storage of the global stiffness matrix is critical.

The present approach is dedicated to a maximum increase of model size and optimal performance in the mechanical analysis of heterogeneous solids. At the same time, other important physical and numerical aspects are out of scope. However, adapted methods for advanced finite elements are presented in [9] and for damage analysis in [11]. In combination with these approaches the present concept of grid-based modeling gains in relevance. As a main property of this concept the storage of a global stiffness matrix is superseded and replaced by element-based formulations. Several variants to model heterogeneous materials are introduced which can be adapted to such a local scheme. The methods are short and transparent in implementation. Beyond the classical and well-known formulations of the multigrid method and the multigrid preconditioned conjugate gradient method, this article provides transparent access to these methods with respect to the defined engineering application.

It is an important property of the multigrid method that the computational effort only increases linear with problem size. This is declared as an optimal property. In fact, therefore the multigrid method gains in importance the larger the problem becomes. This advantage of the multigrid method will even be strengthened with increasing computational power. However, for heterogeneous models with very large ratios of Young's moduli the multigrid method essentially slows down. Such large ratios of Young's moduli are expected for certain heterogeneous materials as well as in damage analysis [11]. A remedy is found in the multigrid preconditioned conjugate gradient method. Corresponding results are presented. Besides, a modified multigrid cycle is introduced.

2 PROBLEM STATEMENT AND NOTATION

In the static case the finite element method leads to the linear equation system

$$\boldsymbol{K}\,\boldsymbol{u} = \boldsymbol{f} \tag{2.1}$$

where $\boldsymbol{K}$ denotes the stiffness matrix, $\boldsymbol{f}$ is the vector of prescribed forces and $\boldsymbol{u}$ is the solution, the displacement vector which is initially unknown. For the mechanical background of the finite element formulation, it is referred to [9]. The global stiffness matrix $\boldsymbol{K}$ corresponds to the sum of element stiffness matrices $\bar{\boldsymbol{K}}_e$ after rearrangement according to the global degrees of freedom

$$\boldsymbol{K} = \sum_{e=1}^{n^e} \bar{\boldsymbol{K}}^e\,, \qquad K_{ij} = \sum_{e=1}^{n^e} \bar{K}^e_{ij} \tag{2.2}$$

and the global force vector corresponds to the sum of reordered element force vectors $\bar{\boldsymbol{f}}^e$.

$$\boldsymbol{f} = \sum_{e=1}^{n^e} \bar{\boldsymbol{f}}^e \,, \qquad f_i = \sum_{e=1}^{n^e} \bar{f}_i^e \tag{2.3}$$

In the case that (also) displacements are prescribed, these predefined displacements can be eliminated according to the following procedure, which again leads to the form of Eq. 2.1. Considering all degrees of freedoms at all nodes at a finite element model, then here the index 1 corresponds to degrees of freedom with a prescribed force and the index 2 corresponds to those of a prescribed displacement.

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,2} \\ \boldsymbol{K}_{2,1} & \boldsymbol{K}_{2,2} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix} \tag{2.4}$$

Then the initially unknowns of this system are $\boldsymbol{u}_1$ and $\boldsymbol{f}_2$. As the right hand side of the following equation is known, it is of the same form as Eq. 2.1 and enables analogous computation of $\boldsymbol{u}_1$

$$\boldsymbol{K}_{1,1}\,\boldsymbol{u}_1 = \boldsymbol{f}_1 - \boldsymbol{K}_{1,2}\,\boldsymbol{u}_2 \tag{2.5}$$

independent of $\boldsymbol{f}_2$. If the resulting force vector $\boldsymbol{f}_2$ at rigid degrees of freedom is also of interest, then it can straightforward be computed by

$$\boldsymbol{f}_2 = \boldsymbol{K}_{2,1}\,\boldsymbol{u}_1 + \boldsymbol{K}_{2,2}\,\boldsymbol{u}_2 \tag{2.6}$$

In the following discussion of iterative solver methods it is assumed that the problem is of the form as in Eq. 2.1 and the vector $\boldsymbol{u}$ corresponds to the displacements of the effective degrees of freedom. In the scalar variable u_i^k the superscript k denotes the k-th iteration and the subscript i indicates the i-th element of the vector. The increment of displacements $\Delta\boldsymbol{u}^k$ after the k-th iteration is

$$\Delta\boldsymbol{u}^k = \boldsymbol{u}^{k+1} - \boldsymbol{u}^k \tag{2.7}$$

and with

$$\boldsymbol{f}^k = \boldsymbol{K}\boldsymbol{u}^k \tag{2.8}$$

the increment of forces $\Delta\boldsymbol{f}^k$ is

$$\Delta\boldsymbol{f}^k = \boldsymbol{f}^{k+1} - \boldsymbol{f}^k \tag{2.9}$$

The error $\boldsymbol{e}^k$ is defined as the difference between the iterative displacement approximation $\boldsymbol{u}^k$ after the k-th iteration and the solution vector $\boldsymbol{u}$.

$$\boldsymbol{e}^k = \boldsymbol{u}^k - \boldsymbol{u} \tag{2.10}$$

The residual $\boldsymbol{r}^k$ is the difference of the prescribed force vector $\boldsymbol{f}$ and the effective right hand side $\boldsymbol{f}^k$ after the k-th iteration

$$\boldsymbol{r}^k = \boldsymbol{f} - \boldsymbol{f}^k \tag{2.11}$$

Hence, from Eqs. 2.1, 2.9, 2.10 2.11 it can be derived that

$$-\boldsymbol{K}\,\boldsymbol{e}^k = \boldsymbol{r}^k \tag{2.12}$$

3 BASIC ITERATIVE SOLVER METHODS

3.1 Stationary iterative methods

As identifying characteristic, stationary iterative methods can be expressed in the simple form

$$\boldsymbol{u}^{k+1} = \boldsymbol{Q}\,\boldsymbol{u}^{k} + \boldsymbol{q} \tag{3.1}$$

where $\boldsymbol{Q}$ and $\boldsymbol{q}$ are placeholders for a matrix and a vector, respectively, both independent of iteration step k. Among others, stationary iterative methods, such as e.g. the Gauss-Seidel method, may act as smoothers within the later proposed multigrid method. Within this engineering framework these methods are briefly introduced. Further mathematical background of these iterative methods is treated in e.g. [1], [6] and [17].

The considered problem is a linear equation system of the form $\boldsymbol{K}\,\boldsymbol{u} = \boldsymbol{f}$. Then it is trivial to introduce any regular matrix $\boldsymbol{S}$, called the splitting matrix, as follows

$$\boldsymbol{K} = \boldsymbol{S} + (\boldsymbol{K} - \boldsymbol{S}) \tag{3.2}$$

The linear equation system can be rewritten as

$$\boldsymbol{S}\boldsymbol{u} = (\boldsymbol{S} - \boldsymbol{K})\,\boldsymbol{u} + \boldsymbol{f} \tag{3.3}$$

and the iterative scheme of the splitting method is defined as

$$\boldsymbol{u}^{k+1} = \boldsymbol{S}^{-1}\left((\boldsymbol{S} - \boldsymbol{K})\,\boldsymbol{u}^{k} + \boldsymbol{f}\right) \tag{3.4}$$

For further analysis of this method it is useful to introduce the iteration matrix $\boldsymbol{M}$ as

$$\boldsymbol{M} = \boldsymbol{S}^{-1}\,(\boldsymbol{S} - \boldsymbol{K}) \tag{3.5}$$

In consideration of Eq. 3.5 the method of Eq. 3.4 corresponds to the scheme of the stationary iterative method (Eq. 3.1) with $\boldsymbol{Q} = \boldsymbol{M}$ and $\boldsymbol{q} = \boldsymbol{S}^{-1}\boldsymbol{f}$. From the iteration scheme of Eq. 3.1, the definition of the error in Eq. 2.10 and the equality $\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{u} + \boldsymbol{q}$, it follows that

$$\boldsymbol{e}^{k+1} = \boldsymbol{M}\boldsymbol{e}^{k} \tag{3.6}$$

The spectral radius ρ of a matrix is its largest absolute eigenvalue. The stationary iterative method converges if and only if the spectral radius ρ of the iteration matrix $\boldsymbol{M}$ satisfies the condition

$$\rho(\boldsymbol{M}) < 1 \tag{3.7}$$

This is true for any initial start vector $\boldsymbol{u}^{0}$ and any right hand side $\boldsymbol{f}$.

3.2 Jacobi Method

In case of the Jacobi method the diagonal $\boldsymbol{D}$ of the system matrix $\boldsymbol{K}$ is chosen as splitting matrix $\boldsymbol{S}_J = \boldsymbol{D}$. Then, analogous to Eq. 3.2 the decomposition reads as

$$\boldsymbol{K} = \boldsymbol{D} + (\boldsymbol{K} - \boldsymbol{D}) \quad \text{or} \quad \boldsymbol{K} = \boldsymbol{D} + (-\boldsymbol{L} - \boldsymbol{U}) \tag{3.8}$$

where $(-\boldsymbol{U})$ denotes the strictly upper triangle and $(-\boldsymbol{L})$ denotes the strictly lower triangle of the system matrix $\boldsymbol{K}$. Thus, in accordance to Eq. 3.4 the matrix form of the Jacobi method is defined by

$$\boldsymbol{u}^{k+1} = \boldsymbol{D}^{-1}\left((\boldsymbol{L} + \boldsymbol{U})\,\boldsymbol{u}^{k} + \boldsymbol{f}\right) \tag{3.9}$$

Correspondingly, the index form of the Jacobi method results as

$$u_i^{k+1} = \frac{1}{K_{ii}} \left(f_i - \sum_{j=1, j\neq i}^{n} K_{ij} u_j^k \right) \tag{3.10}$$

3.3 Gauss–Seidel Method

In case of the Gauss–Seidel method the splitting matrix is chosen as $\boldsymbol{S}_{GS} = (\boldsymbol{D} - \boldsymbol{L})$. Then, analogous to Eq. 3.2 the decomposition reads as

$$\boldsymbol{K} = (\boldsymbol{D} - \boldsymbol{L}) + (\boldsymbol{K} - (\boldsymbol{D} - \boldsymbol{L})) \quad \text{or} \quad \boldsymbol{K} = (\boldsymbol{D} - \boldsymbol{L}) + (-\boldsymbol{U}) \tag{3.11}$$

Thus, in accordance to the splitting method of Eq. 3.4 the Gauss–Seidel method is

$$\boldsymbol{u}^{k+1} = (\boldsymbol{D} - \boldsymbol{L})^{-1} \left(\boldsymbol{U}\boldsymbol{u}^k + \boldsymbol{f} \right) \tag{3.12}$$

Some further algebraic transformations of this equation are required for the derivation of the index form of the Gauss–Seidel method, such as

$$\boldsymbol{D}\boldsymbol{u}^{k+1} - \boldsymbol{L}\boldsymbol{u}^{k+1} = \boldsymbol{U}\boldsymbol{u}^k + \boldsymbol{f} \tag{3.13}$$

and

$$\boldsymbol{u}^{k+1} = \boldsymbol{D}^{-1} \left(\boldsymbol{L}u^{k+1} + \boldsymbol{U}u^k + \boldsymbol{f} \right) \tag{3.14}$$

where the index form is evident.

$$u_i^{k+1} = \frac{1}{K_{ii}} \left(f_i - \sum_{j=1}^{i-1} K_{ij} u_j^{k+1} - \sum_{j=i+1}^{n} K_{ij} u_j^k \right) \tag{3.15}$$

3.4 Comparison of Jacobi- and Gauss–Seidel Method

In the computation of the i-th element u_i^{k+1} in the $(k+1)$–iteration, the Gauss–Seidel method already includes all available iterates u_j^{k+1} with $j = 1 \ldots (i-1)$, while the Jacobi method only uses the u_j^k with $j = 1 \ldots N$ of the (k)–iteration. This is the only, but momentous difference in the algorithms of these two methods. In other words, while the Jacobi method adds all increments *simultaneously* only after cycling through all degrees of freedom, the Gauss–Seidel method adds all increments *successively* as soon available. As an advantage of the Gauss-Seidel method, one vector is sufficient to update its vector elements i successively in contrast to the Jacobi method where an additional vector is required. However, in practical problems this difference of memory demand is rather not relevant. As the operations for the iteration of different vector elements do not coincide in the Jacobi method, it is not dependent on the nodal ordering and parallelization is straightforward possible. In the Gauss-Seidel method the nodal ordering influences the convergence behavior. For regular grids specific nodal orderings are summarized in [5], such as the red–black, lexicographical, zebra-line or four-colour ordering. Advanced algorithms are required for successful parallelization of the Gauss-Seidel method or uncontrolled splitting of the process leads to the so-called *chaotic* Gauss-Seidel method. The convergence of both these stationary iterative methods is dependent on the spectral radius of the corresponding iteration matrix $\boldsymbol{M}$ as stated in Eq. 3.7. In general, if both methods converge, the convergence of the Gauss-Seidel method is better as each operation reverts to the latest data. The Jacobi method converges if the system matrix $\boldsymbol{K}$ is strictly diagonally dominant as a sufficient condition [6].

$$|K_{ii}| > \sum_{j=1, j\neq i}^{n} |K_{ij}| \qquad \text{for all } i \tag{3.16}$$

This does not apply to the considered structural finite element problems. However, convergence can be achieved by additional damping. The Gauss-Seidel converges if the system matrix $\boldsymbol{K}$ is strictly diagonally dominant or if the system matrix $\boldsymbol{K}$ is positive definite. The second condition is true, if the finite element model is properly restrained and stable [2]. Besides the use as standalone iterative solvers, the damped Jacobi method or the Gauss–Seidel can be applied as smoothers within the multigrid method.

3.5 Symmetric Gauss-Seidel Method

The symmetric Gauss-Seidel Method bases on two half steps, the classical Gauss-Seidel iteration, labeled as forward sweep (fGS), and additionally a backward sweep (bGS), with the splitting matrices

$$\boldsymbol{S}_{fGS} = \boldsymbol{D} - \boldsymbol{L} \quad \text{and} \quad \boldsymbol{S}_{bGS} = \boldsymbol{D} - \boldsymbol{U} \tag{3.17}$$

The following equations together describe one step of the symmetric Gauss-Seidel method

$$\boldsymbol{u}^{k+\frac{1}{2}} = (\boldsymbol{D} - \boldsymbol{L})^{-1} \left(\boldsymbol{U}\boldsymbol{u}^{k} \quad + \boldsymbol{f} \right) \tag{3.18}$$

$$\boldsymbol{u}^{k+1} = (\boldsymbol{D} - \boldsymbol{U})^{-1} \left(\boldsymbol{L}\boldsymbol{u}^{k+\frac{1}{2}} + \boldsymbol{f} \right) \tag{3.19}$$

By condensation of $\boldsymbol{u}^{k+\frac{1}{2}}$, it is straightforward to derive the effective splitting matrix

$$\boldsymbol{S}_{sGS} = (\boldsymbol{D} - \boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D} - \boldsymbol{U}) \tag{3.20}$$

of the symmetric Gauss-Seidel method and its effective iteration matrix

$$\boldsymbol{M}_{sGS} = (\boldsymbol{D} - \boldsymbol{U})^{-1}\boldsymbol{L}(\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{U} \tag{3.21}$$

Among other things, this matrix form of the iteration matrix is valuable for convergence analysis. However, for implementation issues it is reasonable to consider just the index form of the symmetric Gauss-Seidel method. The forward sweep corresponds to the Gauss-Seidel method of Eq. 3.15 assuming that i counts from 1 to n. Then in the subsequent backward sweep i counts backwards from n to 1. The symmetric Gauss-Seidel method has special properties. Summarily, it is generally not twice as fast convergent as the Gauss-Seidel method which results from analysis of the iteration matrix. On the other hand by a specific implementation with buffering of certain data, one iteration step of the symmetric method counts less operations than two of the Gauss-Seidel method. Nevertheless with respect to advanced solver methods, these details are rather marginal and will not be considered any further in this work. But as an essential advantage of the symmetric version, it is well suited as preconditioner.

3.6 Relaxation methods

Relaxation methods are stationary iterative methods. Thus, they can also be presented in the form of Eq. 3.1. For each of the previously introduced methods, there also exists a corresponding relaxation method.

In comparison to the precedent methods, the relaxation methods propose to scale each increment by a constant relaxation factor ω. The general form of the relaxation methods is given by the following short algorithmic expression

$$u_i^{k+1} := (1 - \omega)\, u_i^k + \omega \breve{u}_i^{k+1} \tag{3.22}$$

where for each individual index i, the temporary variable $\breve{u}_i^{k+1}$ is computed as u_i^{k+1} of the Jacobi method in case of the simultaneous over-relaxation method (JOR method) or as u_i^{k+1} of the Gauss–Seidel method in case of the successive over-relaxation method (SOR method). Then, if $\omega = 1$ this relaxation scheme is identical to the Jacobi and the Gauss–Seidel method, respectively.

The optimum relaxation factor can theoretically be derived from the spectral radius (largest absolute eigenvalue) of the iteration matrix. But as this is expensive, several methods on the practical determination of ω are proposed in [6] and [17]. Usually in finite element analysis the optimum relaxation for the SOR method is between 1.3 and 1.9 [2]. The convergence of the JOR method in a model case [10] was achieved by a heuristic damping factor of $\omega = 0.8$.

3.7 Conjugate Gradient Method

The conjugate gradient method has been developed by *Hestenes & Stiefel* [12]. In contrast to the method described in 3.1, the conjugate gradient method is instationary. It is presumed that the matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ and the vector $\boldsymbol{f} \in \mathbb{R}^n$ are real and that the matrix $\boldsymbol{K}$ is symmetric ($\boldsymbol{K} = \boldsymbol{K}^{\mathrm{T}}$) and positive definite

$$\boldsymbol{u}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{u} > 0 \qquad \text{for all } \boldsymbol{u} \neq \boldsymbol{0} \tag{3.23}$$

Then, the minimization problem of the quadratic form $F(x) = \min$

$$F(\boldsymbol{u}) = \frac{1}{2} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{u} - \boldsymbol{f}^{\mathrm{T}} \boldsymbol{u} \tag{3.24}$$

is equivalent to setting its derivative

$$\operatorname{grad} F(\boldsymbol{u}) = \boldsymbol{K} \boldsymbol{u} - \boldsymbol{f} \tag{3.25}$$

equal to the zero vector

$$\operatorname{grad} F(\boldsymbol{u}) = \boldsymbol{0} \tag{3.26}$$

The conjugate gradient method is an iterative minimizer of the provided quadratic form and thus an iterative solver method for the linear equation system $\boldsymbol{K} \boldsymbol{u} = \boldsymbol{f}$. As an important characteristic of the corresponding algorithm, the quadratic form is always minimized from an approximate vector $\boldsymbol{u}^k$ in the direction of a provided search vector $\boldsymbol{p}^k \neq \boldsymbol{0}$, which can be written as

$$F(\boldsymbol{u}^k + \lambda \boldsymbol{p}^k) = \min \tag{3.27}$$

with both $\boldsymbol{u}^k$ and $\boldsymbol{p}^k$ are constant vectors $\in \mathbb{R}^n$ and a scalar variable $\lambda \in \mathbb{R}$. In detail this leads to the following parabola function with respect to λ

$$\left(\frac{1}{2} \boldsymbol{p}^{k\mathrm{T}} \boldsymbol{K} \boldsymbol{p}^k \right) \lambda^2 + \left(\boldsymbol{p}^{k\mathrm{T}} \boldsymbol{K} \boldsymbol{u}^k - \boldsymbol{p}^{k\mathrm{T}} \boldsymbol{f} \right) \lambda + \left(\frac{1}{2} \boldsymbol{u}^{k\mathrm{T}} \boldsymbol{K} \boldsymbol{u}^k - \boldsymbol{u}^{k\mathrm{T}} \boldsymbol{f} \right) = \min \tag{3.28}$$

This yields that the quadratic form is minimized for

$$\lambda = \frac{\boldsymbol{p}^{k\mathrm{T}} \left(\boldsymbol{f} - \boldsymbol{K} \boldsymbol{u}^k \right)}{\boldsymbol{p}^{k\mathrm{T}} \boldsymbol{K} \boldsymbol{p}^k} \tag{3.29}$$

The ideal search direction $\boldsymbol{p}^k$ would be the error $\boldsymbol{e}^k$, but this would presume the knowledge of the exact solution $\boldsymbol{u}$. Therefore, the negative gradient of the quadratic form at $\boldsymbol{u}^k$ represents the

best intuitive search direction from the local view of $\boldsymbol{u}^k$. Then, with Eqs. 3.25, 2.8 and 2.11, the search direction corresponds to the residuum $\boldsymbol{r}^k$

$$-\text{grad}\, F(u^k) = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^k = \boldsymbol{r}^k \tag{3.30}$$

With $\boldsymbol{p}^k = \boldsymbol{r}^k$ in Eq. 3.29, the following equations can be defined

$$\lambda_k = \frac{\boldsymbol{r}^{k\mathrm{T}}\boldsymbol{r}^k}{\boldsymbol{r}^{k\mathrm{T}}\boldsymbol{K}\boldsymbol{r}^k} \tag{3.31}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \lambda_k \boldsymbol{r}^k \tag{3.32}$$

which describe one step of an iterative method which is called the Method of Steepest Descent due to the property that for any iteration step k the search direction $\boldsymbol{p}^k$ is defined by $\big(-\text{grad}\, F(\boldsymbol{u}^k)\big)$.

The Method of the Steepest Descent is a relevant step towards the Conjugate Gradient Method, but it turns out that the choice of the search directions $\boldsymbol{p}^k$ is not optimal. As $\boldsymbol{u}^{k+1}$ is optimized with respect to the previous search direction $\boldsymbol{p}^k = \boldsymbol{r}^k$, it is clear that successive search directions are orthogonal or in symbolic form $\big(-\text{grad}\, F(\boldsymbol{u}^{k+1}) \perp \boldsymbol{p}^k\big)$. However, although it can be shown that $\boldsymbol{r}^k \perp \boldsymbol{r}^{k+1}$ and $\boldsymbol{r}^{k+1} \perp \boldsymbol{r}^{k+2}$, it is generally not true that $\boldsymbol{r}^k \perp \boldsymbol{r}^{k+2}$. Therewith $\boldsymbol{u}^{k+2}$ has lost its optimum with respect to the previously optimized direction $\boldsymbol{r}^k$.

If u^{k+1} is *optimal with respect to* $p^k \neq \boldsymbol{0}$, then this property is passed to u^{k+2}, if and only if

$$\boldsymbol{K}\, p^{k+1} \perp p^k \tag{3.33}$$

and the vectors p^{k+1} and p^k are called *conjugate*. In the conjugate gradient method the search directions are pairwise conjugate. As each new direction is derived from the actual remaining residual and conjugate to the prior search direction, it is also conjugate to all previous search directions. Thus a system of conjugate search directions is obtained or equivalent a system of orthogonal residuals. This property can be proofed by induction. With the initial values

$$\boldsymbol{r}^0 = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^0\,; \quad \boldsymbol{p}^0 = \boldsymbol{r}^0 \tag{3.34}$$

the following equations describe the algorithm of the conjugate gradient method

$$\lambda_k = \frac{\boldsymbol{r}^{k\mathrm{T}}\boldsymbol{p}^k}{\boldsymbol{p}^{k\mathrm{T}}\boldsymbol{K}\boldsymbol{p}^k} \tag{3.35}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \lambda_k \boldsymbol{p}^k \tag{3.36}$$

$$\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \lambda_k \boldsymbol{K}\boldsymbol{p}^k \tag{3.37}$$

$$\boldsymbol{p}^{k+1} = \boldsymbol{r}^{k+1} - \frac{\boldsymbol{r}^{k+1\mathrm{T}}\boldsymbol{K}\boldsymbol{p}^k}{\boldsymbol{p}^{k\mathrm{T}}\boldsymbol{K}\boldsymbol{p}^k}\boldsymbol{p}^k \tag{3.38}$$

Eq. 3.35 corresponds to Eq. 3.29. Eq. 3.37 is equivalent to $\boldsymbol{r}^{k+1} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{k+1}$. As documented in e.g. [6] and with regard to an efficient implementation, it is possible to replace Eq. 3.35 by

$$\lambda = \frac{\boldsymbol{r}^{k\mathrm{T}}\boldsymbol{r}^k}{\boldsymbol{p}^{k\mathrm{T}}\boldsymbol{K}\boldsymbol{p}^k} \tag{3.39}$$

and Eq. 3.38 by

$$\boldsymbol{p}^{k+1} = \boldsymbol{r}^{k+1} + \frac{\boldsymbol{r}^{k+1\mathrm{T}}\boldsymbol{r}^{k+1}}{\boldsymbol{r}^{k\mathrm{T}}\boldsymbol{r}^k}\boldsymbol{p}^k \tag{3.40}$$

By exact arithmetic the conjugate gradient method will reach the exact solution of an n-dimensional problem after n steps. In theory this method can be regarded as a direct solver. However, due to numerical round-off errors the orthogonality is lost and such an ideal result is not achieved. In practice, with respect to a reasonable error tolerance the conjugate gradient method can generally be terminated after considerably less than n steps. This supports the view of the conjugate gradient method as an iterative method, while an iterative method should not converge to the exact solution (even or especially in theory) after n steps. Thus the conjugate gradient method is labeled as semi-iterative method.

4 MULTIGRID METHOD

4.1 Basic ideas of the Multigrid Method

The term multigrid method describes an iterative solver strategy rather than one certain algorithm. It is transparent to the FE practitioner's intuition that somehow a coarse mesh solution of a physical problem is a relative good approximation for the same problem on a fine mesh. Therefore certain operators need to be defined to transfer the *problem* from the fine to the coarse mesh and to transfer the *solution* in reverse direction. While it is clear, that a coarse mesh solution requires essentially less computational effort than the fine mesh solution. Then, a coarse mesh solution might provide a good start vector for the iterative process on the fine mesh. In a hierarchy of meshes one might adaptively step from the coarsest mesh to the finest mesh and start each iterative process on a certain mesh by an (approximate) solution of the previous coarser mesh. However, it is important to note that this scheme does not yet really touch the potential of the multigrid method. Assuming that for each mesh one of the introduced classical iterative solver methods is applied, the convergence of the iterative process becomes penally slow on the finest mesh. Then again, the current disequilibrium on the finest mesh, namely the current residual, can be regarded as a new physical problem and the same adaptive procedure through all meshes provides a fast approximation and accelerates the solution process on the fine mesh. This kind of the algorithm is called coarse grid correction. Again, this example rather sketches the basic idea. The sophisticated algorithm of the multigrid method applies certain schemes of *problem* and *solution* transfers in e.g. so called V- or W-cycles, while on the individual meshes the solution is improved by one or a few iteration steps of the classical iterative solver methods. These methods are referred to as smoothers, apparently as local disequilibrium is quickly improved or smoothed, while the global disequilibrium reduces very slowly when applied on a fine mesh. The multigrid method can be treated as stationary iterative solver method, as e.g. illustrated in [16].

4.2 Algorithms of the Multigrid Method

The multigrid method is applied to solve a linear equation system. Here, it is assumed that this equation system corresponds to a finite element problem on a uniform orthogonal mesh. This mesh, which defines the problem, is referred to as the finest mesh. It is assumed, that there exists a hierarchy of $m+1$ meshes $i = 0 \dots m$, where m denotes the finest mesh. The problem is defined as

$$\boldsymbol{K}_m \, \boldsymbol{u}_m = \boldsymbol{f}_m \tag{4.1}$$

The index c refers to any coarser mesh $0 \leq c < m$. The items (a) to (g) outline the algorithm of one coarse grid correction:

(a) Pre-Smoothing: The actual vector is $\boldsymbol{u}_m^k$. One or a few classical iteration steps are applied.

(b) Evaluation of residual forces on the finest grid

$$\boldsymbol{r}_m^k = \boldsymbol{f} - \boldsymbol{K}_m \boldsymbol{u}_m^k \tag{4.2}$$

(c) Restriction: Transfer of residual forces from fine grid m to coarse grid c by a corresponding matrix $\boldsymbol{P}_m^c$. In practice, the matrix-vector product corresponds to an algorithm.

$$\boldsymbol{r}_c := \boldsymbol{P}_m^c \, \boldsymbol{r}_m^k \tag{4.3}$$

(d) Correction: The actual coarse grid problem is then

$$\boldsymbol{K}_c \, \boldsymbol{u}_c = \boldsymbol{r}_c \tag{4.4}$$

Starting by an initial vector $\boldsymbol{u}_c^0 = \boldsymbol{0}$, one or a few iteration steps k^* of a classical iterative solver method yield an improved approximation $\boldsymbol{u}_c^{k^*}$ which is also a potential correction of the problem on mesh m.

(e) Prolongation: Interpolation of the correction $\boldsymbol{u}_c^{k^*}$ of grid c onto the fine mesh m by a corresponding matrix $\boldsymbol{P}_c^m$. Analogous to step (b) an algorithmic implementation of this operation is usual.

$$\Delta \boldsymbol{u}_m := \boldsymbol{P}_c^m \boldsymbol{u}_c^{k^*} \tag{4.5}$$

(f) The displacement increment on mesh m is added to the current displacement vector on mesh m which leads to a temporary variable $\hat{\boldsymbol{u}}$.

$$\hat{\boldsymbol{u}}_m = \boldsymbol{u}_m^k + \Delta \boldsymbol{u}_m \tag{4.6}$$

(g) Post-Smoothing: Only after one or a few classical iteration steps on $\hat{\boldsymbol{u}}_m$ the correction is complete and generates the next approximation $\boldsymbol{u}_m^{k+1}$.

Equation 4.4 states a problem similar to Equation 4.1. Therefore the algorithm can be restarted for the problem of Equation 4.4. For $m = 4$ such a recursive procedure is shown in the center of Fig. 4.1 and denoted as V-cycle. Generally it is proposed to solve the problem exactly by a direct solver on the coarsest mesh $i = 0$. In analogy to the shape of the letter V, also the possibility of a W-cycle shall only be mentioned (without illustration)[16]. On the left of Fig. 4.1 a F-cycle is illustrated. By the defined operations of restriction, prolongation and smoothing, it is straightforward to construct any of these or also other cycles.

In the present approach a modified cycle is implemented. It is illustrated on the right of Fig. 4.1. It requires mesh transfer operators between the finest and any coarser mesh. It is designed to balance the computational effort on the different meshes. The proposed number of iteration steps on mesh i is

$$s_i = c(m - i)^2 \tag{4.7}$$

where c is a heuristic scalar factor. The observed convergence was good for $c = 1$ and best for $c = 3 \ldots 5$ in the considered examples.

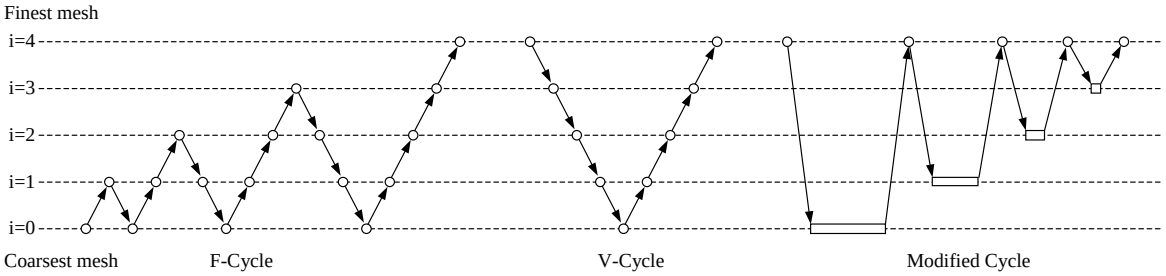

Fig. 4.1: Various multigrid cycles with the following steps: prolongation ($\nearrow$), restriction ($\searrow$), one or a few smoothing iterations ($\circ$) and a large number of smoothing iterations according to length of box ($\square$).

5 MULTIGRID PRECONDITIONED CONJUGATE GRADIENT METHOD

The Sections 3 and 4 introduce to the conjugate gradient method and classical iterative solver methods as a basis of the multigrid method. But before an efficient combination of these methods, in terms of the multigrid preconditioned conjugate gradient method, is presented, this topic is motivated with respect to the present engineering application.

The multigrid method is demonstrated as a very effective, iterative solver method for the mechanical analysis of heterogeneous material samples in [8]. However, an increase of the ratio of Young's moduli between inclusion and matrix material leads to a significantly slower solution process. The reason can be assigned to the worse conditioning of the problem as well as to the worse material representation on coarse grids. For a similar problem, the Poisson's equation with large coefficient jumps, it was shown in [15] that the multigrid preconditioned conjugate gradient method shows an essentially superior convergence rate over the multigrid method. In fact, the considered conjugate gradient acceleration can principally be applied to any iterative method [16]. Such an acceleration is not required or even disadvantageous, if the algorithm of the multigrid method is well designed and appropriate for the considered problem. Otherwise, one can try to identify and reduce the essential defect by e.g. additional local smoothing [16] or advanced algebraic transfer operators [3]. A potential improvement, which is less dependent of the considered problem, is given by applying the multigrid method as a preconditioner of the conjugate gradient method. An efficient combined preconditioning by the multigrid method and an adaption to shell elements has recently been proposed in [4]. In the present context multigrid preconditioning will be examined with respect to the aforementioned analysis of heterogeneous materials, whereas all global operations are formulated by effective, element-based procedures (Sections 7 to 10), such that the storage of a global stiffness matrix is superseded. This leads to a substantially reduced memory demand which is particularly relevant for achieving a high-resolution analysis on the material level.

A linear equation system $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$ can be preconditioned by a matrix $\boldsymbol{H}^{-1}$ as follows.

$$\boldsymbol{H}^{-1}\boldsymbol{K}\boldsymbol{u} = \boldsymbol{H}^{-1}\boldsymbol{f} \tag{5.1}$$

The matrix $\boldsymbol{H}^{-1}$ should be symmetric and positive definite [15]. The conditions of the multigrid preconditioner to match these properties are examined in [15]. According to [16] the multigrid method will potentially provide a valid preconditioner if the smoother is symmetric[1]. For a

[1]In the following study (Section 11) only the plain Gauss-Seidel method (Section 3.3) in connection with the proposed modified multigrid cycle (Section 4.2) has been applied, instead of e.g. the symmetric Gauss-Seidel method (Section 3.5). This first implementation of the algorithm also showed stable convergence in our examples. However, the current implementation will be updated to validate the numerical stability of this method.

derivation of the preconditioned conjugate gradient method a matrix $\boldsymbol{E}$ is defined which satisfies $\boldsymbol{H}^{-1} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ as exemplified in [16]. However, a final back substitution leads to an algorithm which only includes $\boldsymbol{H}^{-1}$.

The convergence is improved by Eq. 5.1 if the condition number of the preconditioned matrix $(\boldsymbol{H}^{-1}\boldsymbol{K})$ is lower than that of the original matrix $\boldsymbol{K}$, which can be determined from the analysis of eigenvalues [6]. If the preconditioning matrix would be $\boldsymbol{H}^{-1} = \boldsymbol{K}^{-1}$ then after one iteration step the exact solution $\boldsymbol{u}$ is found. In this sense, an ideal matrix $\boldsymbol{H}^{-1}$ is a good, but reasonably efficient approximation of $\boldsymbol{K}^{-1}$. With respect to the initial search direction, the vector $\boldsymbol{p}^0 = \boldsymbol{H}^{-1}\boldsymbol{r}^0$ would correspond to the error $-\boldsymbol{e}^0$ (Eq. 2.12), if $\boldsymbol{H}^{-1} = \boldsymbol{K}^{-1}$ (Section 3.7). An adequate matrix $\boldsymbol{H}^{-1}$ leads to an improved initial search direction $\boldsymbol{p}^0$. Therefore the preconditioned conjugate gradient method applies the following start conditions.

$$\boldsymbol{r}^0 = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^0 \; ; \quad \tilde{\boldsymbol{r}}^0 = \boldsymbol{p}^0 = \boldsymbol{H}^{-1}\boldsymbol{r}^0 \tag{5.2}$$

The following equations of the preconditioned conjugate gradient method are adopted from [15] in the notation of the conjugate gradient method in Section 3.7.

$$\lambda_k = \frac{\tilde{\boldsymbol{r}}^{k\mathrm{T}}\boldsymbol{r}^k}{\boldsymbol{p}^{k\mathrm{T}}\boldsymbol{K}\boldsymbol{p}^k} \tag{5.3}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \lambda_k \boldsymbol{p}^k \tag{5.4}$$

$$\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \lambda_k \boldsymbol{K}\boldsymbol{p}^k \tag{5.5}$$

$$\tilde{\boldsymbol{r}}^{k+1} = \boldsymbol{H}^{-1}\boldsymbol{r}^{k+1} \tag{5.6}$$

$$\boldsymbol{p}^{k+1} = \tilde{\boldsymbol{r}}^{k+1} + \frac{\tilde{\boldsymbol{r}}^{k+1\mathrm{T}}\boldsymbol{r}^{k+1}}{\tilde{\boldsymbol{r}}^{k\mathrm{T}}\boldsymbol{r}^k}\boldsymbol{p}^k \tag{5.7}$$

In each iteration step preconditioning only takes place in Eq. 5.6 and generates a new vector $\tilde{\boldsymbol{r}}^{k+1}$. For the present multigrid preconditioning the matrix $\boldsymbol{H}^{-1}$ is not explicitly build, but the operation defined in Eq. 5.6 is replaced by a multigrid cycle for the residual r^{k+1} and its solution is assigned to $\tilde{\boldsymbol{r}}^{k+1}$. Apart from this final transfer, it is noted that the the multigrid cycle for Eq. 5.6 needs to be performed on an own, individual set of variables which is independent of the variables in Eqs. 5.3 to 5.7. The preconditioned version of the method preserves a system of conjugate directions (Section 3.7), while each increment is optimized for each improved search direction based on the multigrid method. Thus, with respect to the present application, it follows that this optimization leads to considerably improved increments, if the stiffness of the coarse meshes is generally overestimated with further reference to Section 10.1.

6 ERROR MEASURES

Generally it is assumed that the exact error $\boldsymbol{e}$ is not known, such that an error measure is constructed in terms of the residual $\boldsymbol{r}$. In the following error measures the set A defines the sequence of all degrees of freedom with a prescribed force (also if equal to zero) and B all those with a prescribed displacement. The Euclidean norm of residual forces (in A) is a usual measure

$$\epsilon_{\mathrm{r}}^k = \left[\sum_{i \in A} (r_i^k)^2\right]^{\frac{1}{2}} \tag{6.1}$$

It is more objective to use the relative measure

$$\epsilon_{\mathrm{r}}^{k} = \frac{\left[\sum_{i\in A}(r_i^k)^2\right]^{\frac{1}{2}}}{\left[\sum_{i\in A}\left(f_i\right)^2 + \sum_{j\in B}\left(f_j^k\right)^2\right]^{\frac{1}{2}}} \tag{6.2}$$

similar to [14]. It is noted that in [14] part of the set B, of degrees of freedom with prescribed displacements, is probably not included. In [8] the following measure is proposed

$$\epsilon_{\mathrm{E}}^{k} = \frac{\sum_{i\in A}\left|r_i^k u_i^k\right|}{\sum_{i\in A} f_i u_i^k + \sum_{j\in B} f_j^k u_j} \tag{6.3}$$

This measure can be interpreted in terms of relative error in energy. A comparison of these measures will be part of our further studies. Generally this algebraic error in solving the linear equation system shall be considerably lower than other error sources as e.g. the discretization error of finite elements. With respect to an overall error efficiency an error measure in terms of error in energy would principally be comparable (which is also discussed in [9]).

7 ELABORATE PROPERTIES OF SOLID FINITE ELEMENTS

7.1 Scaling of solid finite elements

A finite element is denoted isoparametric if the displacements $\boldsymbol{U}$ within the finite element are evaluated by the same interpolation functions, assembled in $\boldsymbol{N}(\boldsymbol{X})$, as the element coordinates $\boldsymbol{X}$

$$\boldsymbol{X} = \boldsymbol{N}(\boldsymbol{X})\boldsymbol{x} \tag{7.1}$$
$$\boldsymbol{U} = \boldsymbol{N}(\boldsymbol{X})\boldsymbol{u} \tag{7.2}$$

where it is implied that the nodal coordinates $\boldsymbol{x}$ are arranged in the same order as the coordinates of nodal displacements $\boldsymbol{u}$. With regard to the multigrid method hierarchical finite element meshes will be defined with isoparametric finite elements being similar in shape but various in size. The matrices of these similar finite elements at various scale follow a certain relationship which will subsequently be evaluated.

It is assumed that the finite element geometry of an initial size is given in a coordinate system $\boldsymbol{Z}$. This finite element will be mapped to a coordinate system $\boldsymbol{X}$. The corresponding geometry and interpolation is then given by

$$\boldsymbol{X} = \boldsymbol{N}(\boldsymbol{Z})\boldsymbol{x} \tag{7.3}$$
$$\boldsymbol{U} = \boldsymbol{N}(\boldsymbol{Z})\boldsymbol{u} \tag{7.4}$$

The procedure follows [2, 18]. Therefore the following equality of partial derivatives is introduced

$$\frac{\partial}{\partial \boldsymbol{Z}} = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{Z}}\frac{\partial}{\partial \boldsymbol{X}}\,, \qquad \frac{\partial}{\partial Z_i} = \frac{\partial X_j}{\partial Z_i}\frac{\partial}{\partial X_j} \tag{7.5}$$

with the Jacobian matrix $\boldsymbol{J}$ defined as

$$\boldsymbol{J} = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{Z}} \,, \qquad J_{ij} = \frac{\partial X_j}{\partial Z_i} \tag{7.6}$$

The Jacobian matrix can be calculated by establishing the partial derivatives of $\boldsymbol{X}$ with respect to $\boldsymbol{Z}$ in terms of $\boldsymbol{N}(\boldsymbol{Z})\boldsymbol{x}$ according to Equation 7.3. For regular mappings the inverse of the Jacobian matrix can be created which provides the inverse relationship

$$\boldsymbol{J}^{-1} = \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{X}} \tag{7.7}$$

For the special case that the element is scaled by a constant factor c, expressed by $\boldsymbol{x} = c\boldsymbol{z}$, it follows that

$$J_{ij} = \delta_{ij} c \tag{7.8}$$

which directly leads to

$$\frac{\partial}{\partial Z_i} = c \frac{\partial}{\partial X_i} \qquad \text{and} \qquad \frac{\partial}{\partial X_i} = \frac{1}{c} \frac{\partial}{\partial Z_i} \tag{7.9}$$

From the kinematics it follows that the strain-displacement matrix $\boldsymbol{B}(\boldsymbol{X})$ exactly includes only terms of first-order partial derivatives with respect to X_i. Then, with Equation 7.9 the following relationship to $\boldsymbol{B}(\boldsymbol{Z})$ holds

$$\boldsymbol{B}(\boldsymbol{X}) = \frac{1}{c} \boldsymbol{B}(\boldsymbol{Z}) \tag{7.10}$$

The integral over the domain of the mapped geometry $\Omega_X(\boldsymbol{X})$ can be replaced by an integral over the domain of the initial geometry $\Omega_Z(\boldsymbol{Z})$

$$d\Omega_X = \det\boldsymbol{J} \, d\Omega_Z \tag{7.11}$$

where $\det\boldsymbol{J}$ is the determinant of the Jacobi matrix. According to Equation 7.8 and the dimension D of the domains, Ω_X and Ω_Z, the determinant of the Jacobi matrix is

$$\det\boldsymbol{J} = c^D \tag{7.12}$$

with D=1, 2 or 3. Therewith the element stiffness matrix of the finite element scaled to $\boldsymbol{X}$ by the constant factor c results as

$$\boldsymbol{K}_X = \boldsymbol{C}_{T,D} \int_{\Omega_X} \boldsymbol{B}(\boldsymbol{X})^{\mathrm{T}} \boldsymbol{C} \boldsymbol{B}(\boldsymbol{X}) d\Omega_X \tag{7.13}$$

where $C_{T,D}$ is a constant term. For three-dimensional elements the constant term is $C_{T,D=3} = 1$. For two-dimensional elements the thickness t is constant and therefore $C_{T,D=2} = t$. Analog, for one-dimensional elements the area A is not included in the integral which leads to $C_{T,D=1} = A$. With Equations 7.10 to 7.12 the stiffness matrix of Equation 7.13 is established in terms of $\boldsymbol{Z}$

$$\boldsymbol{K}_X = c^{D-2} \left(\boldsymbol{C}_{T,D} \int_{\Omega_X} \boldsymbol{B}(\boldsymbol{Z})^{\mathrm{T}} \boldsymbol{C} \boldsymbol{B}(\boldsymbol{Z}) d\Omega_Z \right) \tag{7.14}$$

This means that the stiffness matrix of the scaled element $\boldsymbol{K}_X$ is (c^{D-2})-times the stiffness matrix of the initial element $\boldsymbol{K}_Z$

$$\boldsymbol{K}_X = c^{D-2} \boldsymbol{K}_Z \tag{7.15}$$

Herewith, this characteristic is generally shown for solid finite elements of any dimension, independent of number of nodes per element and independent of the initial element geometry. For two-dimensional elements $(D = 2)$ this means that the stiffness matrix of an element does not change, if the element is scaled by a factor c. Otherwise, $(D \neq 2)$, Equation 7.15 establishes a simple relationship to create stiffness matrices for similar finite elements of different size based on an initial stiffness matrix.

7.2 Variation of elastic material properties

For linear elastic materials the Young's modulus E can be factored out of the material tensor $\mathbb{C}(E, \mu)$

$$\mathbb{C}(E, \mu) = E\mathbb{C}(1, \mu) \tag{7.16}$$

in contrast to Poisson's ratio μ. In straight equivalence this property is also true for the material matrix $\boldsymbol{C}$ of the finite element method. Under the condition that the Young's modulus E^e is constant within the domain of the finite element e, it can as well be factored out of the stiffness matrix

$$\boldsymbol{K}^e(E^e, \mu^e) = E^e \boldsymbol{K}^e(1, \mu^e) \tag{7.17}$$

For a regular, uniform grid of finite elements which only vary in Young's modulus E^e, this characteristic enables a fast evaluation of element stiffness matrices based on one initial matrix $\boldsymbol{K}^e(1, \mu^e)$. The equality of Equation 7.17 can be transfered to linear algebraic operations, such as the matrix–vector product of $\boldsymbol{K}^e(E^e, \mu^e)$ and an arbitrary vector $\boldsymbol{v}^e$

$$\left(E^e \boldsymbol{K}^e\left(1, \mu^e\right)\right) \boldsymbol{v}^e = E^e \left(\boldsymbol{K}^e\left(1, \mu^e\right) \boldsymbol{v}^e\right) \tag{7.18}$$

where the term of the right hand side counts less operations. Thus, it is neither required to explicitly build nor to store the stiffness matrices for finite elements with various Young's modulus to perform such operations on the element level.

The procedure is different for finite elements with various Poisson's ratio. It is possible to split the element stiffness matrix into parts, where each part is of equal order with respect to Poisson's ratio, such that it is possible to assemble element stiffness matrices of various Poisson's ratio efficiently.

The current implementation only includes operations according to Equation 7.18. The theoretical range of Poisson's ratio is limited to $\mu = [0.00; 0.50]$. For phases in concrete this range usually reduces to $\mu = [0.15; 0.25]$. It would be possible to cover the occurring range by a predefined set of stiffness matrices at equidistant Poisson's ratios, e.g. fifty-one stiffness matrices would cover the full range by an accuracy of $\mu = \pm 0.005$. For a very high number of different Poisson's ratio in the elements, this can be reasonable, as the usual variation of Poisson's ratio is less significant for the mechanical behaviour than the usual variation of Young's modulus. Furthermore, it is quite seldom that more than two effective digits of the Poisson's ratio are provided anyway. Otherwise, ideal for a low number of different Poisson's ratios, as in the implementation, a direct representation of initial stiffness matrices for all occurring Poisson's ratios is useful.

In fact, with the identities of Equation 7.15 and 7.17 for finite elements of various Young's modulus and various element size (but similar shape and equal Poisson's ratio), only one initial finite element stiffness matrix is required. On one hand this theoretical aspect is practical for implementation. Moreover, with respect to various Poisson's ratios and various element types, as in the B-spline finite element method [9], the number of initial element stiffness matrices to

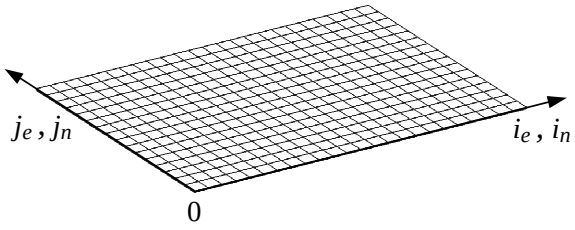

Fig. 8.1: Two-dimensional, uniform, orthogonal, mesh

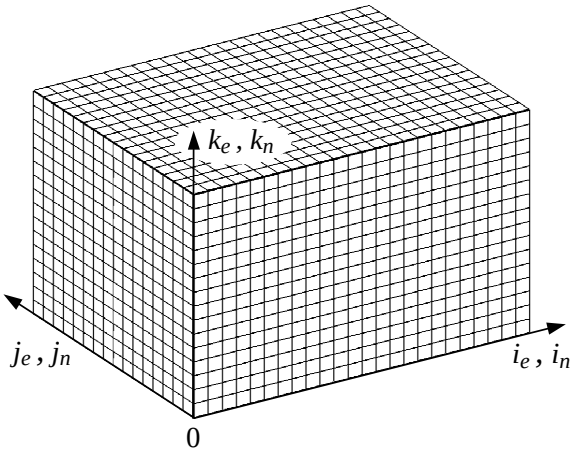

Fig. 8.2: Three-dimensional, uniform, orthogonal, mesh

be built can drastically be reduced. Especially with B-spline finite elements of higher order the corresponding gain in generation effort and memory requirement will be relevant.

8 DEFINITION OF GLOBAL FINITE ELEMENT PROBLEM ON ORTHOGONAL GRID

The priorly stated principle of virtual displacements leads to the finite element formulation which is applicable to problems of structural mechanics in general. Clearly, general finite element programs include the possibility to compute finite element problems on orthogonal grids. Due to their flexibility these programs usually include a element table to store the node numbers of each element and a node table to store the coordinates of each node. Moreover, the effective finite element problem will be stored in a global stiffness matrix. The solver method will be of a general kind for sparse matrices. For all postprocessing during or after computation the various tables need to be recalled. Thereby all element formulations will be based on a general description which is flexible in geometry.

Subsequently a special attempt is introduced for two- and three-dimensional finite elements on orthogonal grids with focus on high computational performance with low memory demand in respect to this particular problem. It will especially be applicable to the mechanical analysis of heterogeneous materials on the mesoscale, where a macroscopic simple domain of a corresponding test specimen is sufficient.

Figure 8.1 shows a two-dimensional mesh of $n_{ex} \times n_{ey}$ elements and $n_{nx} \times n_{ny}$ nodes where

$$n_{nx} = n_{ex} + 1 \tag{8.1}$$

$$n_{ny} = n_{ey} + 1 \tag{8.2}$$

The variables

$$i_e = 0 \ldots (n_{ex} - 1) \tag{8.3}$$

$$j_e = 0 \ldots (n_{ey} - 1) \tag{8.4}$$

refer to the elements and

$$i_n = 0 \ldots (n_{nx} - 1) \tag{8.5}$$

$$j_n = 0 \ldots (n_{ny} - 1) \tag{8.6}$$

refer to the nodes. The coordinates (x,y) of a node located at (i_n, j_n) is defined as

$$x = i_n \, l_a \tag{8.7}$$

$$y = j_n \, l_b \tag{8.8}$$

where l_a and l_b mean the element size in the corresponding direction. It follows that the total dimensions l_x and l_y of the mesh are

$$l_x = n_{ex} \, l_a \tag{8.9}$$

$$l_y = n_{ey} \, l_b \tag{8.10}$$

The numbering of elements p_e and nodes p_n is defined according to

$$p_e = i_e + j_e \, n_{ex} \tag{8.11}$$

$$p_n = i_n + j_n \, n_{nx} \tag{8.12}$$

so that $p_e \in [0\,;\ (n_{exy} - 1)]$ with $n_{exy} = n_{ex} n_{ey}$ and $p_n \in [0\,;\ (n_{nxy} - 1)]$ with $n_{nxy} = n_{nx} n_{ny}$. The finite element problem can be defined on vectors of corresponding dimensions. Such essential vectors are the displacements $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$, the type of boundary condition[2] $\boldsymbol{b}_{Tx}$ and $\boldsymbol{b}_{Ty}$ and values of boundary condition $\boldsymbol{b}_{Vx}$ and $\boldsymbol{b}_{Vy}$. For each element an individual Young's modulus can be defined and therefore a corresponding vector $\boldsymbol{v}_E$ is allocated. By an integer vector $\boldsymbol{v}_I$ each element can link to a specified material. The material definition includes the Poisson's ratio for linear analysis and may include several other variables such as the stress limit for nonlinear analysis. Therewith the basic finite element definitions are outlined.

However, a damage state of the structure still requires at least one damage variable for each element [11]. Iterative solver methods additionally require some vectors such as displacement increments and other swap memory. Furthermore for graphical processing some additional, primarily static memory is appropriate. A summary of the overall memory demand is provided in [8].

The extension to a three-dimensional mesh as shown in Figure 8.2 is perfectly straightforward. For the three-dimensional mesh of $n_{ex} \times n_{ey} \times n_{ey}$ elements and $n_{nx} \times n_{ny} \times n_{ny}$ nodes only the following definitions need to be added.

$$n_{nz} = n_{ez} + 1 \qquad \text{analog to Eqs. 8.1, 8.2} \tag{8.13}$$

$$k_e = 0 \ldots (n_{ez} - 1) \qquad \text{analog to Eqs. 8.3, 8.4} \tag{8.14}$$

$$k_n = 0 \ldots (n_{nz} - 1) \qquad \text{analog to Eqs. 8.5, 8.6} \tag{8.15}$$

$$z = k_n \, l_c \qquad \text{analog to Eqs. 8.7, 8.8} \tag{8.16}$$

$$l_z = n_{ez} \, l_c \qquad \text{analog to Eqs. 8.9, 8.10} \tag{8.17}$$

Equivalently, the element numbering p_e and node numbering p_n is extended.

$$p_e = i_e + j_e \, n_{ex} + k_e \, n_{exy} \qquad \text{analog to Eq. 8.11} \tag{8.18}$$

$$p_n = i_n + j_n \, n_{nx} + k_n \, n_{nxy} \qquad \text{analog to Eq. 8.12} \tag{8.19}$$

Also the number of vectors needs to be extended, as e.g. by $\boldsymbol{u}_z$, $\boldsymbol{b}_{Tz}$, $\boldsymbol{b}_{Vz}$ and so on. With respect to various operations of the proposed implementation, It is most appropriate to store these vectors of concurrent meaning, but various orientation (x, y or z), in one continuous data field. For the displacements this means a continuous vector field[3] $\boldsymbol{u}$ of $(\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_z)$ so that global vector operations with $\boldsymbol{u}$ are simply defined, while the subvectors remain directly accessible.

[2] Type of boundary condition is e.g. "0" for prescribed force or "1" for prescribed displacement.

[3] Without consideration of boundary conditions.

9 ELEMENT-BASED GLOBAL OPERATIONS

A two-dimensional and three-dimensional uniform, orthogonal grid of finite elements is considered. Due to the prior definitions, for both cases, iterative solver methods can be applied without storage of the global stiffness matrix. Therefore some global operations will be prepared based on operations of a local finite element topology.

9.1 Global matrix-vector product

For the global matrix vector product $\boldsymbol{K}\boldsymbol{v}$, of global stiffness matrix $\boldsymbol{K}$ and arbitrary global vector $\boldsymbol{v}$, Equation 2.2 is recalled. It follows that this global matrix-vector product can be formulated as a sum of matrix-vector products on the element level

$$\boldsymbol{K}\boldsymbol{v} = \sum_e \left(\bar{\boldsymbol{K}}^e \boldsymbol{v}\right) = \sum_e \left(\bar{\boldsymbol{K}}^e \bar{\boldsymbol{v}}^e\right) \tag{9.1}$$

with element stiffness matrix $\bar{\boldsymbol{K}}^e$ and corresponding displacements $\bar{\boldsymbol{v}}^e$ of element e where the bar over the symbols highlights the assignment to global degrees of freedom. However, it is more practical to perform these operations on the element level within local degrees of freedom, also found in [14]. For the element e the appropriate components of $\boldsymbol{v}$ are gathered in $\boldsymbol{v}^e$. The local operation $\boldsymbol{K}^e\boldsymbol{v}^e$ is performed and the result is added to the corresponding components of the global result vector. This procedure is performed for all elements, in analogy to Equation 9.1 (r.h.s.).

9.2 Operations on finite elements stencils

As one basic idea of the finite element method a body will be divided into parts such that each displacement interpolation function has only small support in the domain of the body. Each displacement interpolation function, or shape function, is assigned to a degree of freedom at a specific node. Therefore, the direct interaction radius of this node is limited. Considering a specific node, a direct stiffness relationship will only be defined to nodes which are attached to the same elements[4].

As a result, an adequate global numbering implied, the global stiffness matrix will be sparse. Among other things, this is an advantageous characteristic for direct solver methods. Special data structures have been developed to store sparse matrices. The following operation is considered

$$f_i = \sum_j K_{ij} u_j \tag{9.2}$$

where i and j refer to the global degrees of freedom. For large matrices $\boldsymbol{K}$ only a small fraction of entries K_{ij} will not be zero. With data structures of sparse matrices all, or most, zero entries are not even processed at all to evaluate the product of Equation 9.2. The same effect is achieved by only including nodal degrees of freedom which refer to the same elements as the degree of freedom i, which motivates a local node-by-node iterative solver method without storage of a global stiffness matrix.

Such a local scheme will especially be efficient if it is recurrent. Then, it is labeled as finite element stencil. This principle will be exemplified for uniform orthogonal grids of four-node two-dimensional and eight-node three-dimensional finite elements. However, it is noted that

[4]This is generally true, but with B-spline finite elements [9] the support of the shape functions will increase and cover several elements dependent on the order of the B-splines.

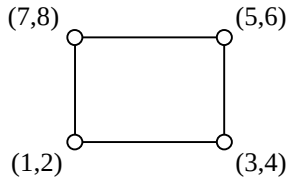

Fig. 9.1: Dof of one element

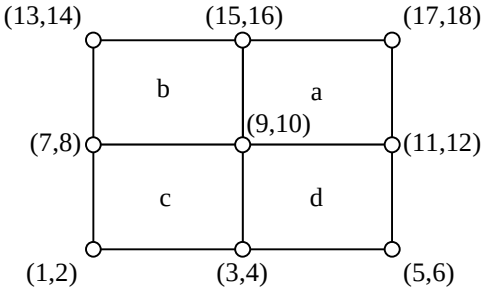

Fig. 9.2: Dof of 2x2 element patch

the fundamental idea is more general, as previously introduced, and may be transfered to finite elements of other shape (e.g. triangular) or of polynomial order, while it can also be regarded as sparse data structure technique.

On a uniform, orthogonal grid of finite elements, for any inner node the topology to its neighbouring nodes is identical. The finite element stencil is similar to the finite difference stencil while the first is based on the principle of virtual displacements and the latter refers to a direct discretization of the differential equation [5].

9.3 Stencil for grid of two-dimensional four-node elements

For a uniform orthogonal grid of finite elements, the degrees of freedom of any interior node can locally and temporarily assigned as (9,10) which refers to the horizontal and vertical component, respectively (Fig. 9.2). The element stiffness matrix with Young's modulus $E = 1$ is denoted as basic element stiffness matrix $\bar{\boldsymbol{K}}(\mu) = \boldsymbol{K}_{\mathrm{e}}(1, \mu)$. Then, the stiffness of the center node can be assembled as follows

$$\hat{K}_{9,9} = E^a \bar{K}(\mu^a)_{1,1} + E^b \bar{K}(\mu^b)_{3,3} + E^c \bar{K}(\mu^c)_{5,5} + E^d \bar{K}(\mu^d)_{7,7} \tag{9.3}$$

$$\hat{K}_{9,10} = E^a \bar{K}(\mu^a)_{1,2} + E^b \bar{K}(\mu^b)_{3,4} + E^c \bar{K}(\mu^c)_{5,6} + E^d \bar{K}(\mu^d)_{7,8} \tag{9.4}$$

$$\hat{K}_{10,10} = E^a \bar{K}(\mu^a)_{2,2} + E^b \bar{K}(\mu^b)_{4,4} + E^c \bar{K}(\mu^c)_{6,6} + E^d \bar{K}(\mu^d)_{8,8} \tag{9.5}$$

The superscript a to d refers to the system of one element according to Fig. 9.1. The hat-symbol denotes the reference to the 2×2 element patch of Fig. 9.2. The corresponding residual forces $\hat{r}_9$ and $\hat{r}_{10}$ are calculated by

$$\hat{r}_9 = \hat{f}_9 - \sum_{i=1}^{8} \left(E^a \bar{K}(\mu^a)_{1,i} u_i^a + E^b \bar{K}(\mu^b)_{3,i} u_i^b + E^c \bar{K}(\mu^c)_{5,i} u_i^c + E^d \bar{K}(\mu^d)_{7,i} u_i^d \right) \tag{9.6}$$

$$\hat{r}_{10} = \hat{f}_{10} - \sum_{i=1}^{8} \left(E^a \bar{K}(\mu^a)_{2,i} u_i^a + E^b \bar{K}(\mu^b)_{4,i} u_i^b + E^c \bar{K}(\mu^c)_{6,i} u_i^c + E^d \bar{K}(\mu^d)_{8,i} u_i^d \right) \tag{9.7}$$

for any interior node, or after correct assignment of $u^{a..d}$ to $\hat{u}$ and corresponding assembly of $\hat{K}_{9,i}$ and $\hat{K}_{10,i}$ from Eqs. (9.6) and (9.7) by

$$\hat{r}_9 = \hat{f}_9 - \sum_{i=1}^{18} \hat{K}_{9,i} \hat{u}_i \tag{9.8}$$

$$\hat{r}_{10} = \hat{f}_{10} - \sum_{i=1}^{18} \hat{K}_{10,i} \hat{u}_i \tag{9.9}$$

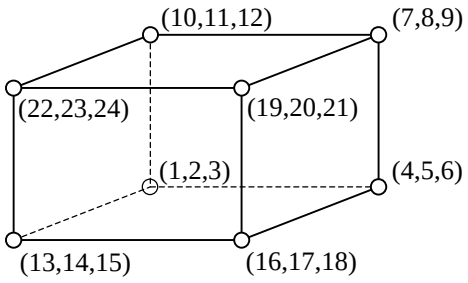

Fig. 9.3: Dof of one brick element

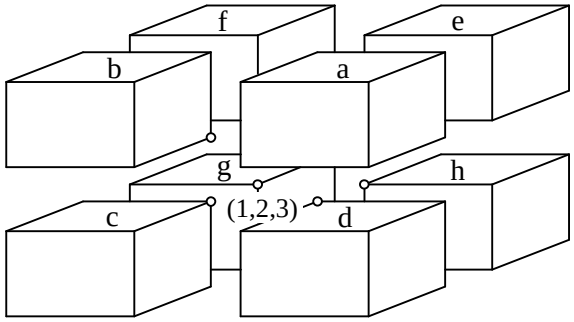

Fig. 9.4: $2 \times 2 \times 2$ brick elements

Boundary nodes can be included by omitting the terms of inexistent neighboring elements. Local equilibrium at this node with respect to the degree of freedom i only, is achieved by adding the increment $\Delta\hat{u}_i$ to $\hat{u}_i$.

$$\Delta\hat{u}_i = \frac{\hat{r}_i}{\hat{K}_{ii}} \tag{9.10}$$

Such an increment can either refer to Eq. 3.10 of the Jacobi method or to Eq. 3.15 of the Gauss-Seidel method (Section 3, especially 3.4).

Local equilibrium for both degrees of freedom $\hat{u}_9$ and $\hat{u}_{10}$ is established by the local increment vector

$$\begin{bmatrix} \Delta\hat{u}_9 \\ \Delta\hat{u}_{10} \end{bmatrix} = \begin{bmatrix} \hat{K}_{9,9} & \hat{K}_{9,10} \\ \hat{K}_{10,9} & \hat{K}_{10,10} \end{bmatrix}^{-1} \begin{bmatrix} \hat{r}_9 \\ \hat{r}_{10} \end{bmatrix} \tag{9.11}$$

For four rectangular finite elements of same Poisson's ratio and same shape several prior equations can be shortened as due to the equalities

$$\bar{K}(\mu)_{1,1} = \bar{K}(\mu)_{3,3} = \bar{K}(\mu)_{5,5} = \bar{K}(\mu)_{7,7} \tag{9.12}$$

$$\bar{K}(\mu)_{1,2} = -\bar{K}(\mu)_{3,4} = \bar{K}(\mu)_{5,6} = -\bar{K}(\mu)_{7,8} \tag{9.13}$$

$$\bar{K}(\mu)_{2,2} = \bar{K}(\mu)_{4,4} = \bar{K}(\mu)_{6,6} = \bar{K}(\mu)_{8,8} \tag{9.14}$$

e.g. the Equations 9.3 to 9.5 can be simplified to

$$\hat{K}(\mu)_{9,9} = (E^a + E^b + E^c + E^d)\bar{K}(\mu)_{1,1} \tag{9.15}$$

$$\hat{K}(\mu)_{9,10} = (E^a - E^b + E^c - E^d)\bar{K}(\mu)_{1,2} \tag{9.16}$$

$$\hat{K}(\mu)_{10,10} = (E^a + E^b + E^c + E^d)\bar{K}(\mu)_{2,2} \tag{9.17}$$

9.4 Stencil for grid of three-dimensional eight-node elements

The extension of the stencil from the two-dimensional to the three-dimensional problem is straightforward. Nevertheless, the number of degrees of freedoms essentially increases. The notation is analog. The superscript a to h refers to the system of one brick element according to Fig. 9.3. The hat-symbol denotes the reference to the eight brick elements of Fig. 9.4. In this Figure only the degrees of freedom $(1, 2, 3)$ which are assigned to the highlighted node will subsequently be relevant. The stiffness of this node is assembled by

$$\hat{K}_{1,1} = E^a\bar{K}(\mu^a)_{1,1} + E^b\bar{K}(\mu^b)_{4,4} + \ldots + E^h\bar{K}(\mu^h)_{22,22} \tag{9.18}$$

$$\hat{K}_{2,2} = E^a\bar{K}(\mu^a)_{2,2} + E^b\bar{K}(\mu^b)_{5,5} + \ldots + E^h\bar{K}(\mu^h)_{23,23} \tag{9.19}$$

$$\hat{K}_{3,3} = E^a\bar{K}(\mu^a)_{3,3} + E^b\bar{K}(\mu^b)_{6,6} + \ldots + E^h\bar{K}(\mu^h)_{24,24} \tag{9.20}$$

and the residual forces are

$$\hat{r}_1 = \hat{f}_1 - \sum_{i=1}^{24} \left(E^a \bar{K}(\mu^a)_{1,i} u_i^a + E^b \bar{K}(\mu^b)_{4,i} u_i^b + \ldots + E^h \bar{K}(\mu^h)_{22,i} u_i^h \right) \tag{9.21}$$

$$\hat{r}_2 = \hat{f}_2 - \sum_{i=1}^{24} \left(E^a \bar{K}(\mu^a)_{2,i} u_i^a + E^b \bar{K}(\mu^b)_{5,i} u_i^b + \ldots + E^h \bar{K}(\mu^h)_{23,i} u_i^h \right) \tag{9.22}$$

$$\hat{r}_3 = \hat{f}_3 - \sum_{i=1}^{24} \left(E^a \bar{K}(\mu^a)_{3,i} u_i^a + E^b \bar{K}(\mu^b)_{6,i} u_i^b + \ldots + E^h \bar{K}(\mu^h)_{24,i} u_i^h \right) \tag{9.23}$$

or in a generic form where $\alpha = 1 \ldots 8$ means $a \ldots h$ in the index.

$$\hat{r}_j = \hat{f}_j - \sum_{\alpha=1}^{8} E^\alpha \sum_{i=1}^{24} \bar{K}(\mu^\alpha)_{[3(\alpha-1)+j],i} u_i^\alpha \tag{9.24}$$

Analog to Eq. 9.10 the increment according to the stationary iterative method can be computed for the three-dimensional case.

9.5 Local schemes of advanced finite elements for heterogeneous material

In the prior sections constant material properties are assigned to each element. This leads to plain pixel or voxel discretization. This method is applicable to obtain reasonable overall properties of a heterogeneous material sample, but due to the grid discretization there will be errors in the stress solution along material interfaces.

It is possible to improve the accuracy of the geometrical discretization by advanced finite elements while maintaining a uniform orthogonal mesh. One option are multiphase B-spline finite elements as presented in [9]. This discretization type is illustrated in Fig. 9.5 (left). The large black dots indicate integration points to each of which individual material properties can be assigned. Thus a smooth transition of material properties is possible within a finite element. As all finite elements are based on the same topology, similar local schemes as described in Section 9.1 to 9.4 can be created and the storage of a global stiffness matrix is not required.

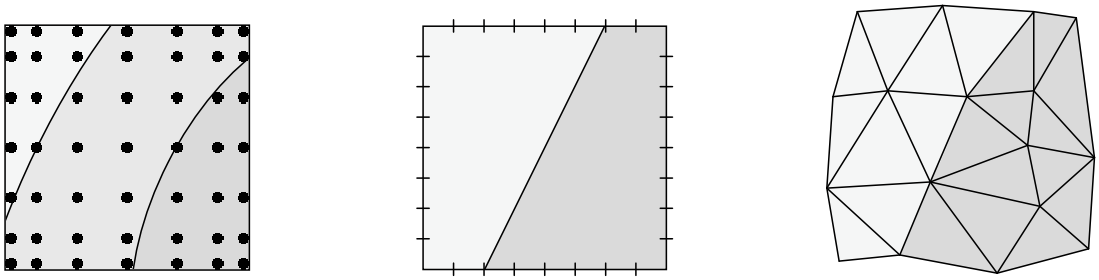

Fig. 9.5: Discretization of heterogeneous material by a multiphase finite element (left), a finite element enriched by additional shape functions (center) and an aligned triangular mesh (right).

The same is principally possible for other advanced finite elements with additional shape functions which are declared as internal degrees of freedom (finite elements with embedded discontinuities) or as global degrees of freedom (extended finite elements). As a condition the material interfaces is not completely variable but assigned to a finite predefined number of possibilities. In Fig. 9.5 (center) from corner to corner of one edge there are nine different settings,

marked by a short line. According to this pattern there are $(23 \cdot 7 + 15)4/2 = 352$ different possible material interfaces which can be assigned to the element. The stiffness matrices need to be stored decoupled for the different material phases in the element. Then, it is possible to include such finite elements into a local scheme of a uniform grid, based on a comparatively low number of predefined element stiffness matrices. And the storage of a global stiffness matrix would not be required.

9.6 Local schemes of irregular triangular mesh

This section shall only briefly highlight that also for a irregular mesh of triangular finite elements (Fig. 9.5 (right)), required parts of a global stiffness matrix can efficiently be assembled based on a comparatively low number of element stiffness matrices. As the element stiffness matrix is invariate with respect to linear scaling of the finite element (Section 7.1), only the angles α and β are required to determine the relevant shape of the finite element. With the definition that $\alpha \leq \beta \leq \gamma$ it follows that there are the following bounds on α and β

$$0^\circ < \alpha \leq 60^\circ \tag{9.25}$$

$$\alpha \leq \beta < 90^\circ \tag{9.26}$$

Hence, for an allowed tolerance of $\pm 0.5^\circ$ there are *only* ≈ 3600 different elements stiffness matrices required. Especially for very large models with $\geq 10^6$ degrees of freedom such an element library represents an effective alternative to storing a global stiffness matrix. Global operations on these elements can be performed as illustrated in Fig. 9.6. The global degrees of freedom $\boldsymbol{u}^e$ are transformed into local degrees of freedom $\bar{\boldsymbol{u}}^e$. These can be multiplied by the basic stiffness matrix $\bar{\boldsymbol{K}}^e(\alpha, \beta)$ to obtain the effective force vector $\bar{\boldsymbol{f}}^e$. A multiplication with the Young's modulus of this element is also required. Then the force vector can be transformed back into the global coordinate system $\boldsymbol{f}^e$. In addition for each element only a transformation angle and a triangle shape type (with respect to α and β) needs to be stored. It is further noted, that this principle is not limited to the three-node triangular finite elements as long as a definition is unique (nodes on center of element edge). It is only a matter of model size when it becomes more efficient to use a predefined library of finite elements. This supports the basic idea of this article to reduce the memory demand to a minimum.

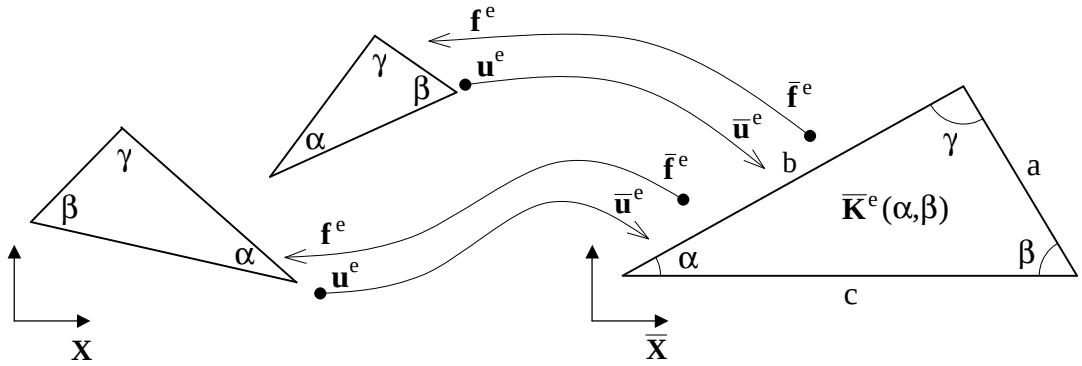

Fig. 9.6: Scheme of establishing a product of element stiffness matrix and displacement vector in a global coordinate system $\boldsymbol{X}$ based on a triangular element with angles α and β in a local coordinate system $\bar{\boldsymbol{X}}$.

10 TRANSFER OPERATORS

10.1 Fine-to-coarse mesh transformation of Young's modulus

A finite element mesh of a heterogeneous material sample is considered. Here, one individual Young's modulus is assigned to each element of a uniform orthogonal grid. As an example in Fig. 10.1 the relevant finite element mesh is at level 7. All coarser grids of finite elements shall represent a good approximation of the heterogeneous material distribution which is defined on the finest mesh. Therefore it is possible to define adequate algebraic operators as discussed in [3]. However, in this approach only the geometric multigrid method is considered such that the local scheme of finite elements can be applied on each mesh without additional memory demand for storing matrices of operators or coarse grid stiffnesses. A doubling of mesh size means that four finite elements of the original mesh are represented by one element on the next coarser mesh. This corresponds to a kind of homogenization. An adequate effective Young's modulus of this element is in between the Reuss bound (lower bound) and the Voigt bound (upper bound) [8]. But as the original patch of elements might have different effective stiffnesses with respect to different directions, according to the prior definitions, such an anisotropic property can not be assigned to one coarse finite element. In this approach just the Voigt bound is applied which corresponds to the arithmetic mean of the various original Young's moduli. Therefore the coarser meshes will effectively be too stiff, which will result in slower but stable convergence. For larger ratios of Young's modulus in the heterogeneous material this defective overestimation of stiffness will become more severe (which explains the remedy by the multigrid preconditioned conjugate gradient method in connection with the last paragraph of Section 5). For the Reuss bound the iteration process can become instable. The Hashin-Shtrikman bound as illustrated in [8] would be an appropriate alternative to define effective Young's moduli on coarser meshes based on the finest mesh. This option would establish an approximation which is closest to the original problem. But for some problems this choice might not be convergent. For the sake of solving stability, the Voigt bound is applied. A corresponding example is shown in Fig. 10.1.

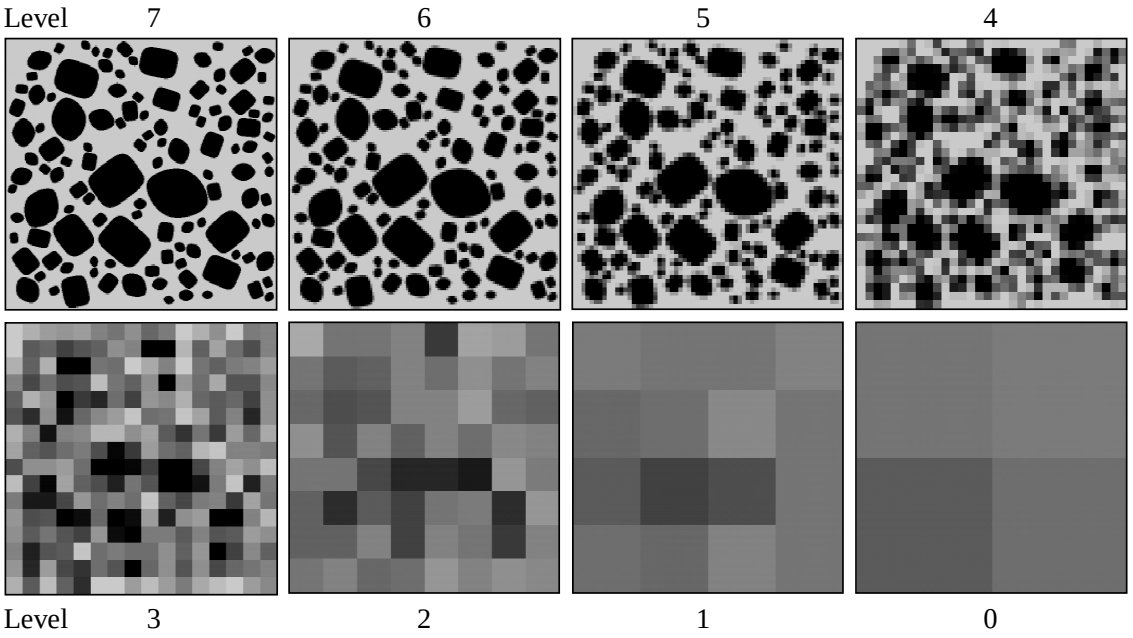

Fig. 10.1: Definition of Young's modulus on finest mesh (Level 7) and corresponding coarse grid approximations of Young's modulus (Level 6 to 0)

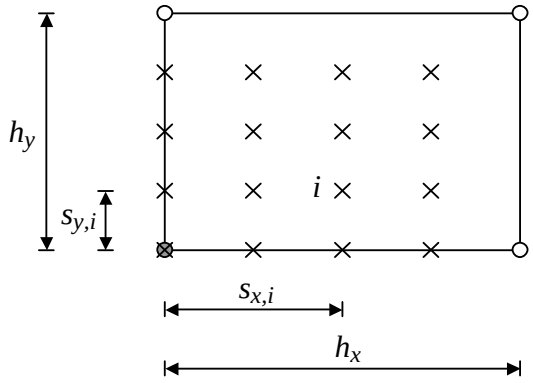

Fig. 10.2: A rectangular four-node finite element of size $h_x \times h_y$ and some selected point loads located at $(s_{x,i}, s_{y,i})$, which may result from the nodal disequilibrium of a finer mesh, relevant with respect to the grey node of the element (lower left corner).

10.2 Restriction: Fine-to-coarse mesh transformation of forces

The starting point is any arbitrary vector of forces on a fine mesh. A best possible, equivalent representation of such forces on a coarse mesh is intended. There are several different restriction operators defined in connection with the multigrid method [16]. The applied restriction operator corresponds to the fundamental finite element definition with respect to the mechanical problem.

$$\boldsymbol{f}^e = \sum_i \left(\boldsymbol{N}^e_{\text{at } \boldsymbol{X}_i}\right)^{\mathrm{T}} \boldsymbol{F}_i \tag{10.1}$$

The vector $\boldsymbol{F}_i$ denotes a point load[5] at position $\boldsymbol{X}_i$ within the element e. This load is assigned to the force vector $\boldsymbol{f}^e$ of element e by the evaluation of the element shape function $\boldsymbol{N}^e$ at coordinate $\boldsymbol{X}_i$. In Fig. 10.2 a force $\boldsymbol{F}_i$ ($\times$-symbol), e.g. from a finer mesh, is assigned to the nodal force vector of the lower left node $\boldsymbol{f}_1$ (grey node) of a bilinear finite element. This yields

$$\boldsymbol{f}_{1,i} = \left(1 - \frac{s_{x,i}}{h_x}\right)\left(1 - \frac{s_{y,i}}{h_y}\right)\boldsymbol{F}_i \tag{10.2}$$

With respect to an implementation it can further be utilized that in the considered example the relevant values of the fractions $\frac{s_{x,i}}{h_x}$ and $\frac{s_{y,i}}{h_y}$ are limited to $\frac{s}{h} \in \left[0; \frac{1}{4}; \frac{1}{2}; \frac{3}{4}\right]$. A corresponding predefined template minimizes the computational effort in contrast to the formal symbolic definition of Eq. 10.1. An extension to three dimensions is straightforward.

10.3 Prolongation: Coarse-to-fine mesh interpolation of displacements

The relationship to determine the displacement $\boldsymbol{U}$ at coordinate $\boldsymbol{X}_i$ within an element e with nodal displacements u^e is defined by the interpolation or shape function $\boldsymbol{N}^e$.

$$\boldsymbol{U}_{\text{at } \boldsymbol{X}_i} = \boldsymbol{N}^e_{\text{at } \boldsymbol{X}_i}\, \boldsymbol{u}^e \tag{10.3}$$

For usual finite elements the displacement at a coordinate of a node corresponds to the nodal value of the degree of freedom. Therefore the basic Equation 10.3 is sufficient to construct any coarse-to-fine mesh interpolation operator. Again, specific templates can reduce the computational effort.

[5]Point loads are not included in the classical mechanical theory, but their contribution within the principle of virtual displacements is clear, as e.g. shown in [2]. In present context, for the computation of an equivalent coarse mesh residual, the problem of a singularity does not apply.

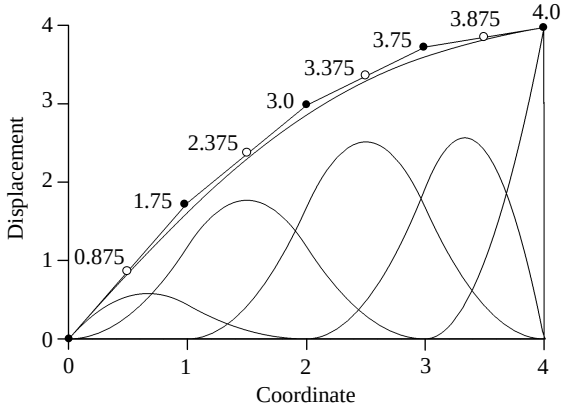

Fig. 10.3: A piecewise, linear function (•) is approximated by quadratic B-splines. The values at (○) denote the coefficient which are assigned to the individual B-spline functions.

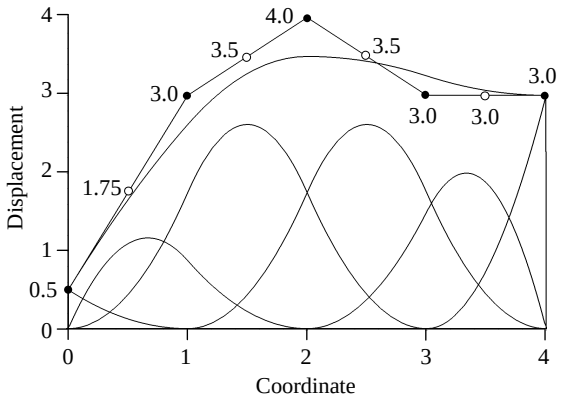

Fig. 10.4: Another example, analog to Fig. 10.3.

10.4 Transfer operators for B-spline finite elements

A multigrid method for web-splines[6] is presented in [13]. In [9] an alternative method is outlined as an extension to the existing multigrid scheme presented here. It includes only transfer operations between a mesh of classical linear finite elements (B-splines of order $k = 1$) and a mesh of B-Spline finite elements of order $k = 2$ with an identical number of elements. The B-spline finite element problem is only transfered to the classical finite elements and then the present multigrid method with classical finite elements can be applied. After each cycle the displacement increment is transfered to the B-spline mesh again. After that smoothing is performed on the finest mesh of B-spline finite elements. It is assumed that the transfer operators need to be reasonably adequate, but not exact, as each coarse grid correction step is always an approximation only.

The restriction operator is not discussed in detail. From the residual forces of a B-spline finite element one could retrieve corresponding body loads $\boldsymbol{p}_b^e$ or surface loads $\boldsymbol{p}_s^e$, which can be compiled into a corresponding element force vector $\boldsymbol{f}^e$.

$$\boldsymbol{f}^e = \int_{V^e} \boldsymbol{N}^{e\mathrm{T}} \boldsymbol{p}_b^e dV + \int_{A^e} \boldsymbol{N}^{e\mathrm{T}} \boldsymbol{p}_s^e dA \tag{10.4}$$

The prolongation operator is more involving. A C^0-continuous displacement solution of bilinear finite elements can not exactly be mapped onto a B-spline finite element space which is continuous in its first partial derivatives by definition. An exact interpolation of B-splines through all nodal values would require to build and solve an equation system. This would be very costly. Alternatively an intuitive operator is introduced and exemplified for one-dimensional examples.

For the transfer of a linear displacement field this operator is exact. This will be clear with respect to the following example and in connection with [9]. Figure 10.3 shows four piecewise linear functions with nodal values (•) on $U = -\frac{1}{4}x^2 + 2x$. The coefficients, or degrees of freedom, of B-spline shape functions are the mean values (○) of neighbouring nodal values. At the boundary the nodal values correspond to the B-spline coefficients. Figure 10.3 shows an

[6] Web-splines means weighted extended b-splines. Web-splines are introduced to model a curved domain which is not aligned to a grid. Such an extension of B-splines along the boundary of the domain assures that each finite element shape function has sufficient support for a stable numerical analysis. In the interior of the domain the web-spline approach is based on B-spline finite elements similar to [9].

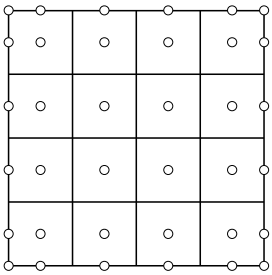

Fig. 10.5: Coordinates for evaluation of coefficients which are assigned to bivariate B-spline functions; the two-dimensional analogy to Figs. 10.3 and 10.4.

adequate approximation of the piecewise linear function by B-splines of order $k = 2$. Another example is shown in Fig. 10.4. Here, the transformation appears less accurate. However, as a global approximation it is reasonable and appropriate to the purpose of coarse grid correction.

The two-dimensional counterpart is illustrated in Figure 10.5. The B-spline coefficients correspond to an evaluation of the bilinear solution at coordinates marked by a circle ($\circ$).

11 BENCHMARK

In [8] a benchmark study of the multigrid method for heterogeneous material samples of up to eight million degrees of freedom is presented. There it turned out that the computation times for convergence increased with an increasing ratio R of Young's moduli between inclusions and matrix, while only ratios $R = 2/1$, $R = 4/1$ and $R = 8/1$ were examined. This article proposes a remedy by the multigrid preconditioned conjugate gradient method. Therefore the multigrid method[7] and the preconditioned version are compared for ratios of up to $R = 200/1$.

In Fig. 11.1 the various iterative methods are compared for $R = 2/1$ with respect to increasing model size. As expected, only for small models the efficiency of the conjugate gradient method is comparable to that of the both multigrid methods. For this low ratio R the multigrid method is more efficient than the multigrid preconditioned conjugate gradient method, but not very significant.

[7] As the convergence criteria, as well as the computational architecture, of the present study is different to the conditions of the former study in [8], a direct comparison to these former results is not given.

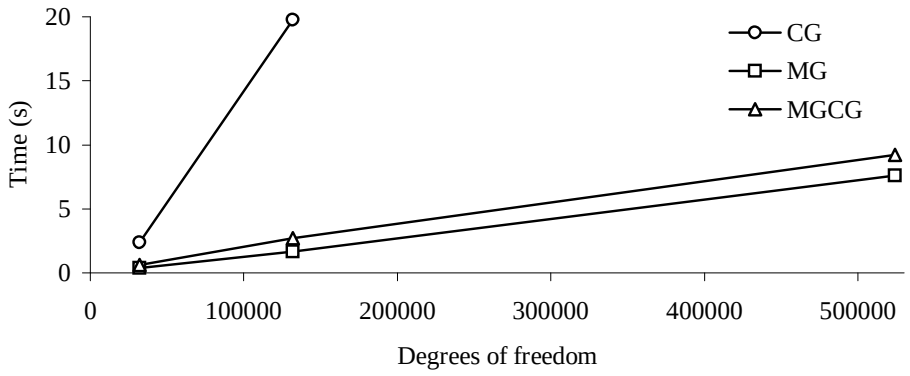

Fig. 11.1: Computation times of conjugate gradient method (CG), multigrid method (MG) and multigrid preconditioned conjugate gradient method (MGCG) in analysis of concrete material (2D). Ratio of Young's modulus of inclusion to matrix is $R = 2/1$. Inclusion volume is 40%. A well-conditioned problem.

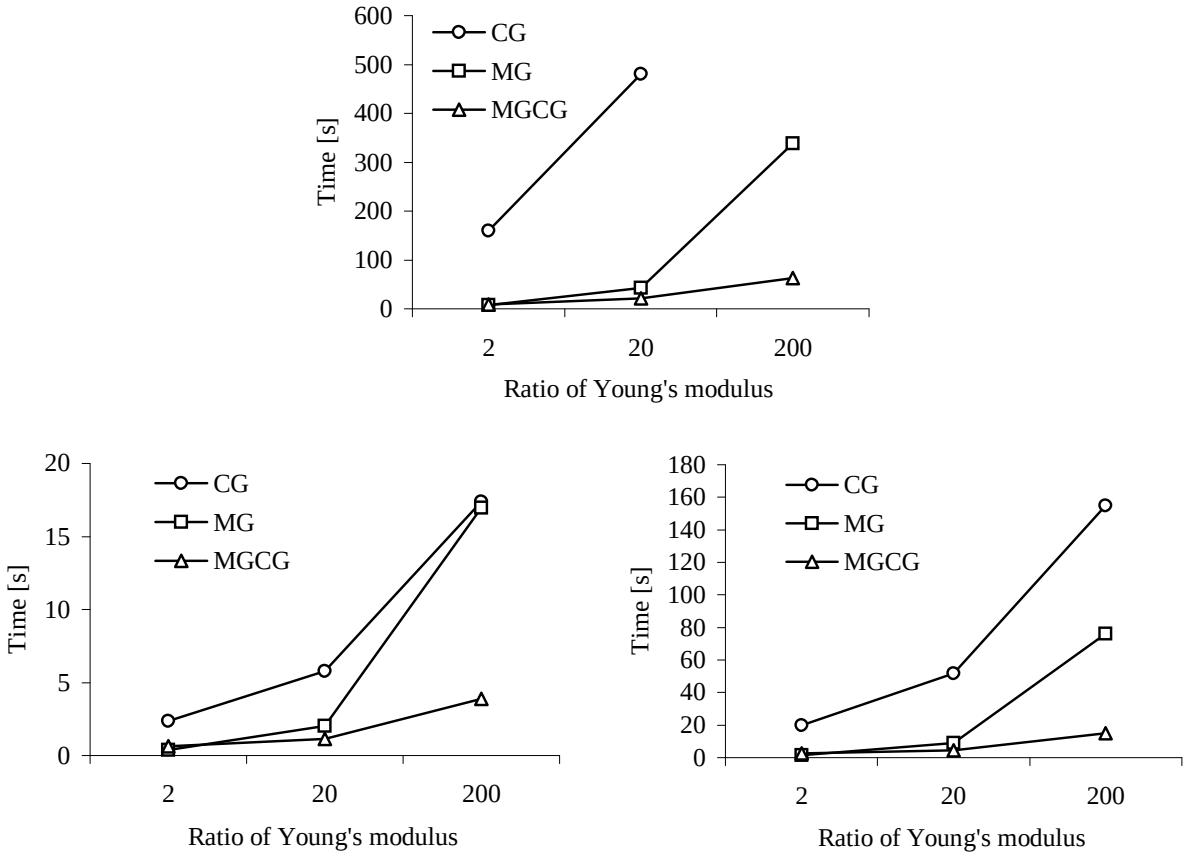

Fig. 11.2: Same problems as in Fig. 11.1, but increase of ratio of Young's modulus in the range $R = 2/1$ to $R = 200/1$. The model size is: 32000 (left), 131000 (right) and 524000 (top) degrees of freedom.

The results with respect to increasing ratio of up to $R = 200/1$ are illustrated in the diagrams of Fig. 11.2. For all model sizes the multigrid preconditioned conjugate gradient method is significantly less sensitive to an increasing ratio R and considerably more efficient than the multigrid method. This is an important improvement with respect to [8]. Such large ratios R may occur in certain heterogeneous materials or in damage analysis [11] to which this article contributes herewith. This first study of the multigrid preconditioned conjugate gradient method is adequate for detecting its relevant positive effect in general. Further more detailed studies are required. Moreover, it will be interesting to examine these solver methods for three-dimensional models.

12 CONCLUSIONS

The article introduces to stationary iterative solver methods in the matrix form and prepares for the later applied index form. For the conjugate gradient method, the multigrid method and multigrid preconditioning of the conjugate gradient method, some relevant fundamentals for an implementation of these methods are provided. The complexity of the general problem is reduced by utilizing certain properties of solid finite elements and defining a clear global numbering system. Based on the discussed index form, element-based global operations lead to a compact and efficient formulation of the heterogeneous finite element problem on orthogonal grids in two and three dimensions. Moreover, similar local schemes are outlined for advanced finite elements of an orthogonal mesh or triangular elements of an irregular mesh. The trans-

fer operators of the multigrid method are adequately derived by the fundamental equations of solid finite elements. Besides, also the transfer of the heterogeneous material to coarser grids is discussed. With regard to B-spline finite elements, an intuitive transfer operator is exemplified for the one-dimensional problem. It is further highlighted that a modified multigrid cycle with balanced computational effort between all grids is successfully applied. As a major conclusion of this article, the multigrid preconditioned conjugate gradient method is essentially more efficient for large and very large ratios of Young's moduli in the material than the multigrid method, which is a relevant improvement with regard to [8] and will be significant for damage analysis [11].

REFERENCES

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[2] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, Englewood Cliffs, 1996.

[3] C. Bayreuther. *Mehrskalenmodelle in der Festkörpermechanik und Kopplung von Mehrgittermethoden mit Homogenisierungsverfahren*. PhD thesis, Universität Stuttgart, Germany, 2004.

[4] M. Gee. *Effiziente Lösungsstrategien in der nichtlinearen Schalenmechanik*. PhD thesis, Universität Stuttgart, Germany, 2004.

[5] W. Hackbusch. *Multi–grid methods and applications*. Springer, Berlin, 1985.

[6] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner, Stuttgart, 1991.

[7] S. Häfner, S. Eckardt, and C. Könke. A geometrical inclusion-matrix model for the finite element analysis of concrete at multiple scales. In K. Gürlebeck, L. Hempel, and C. Könke, editors, *Proc. IKM 2003*, Weimar, Germany, 2003.

[8] S. Häfner, S. Eckardt, T. Luther, and C. Könke. Mesoscale modeling of concrete: Geometry and numerics. *Computers and Structures*, 84(7):450–461, 2006.

[9] S. Häfner, M. Kessel, and C. Könke. Multiphase b-spline finite elements of variable order in the mechanical analysis of heterogenous solids. In K. Gürlebeck and C. Könke, editors, *Proc. IKM 2006*, Weimar, Germany, 2006.

[10] S. Häfner and C. Könke. A multigrid finite element method for the mesoscale analysis of concrete. In P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau, editors, *Proc. ECCOMAS 2004*, Jyväskylä, Finland, 2004.

[11] S. Häfner and C. Könke. Damage simulation of heterogeneous solids by nonlocal formulations on orthogonal grids. In K. Gürlebeck and C. Könke, editors, *Proc. IKM 2006*, Weimar, Germany, 2006.

[12] M. R. Hestenes and E. L. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards Sect.*, 49(5):409–439, 1952.

[13] K. Höllig, U. Reif, and J. Wipper. Multigrid methods with web-splines. *Numerische Mathematik*, 91(2):237–256, 2002.

[14] I. D. Parsons. Parallel adaptive multigrid methods for elasticity, plasticity and eigenvalue problems. In M. Papadrakakis, editor, *Parallel solution methods in computational mechanics*, pages 143 – 180. John Wiley, New York, 1997.

[15] O. Tatebe. The multigrid preconditioned conjugate gradient method. In *Proc. of Sixth Copper Mountain Conference on Multigrid Methods, NASA Conference Publication 3224*, pages 621–634, 1993.

[16] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.

[17] D.M. Young. *Iterative solution of large linear systems*. Academic Press, New York, 1991.

[18] O.C. Zienkiewicz and R.L. Taylor. *The finite element method, Fourth Edition, Volume 1, Basic formulation and linear problems*. McGraw-Hill, UK, 1997.