UNIVERSITÄT KARLSRUHE

# A geometric data structure for parallel finite elements and the application to multigrid methods with block smoothing

Christian Wieners

Preprint Nr. 09/02

Institut für Wissenschaftliches Rechnen
und Mathematische Modellbildung

76128 Karlsruhe

**Anschrift des Verfassers:**

Prof. Dr. Christian Wieners
Institut für Angewandte und Numerische Mathematik
Universität Karlsruhe (TH)
D-76128 Karlsruhe

# A geometric data structure for parallel finite elements and the application to multigrid methods with block smoothing

**Christian Wieners**

Institut für Angewandte und Numerische Mathematik, Universität Karlsruhe, 76 128 Karlsruhe, Germany, www.mathematik.uni-karlsruhe.de e-mail: `wieners@math.uni-karlsruhe.de`

**Abstract** We present a parallel data structure which is directly linked to geometric quantities of an underlying mesh and which is well adapted to the requirements of a general finite element realization. In addition, we define an abstract linear algebra model which supports multigrid methods (extending our previous work in Comp. Vis. Sci. 1 (1997) 27-40). Finally, we apply the parallel multigrid preconditioner to several configurations in linear elasticity and we compute the condition number numerically for different smoothers, resulting in a quantitative evaluation of parallel multigrid performance.

**Key words** parallel computing, finite elements, multigrid methods

## 1 Introduction

Multigrid methods are a well-established class of solution methods for algebraic linear systems arising from discretized partial differential equations (see, e.g., the textbooks [18, 7] for a general overview, and see the special issue [13] for an up-to-date summary on recent developments). Also the parallel implementation of multigrid methods is now well established, and parallel concepts are presented, e.g., in [1, 11, 17, 21, 23, 12, 38]. Now, many fundamental problems in multigrid analysis and also the main principles for efficient parallel multigrid realizations are known. In the last years, the focus in research moved to the application of multigrid methods to a large variety of problem classes; this is still a challenge for, e.g., nonlinear and coupled systems. Moreover we observe that the qualitative multigrid analysis is restricted in most cases to elliptic model problems, and a detailed quantitative analysis is still out of reach.

Here, we focus on parallel multigrid methods on geometry-based data structures that are well adapted to unstructured finite element applications as they are described in [2]. The main new idea is to avoid any global numbering by identifying parallel distributed objects by their geometric position. In particular any degree of freedom of the finite element discretization is linked to its geometric nodal point which directly allows for a parallel identification of interface values by comparing their positions. Within the realization of this data model, we consequently use hash map containers (where the geometric points are used as hash keys), which replace all containers based on doubly-linked lists or balanced trees in [2]. Note that the advantage of hash-based data structures in the context of multigrid methods was first observed in [16].

The paper is organized as follows. In the first part, we summarize the basic principles of the parallel multigrid implementation:

- ▶ We present (in Section 2) the concepts of our new implementation [29] of distributed meshes, based on a non-overlapping decomposition of all cells in the finite element mesh (which is provided by a *load balancing* procedure).
- ▶ This results in an overlapping decomposition of the nodal points. The coordinates of the nodal points are used as a global key for all interface points.
- ▶ On this data structure we define parallel finite elements in Section 3. We require that all solution and correction vectors have the same values at the processor interfaces. This is a parallel consistency requirement.
- ▶ Finite element stiffness matrices, right-hand side vectors, and residuals are only accessed additively via their local contributions on each processor.

This data model is then used for an abstract concept of parallel (geometry-linked) linear algebra (introduced in Section 4), which is well suited for the realization of parallel multigrid methods (Section 5). We then summarize parallel multigrid theory for block smoothing schemes for nested elliptic discretizations in Section 6. We finish with numerical experiments (in Section 7) for the evaluation of the condition number of the parallel multigrid preconditioner in dependence on the block size and on damping factors in the smoothing scheme.

This parallel programming model is designed for maximal flexibility, so that it can easily be applied to many other applications: it is already successfully applied to a geomechanical problem [32] via an element-based interface to an engineering model, to a problem in biomechanics [35], to a triphasic porous media model [33], to infinitesimal plasticity [30,31], to Cosserat plasticity [24], to nonlocal plasticity [25], and to periodic eigenvalue computations of photonic crystals [20].

Nevertheless, our data model is quite restrictive and its design is especially adapted to the requirements of parallel multigrid methods. A more flexible abstract data model (including concepts of generic programming) is presented in [3,4].

## 2 A parallel mesh model

For a transparent (and—last but not least—also debugging-friendly) parallel code it is required to define an abstract programming model. Here, we present a parallel programming model based on the concept of *Distributed Point Objects*, which allows for a realization of a parallel finite element code and a parallel multigrid solver with a minimum of parallel programming overhead. Within this model, all objects are associated with a geometric point which is used as a global identification key. The objects are stored in hash maps using these keys. Note that this requires rounding error tolerant arithmetic for the geometric calculations, which is realized by prescribing a geometric tolerance $\epsilon_{\text{geom}} > 0$ for the comparison of different positions. For simplicity, all quantities will be described in $\mathbb{R}^3$, and lower dimensional computations are embedded into 3-d.

We consider a mesh hierarchy $(\mathcal{C}_j, \mathcal{V}_j, \mathcal{E}_j, \mathcal{F}_j)$ of cells $\mathcal{C}_j$, vertices $\mathcal{V}_j$, edges $\mathcal{E}_j$, and faces $\mathcal{F}_j$ on levels $j = 0, ..., J$. The coarsest level $j = 0$ describes the configuration (including the domain and the boundary condition), and by successive refinement steps we obtain the finest mesh on level $J$, where our finite element problem is defined; the intermediate levels will be used only for the multigrid preconditioner.

Let $c \in \mathcal{C}_j$ be a cell on level $j$, $\mathcal{V}_c \subset \mathbb{R}^3$ be the cell vertices, $\mathcal{E}_c$ be the cell edges, and $\mathcal{F}_c$ be the cell faces. Here, each edge $e \in \mathcal{E}_c$ is represented by a pair $(x_e, y_e) \in \mathcal{V}_c \times \mathcal{V}_c$ and its edge midpoint $z_e \in \mathbb{R}^3$ (which is used in the corresponding hash map container as the associated key). In the same way we use the face midpoint $z_f$ and the cell midpoint $z_c$ as the corresponding keys of $f \in \mathcal{F}_c$ and $c \in \mathcal{C}_j$, respectively. The vertices and midpoints are collected in the set $\mathcal{Z}_c = \mathcal{V}_c \cup \{z_e : e \in \mathcal{E}_c\} \cup \{z_f : f \in \mathcal{F}_c\} \cup \{z_c\}$ of hash keys associated to the cell $c$.

Together, the set of cells $\mathcal{C}_j$ define the set of vertices $\mathcal{V}_j = \bigcup_{c\in\mathcal{C}_j} \mathcal{V}_c$, edges $\mathcal{E}_j = \bigcup_{c\in\mathcal{C}_j} \mathcal{E}_c$, and faces $\mathcal{F}_j = \bigcup_{c\in\mathcal{C}_j} \mathcal{F}_c$. An interior face $f \in \mathcal{F}_j$ is associated to the pair of midpoints $(z_c, z_{c'})$ of the two cells $c, c' \in \mathcal{C}_j$ with $f \in \mathcal{F}_c \cap \mathcal{F}_{c'}$. Introducing the exceptional point $z = \infty$, boundary faces $f \in \mathcal{F}$ are associated to the pair $(z_c, \infty)$.

In parallel, it is required to determine a distribution of the meshes by a load balancing procedure. In principle, this can be done independently on every level, and within an adaptive time-depending simulation it may be advantageous to redistribute the mesh by a dynamic load balancing process [10,14,22]. For simplicity, we describe in this paper only the static configuration with a distribution of the coarsest level and uniform refinement.

On the processor set $\mathcal{P} = \{1, ..., P\}$, a load balancing is given by a function

$$\text{dest}_0 \colon \mathcal{C}_0 \longrightarrow \mathcal{P}\,, \qquad \mathcal{C}_0^p = \{c \in \mathcal{C}_0 \colon \text{dest}_0(c) = p\}\,,$$

defining a disjoint decomposition $\mathcal{C}_0 = \mathcal{C}_0^1 \cup \cdots \cup \mathcal{C}_0^P$. This also defines $\mathcal{V}_0^p = \bigcup_{c\in\mathcal{C}_0^p} \mathcal{V}_c$, $\mathcal{E}_0^p = \bigcup_{c\in\mathcal{C}_0^p} \mathcal{E}_c$, $\mathcal{F}_0^p = \bigcup_{c\in\mathcal{C}_0^p} \mathcal{F}_c$, and $\mathcal{Z}_0^p = \bigcup_{c\in\mathcal{C}_0^p} \mathcal{Z}_c$, resulting in overlapping decompositions of the vertices $\mathcal{V}_0 = \mathcal{V}_0^1 \cup \cdots \cup \mathcal{V}_0^P$, the edges $\mathcal{E}_0 = \mathcal{E}_0^1 \cup \cdots \cup \mathcal{E}_0^P$, the faces $\mathcal{F}_0 = \mathcal{F}_0^1 \cup \cdots \cup \mathcal{F}_0^P$, and the hash keys $\mathcal{Z}_0 = \mathcal{Z}_0^1 \cup \cdots \cup \mathcal{Z}_0^P$.

The overlapping decomposition of the hash keys $\mathcal{Z}_0$ defines a set-valued *partition map*

$$\pi_0 \colon \mathcal{Z}_0 \longrightarrow 2^{\mathcal{P}}\,, \qquad \pi_0(z) = \{p \in \mathcal{P} \colon z \in \mathcal{Z}_0^p\}\,.$$

This partition map (also realized by a hash map container) is constructed on one processor; then, the restrictions $\pi_0^p = \pi_0|_{Z_0^p}$ are sent to the processors $p \in \mathcal{P}$.

In the next step nested refinement can be performed without communication. In the refinement step from level $j-1$ to level $j$ on processor $p \in \mathcal{P}$, a cell $c \in \mathcal{C}_{j-1}^p$ is refined to cells $\mathcal{C}_c$ on level $j$, defining $\mathcal{C}_j^p = \bigcup_{c\in\mathcal{C}_{j-1}^p} \mathcal{C}_c$ and then the vertices $\mathcal{V}_j^p = \bigcup_{c\in\mathcal{C}_j^p} \mathcal{V}_c$. Correspondingly, an edge $e \in \mathcal{E}_{j-1}^p$ is refined to edges $\mathcal{E}_e$ on level $j$ defining $\mathcal{E}_j^p = \bigcup_{e\in\mathcal{E}_{j-1}^p} \mathcal{E}_e$, and a face $f \in \mathcal{F}_{j-1}^p$ is refined to faces $\mathcal{F}_e$ on level $j$ defining $\mathcal{F}_j^p = \bigcup_{f\in\mathcal{F}_{j-1}^p} \mathcal{F}_f$. In case of uniform 3-d refinement, a coarse cell $c$ is refined into 8 cells $\mathcal{C}_c$, a coarse face $f$ is refined into 4 faces $\mathcal{F}_f$, and a coarse edge $e$ is refined into two edges $\mathcal{E}_e$.

Locally, on every processor this defines the partition map $\pi_j^p$ by $\pi_j^p(z) = \pi_{j-1}^p(z)$ for $z \in \mathcal{V}_{j-1} \subset \mathcal{V}_j$, $\pi_j^p(z_{e'}) = \pi_{j-1}^p(z_e)$ for $e' \in \mathcal{E}_e$ and $e \in \mathcal{E}_{j-1}$, $\pi_j^p(z_{f'}) = \pi_{j-1}^p(z_f)$ for $f' \in \mathcal{F}_f$ and $f \in \mathcal{F}_{j-1}$, and $\pi_j^p(z_c) = \{p\}$ for $c \in \mathcal{C}_j^p$. This defines the partition map for all keys in $Z_j^p = \mathcal{V}_j^p \cup \{z_e \colon e \in \mathcal{E}_j^p\} \cup \{z_f \colon f \in \mathcal{F}_j^p\} \cup \{z_c \colon c \in \mathcal{C}_j^p\}$.

Note that this construction of local partition maps does not require any parallel communication within the nested refinement procedure. In addition, it guarantees local consistency, i.e.,

$$\pi_j^p(z) = \pi_j^q(z), \qquad z \in \mathcal{Z}_j^p \cap \mathcal{Z}_j^q\,.$$

Following [22], this concept for a parallel data structure can be directly extended to parallel adaptive refinement together with a parallel redistribution by introducing (and communicating) copy elements. Note that copy elements are also used for the parallel realization of an overlapping domain decomposition method in [34].

## 3 Parallel finite elements

A finite element mesh on level $j$ is given by a decomposition $\bar{\Omega}_j = \bigcup_{c\in\mathcal{C}_j} \bar{\Omega}_c$, where $\Omega_c \subset \mathbb{R}^3$ is the interior of the cell such that $\bar{\Omega}_c = \operatorname{conv}\{\mathcal{V}_c\}$. The non-overlapping cell decomposition defines a non-overlapping domain decomposition $\Omega_j^1 \cup \dots \cup \Omega_j^P$ of $\Omega_j$ with sub-domains $\bar{\Omega}_j^p = \bigcup_{c\in\mathcal{C}_j^p} \bar{\Omega}_c$ and the skeleton

$$\Gamma_j = \bigcup_{p<q} \Gamma_j^{pq}\,, \qquad \Gamma_j^{pq} = \partial\Omega_j^p \cap \partial\Omega_j^q\,.$$

Let $V_j = \operatorname{span}\{\phi_{j,i} \colon i \in \mathcal{I}_j\}$ be a finite element space with basis $\phi_{j,i}$, where $\mathcal{I}_j$ is the corresponding index set. Each basis function $\phi_{j,i}$ is associated with a dual function $\phi'_{j,i} \in V'_j$ such that $\langle \phi'_{j,i}, \phi_{j,k}\rangle = \delta_{ik}$. To every index $i \in \mathcal{I}_j$ we assign a nodal point $z_i \in \bar{\Omega}_j$ and—in case of a system of equations—a component $k_i$.

Let $\mathcal{N}_j = \{z_i \colon i \in \mathcal{I}_j\}$ be the set of nodal points. In our programming model we assume $\mathcal{N}_j \subset \mathcal{Z}_j$, which automatically defines the partition map also on $\mathcal{I}_j$: we set $\pi_j(i) = \pi_j(z_i)$. Let $\mathcal{I}_j = \mathcal{I}_j^1 \cup \dots \cup \mathcal{I}_j^P$ be the overlapping decomposition of the index set with $\mathcal{I}_j^p = \{i \in \mathcal{I}_j \colon p \in \pi_j(i)\}$, and set $N_j^p = \#\mathcal{I}_j^p$. Note that in parallel an independent numbering for each local index set $\mathcal{I}_j^p$ can be used. No global numbering is required, since the realization of the same index on different processors can be uniquely identified by the pair $(z_i, k_i)$.

A finite element function $v_j = \sum_{i\in\mathcal{I}_j} v_{j,i}\phi_{j,i} \in V_j$ is represented uniquely by its coefficient vector $(v_{j,i})_{i\in\mathcal{I}_j}$ with $v_{j,i} = \langle \phi'_{j,i}, v_j\rangle$. Analogously, a discrete functional $f_j = \sum_{i\in\mathcal{I}_j} f_{j,i}\phi'_{j,i} \in V'_j$ is represented uniquely by its coefficient vector $(f_{j,i})_{i\in\mathcal{I}_j}$ with $f_{j,i} = \langle f_j, \phi_{j,i}\rangle$.

The main idea of parallel finite elements is to distinguish clearly the parallel vector representations of finite element functions and discrete functionals. Thus, we introduce two different vector representations.

### *Consistent vector representation*

The coefficient vector $(v_{j,i})_{i\in\mathcal{I}_j}$ of a finite element function $v_j \in V_j$ is represented in parallel by its local restrictions $\underline{v}_j^p = (v_{j,i})_{i\in\mathcal{I}_j^p} \in \mathbb{R}^{N_j^p}$ which defines a mapping

$$\begin{aligned} E_j \colon V_j &\longrightarrow \underline{V}_j^{\mathcal{P}} := \prod_{p\in\mathcal{P}} \mathbb{R}^{N_j^p} \\ v_j &\longmapsto \underline{v}_j = (\underline{v}_j^p)_{p\in\mathcal{P}} \end{aligned}$$

to distributed vectors in the corresponding Euclidean product space. Moreover, the coefficient vector of a finite element function is *consistent*, i.e., the coefficients coincide on the processor interfaces. Thus, we define the constrained product space

$$\underline{V}_j = \Big\{ \underline{v}_j \in \underline{V}_j^{\mathcal{P}} : v_{j,i}^p = v_{j,i}^q \text{ for } q \in \pi^p(i) \Big\}\,.$$

We have $\underline{V}_j = E_j(V_j)$, and for $\underline{v}_j \in \underline{V}_j$ a unique finite element function $v_j \in V_j$ with $E_j(v_j) = \underline{v}_j$ is well-defined.

### *Additive vector representation*

In parallel, the assembling of right-hand side vectors (and also stiffness matrices) usually is done only on the corresponding sub-domains $\Omega_j^p$. This directly gives parallel local representations $\underline{f}_j^p = (f_{j,i})_{i\in\mathcal{I}_j^p} \in \mathbb{R}^{N_j^p}$. In order to obtain the full coefficient vector $(f_{j,i})_{i\in\mathcal{I}_j}$, parallel communication is required for the collection of the local contributions. Obviously, the additive decomposition

$$f_{j,i} = \sum_{p\in\pi(i)} f_{j,i}^p \tag{1}$$

depends on the load balancing.

Thus, we use a non-unique representation of functionals in $V'_j$ in the quotient space

$$\underline{V}'_j = \underline{V}_j^{\mathcal{P}} / \underline{V}_j^0$$

factoring out the kernel of the collection operation (1)

$$\underline{V}_j^0 = \{ \underline{f}_j \in \underline{V}_j^{\mathcal{P}} \colon \sum_{p\in\pi(i)} f_{j,i}^p = 0,\ i \in \mathcal{I}_j \}\,.$$

Note that the parallel Euclidean inner product

$$\underline{v}_j \cdot \underline{f}_j = \sum_{p\in\mathcal{P}} \underline{v}_j^p \cdot \underline{f}_j^p\,, \qquad \underline{v}_j \in \underline{V}_j, \underline{f}_j \in \underline{V}'_j$$

is well-defined in $\underline{V}_j \times \underline{V}'_j$. Moreover, this characterizes the kernel space as the polar space

$$\underline{V}_j^0 = \{ \underline{v}_j \in \underline{V}_j^{\mathcal{P}} : \underline{v}_j \cdot \underline{w}_j = 0,\ \underline{w}_j \in \underline{V}_j \}\,,$$

where the parallel Euclidean inner product is used as the dual pairing. With respect to this dual pairing we obtain the adjoint operator to the embedding operator $E_j$. This mapping is defined by

$$\begin{aligned} E'_j \colon \underline{V}'_j &\longrightarrow V'_j \\ \underline{f}_j &\longmapsto f_j = \sum_{p\in\mathcal{P}} \sum_{i\in\mathcal{I}_j^p} f_{j,i}^p \phi'_{j,i} \end{aligned}$$

and assigns a functional to a distributed vector. Note that this mapping is not one-to-one. In order to obtain a unique parallel representation of functionals we assign $p = \min \pi_j(i)$ as the master processor for every index, and defining the index subset

$$\widetilde{\mathcal{I}}_j^p = \{ i \in \mathcal{I}_j^p \colon p = \min \pi_j(i) \Big\}$$

yields a non-overlapping decomposition of the index set $\mathcal{I}_j = \widetilde{\mathcal{I}}_j^1 \cup \dots \cup \widetilde{\mathcal{I}}_j^P$. This defines the subspace of $\underline{V}'_j$

$$\begin{aligned} \widetilde{\underline{V}}'_j &= \Big\{ \underline{f}_j = (\underline{f}_j^p)_{p\in\mathcal{P}} \in \underline{V}'_j \colon f_{j,i}^p = 0 \text{ for } p \neq \min \pi_j(i) \Big\} \\ &= \Big\{ \underline{f}_j = (\underline{f}_j^p)_{p\in\mathcal{P}} \in \underline{V}'_j \colon f_{j,i}^p = 0 \text{ for } i \notin \widetilde{\mathcal{I}}_j^p \Big\}\,. \end{aligned}$$

Note that $E'_j$ restricted to $\widetilde{\underline{V}}'_j$ is one-to-one.

*Parallel Euclidean norms in $\underline{V}_j$ and $\underline{V}_j'$*

In the parallel Euclidean spaces it is recommended to use only norms which are induced by norms of the finite element spaces, since the parallel Euclidean norm of the product space $\underline{V}_j^{\mathcal{P}}$

$$\|\underline{v}_j\|_{\underline{V}_j^{\mathcal{P}}} = \sqrt{\sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{I}^p} |\underline{v}_{j,i}^p|^2}\,, \qquad \underline{v}_j \in \underline{V}_j^{\mathcal{P}}$$

is mesh-dependent and not invariant of the load balancing. Its absolute order of magnitude has no physical interpretation. On the other hand the evaluation of induced norms is quite involved, so that for the convergence control of iterative methods the error reduction with respect to the Euclidean norm is quite efficient (and up to a mesh-dependent norm equivalence factor also reliable).

For consistent vectors in $\underline{V}_j$ we define the norm

$$\|\underline{v}_j\|_{\underline{V}_j} = \sqrt{\sum_{p\in\mathcal{P}}\sum_{i\in\widetilde{\mathcal{I}}^p} |\underline{v}_{j,i}^p|^2}\,, \qquad \underline{v}_j \in \underline{V}_j$$

which is obviously invariant of the load balancing. For additive vectors in $\underline{V}_j'$ the evaluation of the dual norm

$$\|\underline{f}_j\|_{\underline{V}_j'} = \sup_{\|\underline{v}_j\|_{\underline{V}_j}=1} \underline{f}_j \cdot \underline{v}_j\,, \qquad \underline{f}_j \in \underline{V}_j'$$

requires parallel communication: compute the unique representative $\widetilde{\underline{f}}_j \in \widetilde{\underline{V}}_j'$ of $\underline{f}_j$; then, we have

$$\|\underline{f}_j\|_{\underline{V}_j'} = \|\widetilde{\underline{f}}_j\|_{\underline{V}_j'} = \|\widetilde{\underline{f}}_j\|_{\underline{V}_j^{\mathcal{P}}} = \sqrt{\sum_{p\in\mathcal{P}} \|\widetilde{\underline{f}}_j^p\|^2}\,.$$

*Additive operator representation*

Let $V$ be a Hilbert space, $a(\cdot,\cdot)\colon V\times V \longrightarrow \mathbb{R}$ an elliptic bilinear form, and let $A\colon V \longrightarrow V'$ be the operator defined by $\langle Av, w\rangle = a(v,w)$ for $v,w\in V$. For a finite element space $V_j \subset V$ let $A_j\colon V_j \longrightarrow V_j'$ be the Galerkin approximation defined by

$$\langle A_j v_j, w_j\rangle = a(v_j, w_j)\,, \qquad v_j, w_j \in V_j\,.$$

The matrix representation $\underline{A}_j\colon \underline{V}_j \longrightarrow \underline{V}_j'$ satisfying $A_j = E_j' \circ \underline{A}_j \circ E_j$ depends on a suitable decomposition of the operator. Therefore, we assume that the bilinear form $a(\cdot,\cdot)$ allows for a cell-based additive decomposition

$$a(v,w) = \sum_{c\in\mathcal{C}_j} a_c(v,w)\,, \qquad v,w\in V\,,$$

and we assume that the finite element nodal basis functions $\phi_{j,i}$ have local support such that $a_c(\phi_{j,i},\phi_{j,k}) \neq 0$ only if $z_i, z_k \in \mathcal{N}_c$ with $\mathcal{N}_c = \mathcal{N}_j \cap \bar{\Omega}_c$. This allows for the parallel assembling of the local stiffness matrices $\underline{A}_j^p = (A_{j,i,k})_{i,k\in\mathcal{I}_j^p}$ with

$$A_{j,i,k}^p = \sum_{c\in\mathcal{C}_j^p} a_c(\phi_{j,i},\phi_{j,k})\,, \tag{2}$$

which together gives the parallel additive matrix representation $\underline{A}_j = (\underline{A}_j^p)_{p\in\mathcal{P}}$ of the discrete operator $A_j$, i.e., we have

$$A_j\phi_{j,i} = \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{I}_j^p} A_{j,i,k}^p \phi_{j,k}'\,.$$

Note that within this parallel programming model for a consistent vector $\underline{v}_j \in \underline{V}_j$ the operation

$$\underline{f}_j = \underline{A}_j\underline{v}_j = (\underline{A}_j^p\underline{v}_j^p)_{p\in\mathcal{P}} \in \underline{V}_j'$$

yields the additive result without parallel communication.

*Parallel assembling*

We illustrate the parallel assembling procedure for linear elasticity with $V = H^1(\Omega,\mathbb{R}^3)$ and

$$a(v,w) = \int_\Omega \boldsymbol{\varepsilon}(v) : \mathbb{C} : \boldsymbol{\varepsilon}(w)\,dx\,,$$

where $\boldsymbol{\varepsilon}(v) = \mathrm{sym}(\nabla v)$ and $\mathbb{C} : \boldsymbol{\varepsilon} = 2\mu\boldsymbol{\varepsilon} + \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I}$. Furthermore, let $f\in V'$ with

$$f(v) = \int_\Omega b\cdot v\,dx + \int_{\partial\Omega} g\cdot v\,da$$

be a load functional, and let $u^D$ be Dirichlet data on a part of the boundary $\Gamma \subset \partial\Omega$.

The continuous problem in linear elasticity reads as follows: find $u\in V$ such that $u = u^D$ on $\Gamma$ and

$$a(u,v) = f(v)$$

for all $v\in V$ with $v = 0$ on $\Gamma$.

We use lowest-order $H^1$-conforming elements, where we have the nodal points $z_i \in \mathcal{V}_j$ and components $k_i \in \{1,2,3\}$ (for other applications we also use edge or face degrees of freedom). Let $\phi_z \in C(\bar{\Omega})$ be the standard nodal basis with $\phi_z(z) = 1$ and $\phi_z(y) = 0$ for $z,y\in\mathcal{V}_j$, $z\neq y$, and let $\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3 \in \mathbb{R}^3$ be the Euclidean basis. This defines the finite element space $V_j = \mathrm{span}\{\phi_{j,i}\}$ with $\phi_{j,i} = \phi_{z_i}\mathbf{e}_{k_i}$. The associated dual basis in $V_j'$ is given by $\langle\phi_{j,i}', v\rangle = v_{k_i}(z_i)$.

Obviously, cell-based assembling is possible by defining

$$a_c(v,w) = \int_{\Omega_c} \boldsymbol{\varepsilon}(v) : \mathbb{C} : \boldsymbol{\varepsilon}(w)\,dx$$

and

$$f_c(v) = \int_{\Omega_c} b\cdot v\,dx + \int_{\partial\Omega\cap\partial\Omega_c} g\cdot v\,da\,.$$

In the assembling process, we have to compute on every processor the local stiffness matrix $\underline{A}_j^p$ by (2), the local right-hand side $\underline{f}_j^p = (\underline{f}_{j,i}^p)$ by

$$\underline{f}_{j,i}^p = \sum_{c\in\mathcal{C}_j^p} f_c(\phi_{j,i})\,,$$

and the local set of Dirichlet indices

$$\mathcal{D}_j^p = \{i \in \mathcal{I}_j^p \colon z_i \in \Gamma\}$$

for essential boundary conditions (note that in general parallel communication is required to guarantee parallel consistency of the Dirichlet indices).

The discrete problem in linear elasticity reads as follows: find $u_j \in V_j$ such that $u_j(z) = u^D(z)$ for all $z \in \Gamma \cap \mathcal{V}_j$ and

$$a(u_j, \phi_{j,i}) = f(\phi_{j,i})$$

for all $i \in \mathcal{I}_j^p \setminus \mathcal{D}_j^p$ (i.e., $z_i \in \bar{\Omega} \setminus \Gamma$).

There are several possibilities to incorporate essential boundary conditions into the discrete linear system. Here, the parallel discrete solution $\underline{v}_j \in \underline{V}_j$ is computed as follows. First, we modify $\underline{A}_j^p$ and $\underline{f}_j^p$ locally for the Dirichlet indices by setting

$$A_{j,i,i}^p = 1, \quad A_{j,i,m}^p = 0 \quad (m \neq i), \quad f_{j,i}^p = u_{k_i}^D(z_i)$$

for all $i \in \mathcal{D}_j^p$. Then, solve the linear problem

$$\underline{A}_j \underline{u}_j = \underline{f}_j\,,$$

i.e., find $\underline{u}_j \in \underline{V}_j$ such that

$$(\underline{A}_j^p \underline{u}_j^p - \underline{f}_j^p)_{p \in \mathcal{P}} \in \underline{V}_j^0$$

is satisfied for the parallel defect. Checking of this condition requires parallel communication. In our programming model, this communication is included in extended operations of parallel linear algebra.

## 4 Parallel linear algebra

The basic idea for a reliable, transparent and efficient implementation of parallel algorithms is the formulation of all algorithms in function spaces (i.e., in $V$ and $V'$) and the consequent use of consistent vectors in $\underline{V}_j$ for finite element functions, and additive vectors in $\underline{V}_j'$ for discrete functionals. This automatically yields parallel algorithms, if two parallel operations for changing the vector representation are included:

a) A unique additive representation is obtained by

$$\text{collect} \colon \underline{V}_j' \longrightarrow \underline{\widetilde{V}}_j'$$

defined by

$$\text{collect}(\underline{f}_j)_i^p = \begin{cases} \sum\limits_{q \in \pi(i)} f_{j,i}^q & p = \min \pi(i), \\ 0 & \text{else.} \end{cases}$$

b) A consistent result of local corrections is obtained by

$$\text{accumulate} \colon \underline{V}_j^{\mathcal{P}} \longrightarrow \underline{V}_j$$

defined by

$$\text{accumulate}(\underline{c}_j)_i^p = \sum_{q \in \pi(i)} c_{j,i}^q\,.$$

These two parallel operations require only local communication with the neighboring processors

$$\mathcal{P}_{j,p} = \bigcup_{p \in \pi_j(i)} \pi_j(i)\,.$$

In general, the number of processors in $\mathcal{P}_{j,p}$ is small and should be bounded independently of the total number of processors $P$. In our parallel linear algebra, global communication is required only for inner products.

The parallel data exchange in the collect and accumulate routine is realized by identifying the indices by their nodal point. We illustrate this procedure for the accumulation of a distributed vector $\underline{c} = (\underline{c}^p) \in \underline{V}_j^{\mathcal{P}}$ (see Fig. 1 for the source code of the realization):

S1. On processor $p$ fill exchange buffers

$$\mathcal{B}_{p,q} = \{(c_{j,i}, z_i, k_i) \colon i \in \mathcal{I}_j^p,\ q \in \pi^p(i)\}$$

for all $q \in \mathcal{P}_{j,p}$.

S2. Exchange the buffers $\mathcal{B}_{p,q}$ for $(p,q)$ where $\mathcal{B}_{p,q}$ is non-empty, i.e., on processor $p$ send $\mathcal{B}_{p,q}$ to all processors $q \in \mathcal{P}_{j,p}$. Then, on processor $q$, receive all buffers $\mathcal{B}_{p,q}$ for $p \in \mathcal{P}_{j,q}$.

S3. On processor $q$, read the buffer content $(d,z,k) \in \mathcal{B}_{p,q}$ for all $p$, find $i \in \mathcal{I}_j^q$ with $(z_i,k_i) = (z,k)$ and set $c_{j,i}^q := c_{j,i}^q + d$.

The efficiency of this algorithm relies on the efficient index determination in S3. In our code this is realized with a hash table which maps every nodal point $z \in \mathcal{N}_j^p$ to the associated index $i \in \mathcal{I}_j^p$ with $z_i = z$ and $k_i = 1$.

### *Parallel operators*

We consider operators on one level and between two successive levels. Together, we distinguish four types of parallel operators.

*Discrete (differential) operators* $A_j \colon V_j \longrightarrow V_j'$ are represented additively by parallel distributed matrices

$$\underline{A}_j \colon \underline{V}_j \longrightarrow \underline{\widetilde{V}}_j'\,,$$

where the operation is defined by

$$\underline{A}_j\,\underline{v} = \text{collect}(\underline{A}_j^p \underline{v}^p)\,, \qquad \underline{v} \in \underline{V}_j\,.$$

Here, $\underline{A}_j^p \in \mathbb{R}^{N_j^p \times N_j^p}$ is represented locally by a (sparse) matrix, and the matrix-vector products $\underline{A}_j^p \underline{v}^p$ can be computed independently, followed by the communication of the collect routine.

Of course, since the finite element stiffness matrix is sparse, we use an appropriate storage format for $\underline{A}_j^p$.

*Preconditioners* $B_j \colon V_j' \longrightarrow V_j$ are represented also by parallel distributed operators

$$\underline{B}_j \colon \underline{V}_j' \longrightarrow \underline{V}_j\,,$$

but now the operation is defined by

$$\underline{B}_j\,\underline{f} = \text{accumulate}(\underline{B}_j^p \underline{f}^p)\,, \qquad \underline{f} \in \underline{V}_j'\,.$$

```
void AccumulateParallel (Vector& u) {
    ExchangeBuffer& E=u.AccumulateParallelBuffer();
    for (procset p=u.procsets();
         p!=u.procsets_end(); ++p) {
        int i = u.Id(p.z());
        for (int j=0; j<p.size(); ++j) {
            int q = p[j];
            if (q == my_proc) continue;
            E.Send(q) << p.z();
            for (int k=0; k<u.Dof(i); ++k)
                E.Send(q) << u(i,k);
        }
    }
    E.Communicate();
    for (short q=0; q<procs.size(); ++q) {
        while (E.Receive(q).size()
               < E.ReceiveSize(q)) {
            Point z; E.Receive(q) >> z;
            int i = u.Id(z);
            for (int k=0; k<u.Dof(i); ++k) {
                Scalar a; E.Receive(q) >> a;
                u(i,k) += a;
            }
        }
    }
}
```

**Fig. 1** Source code for the parallel accumulation routine. Here, `u(i,k)` returns the degree of freedom of $\underline{u}_j$ associated to the nodal point $z = z_i$ and the component $k$. Internally in `u.Id(z)`, on every processor a hash table provides a fast access from the nodal point $z$ to its index $i$.

Here, the distributed operators $\underline{B}_j^p$ can be represented locally by a matrix, but in general only the application to a vector $\underline{B}_j^p \underline{v}^p$ is defined (e.g., for a Gauss-Seidel method, using the matrix $\underline{A}_j$). This is done independently on every processor, followed by the communication of the accumulate routine which is required to obtain a consistent result.

Within the symmetric multigrid method, we also need the adjoint preconditioner $B_j' \colon V_j' \longrightarrow V_j$ which is represented by transposed operations $\underline{B}_j^T \colon \underline{V}_j' \longrightarrow \underline{V}_j$ defined by

$$\underline{B}_j^T = \Big(\text{accumulate} \circ (\underline{B}_j^p) \circ \text{collect}\Big)^T \\ = \text{collect}^T \circ (\underline{B}_j^p)^T \circ \text{accumulate}^T ,$$

where we have $\text{accumulate}^T = \text{accumulate}$, and the adjoint collect routine

$$\text{collect}^T \colon \underline{V}_j^{\mathcal{P}} \longrightarrow \underline{V}_j$$

is given by

$$\text{collect}^T(\underline{c}_j)_i^p = c_{j,i}^{\min \pi(i)} ,$$

i.e., for every index $i$ the corresponding master processor $\min \pi(i)$ sends its value to all $p \in \pi(i)$.

Note that in general the construction of the local preconditioners $B_j^p$ requires communication. They rely on the accumulated matrix $\text{accumulate}(\underline{A}_j^p)$ defined by

$$\text{accumulate}(\underline{A}_j^p)_{i,k} = \sum_{q \in \pi(i) \cap \pi(k)} A_{j,i,k}^q .$$

The *prolongation* $I_j \colon V_{j-1} \longrightarrow V_j$ is represented by parallel distributed prolongation matrices

$$\underline{I}_j \colon \underline{V}_{j-1} \longrightarrow \underline{V}_j ,$$

and the operation is defined by

$$\underline{I}_j\, \underline{v}_{j-1} = (\underline{I}_j^p \underline{v}_{j-1}^p) , \quad \underline{v} \in \underline{V}_{j-1} ,$$

where now $\underline{I}_j^p = (I_{j,i,k}^p) \in \mathbb{R}^{N_j^p \times N_{j-1}^p}$ is a rectangular matrix.
We assume that this procedure gives a consistent result without any communication. This requires horizontal consistency of the partition maps, i.e. $\pi_{j-1}(k) \subset \pi_j(i)$ for all $(i,k) \in \mathcal{I}_j \times \mathcal{I}_{j-1}$ with $I_{j,i,k} \neq 0$. Note that for standard finite elements and the nested refinement described above this condition is satisfied. In more general situations this requires to enlarge the processor set on the coarser mesh such that

$$\pi_{j-1}(k) \supset \bigcup_{I_{j,i,k} \neq 0} \pi_j(i) .$$

This extension of the processor maps is realized, e.g., in [2] for locally refined meshes and different load balancing on different levels by introducing copy elements (a more general concept including also copy indices is introduced there). Here, we use a full overlap and set $\pi_0(i) = \mathcal{P}$ for all $i \in \mathcal{I}_0$ if the coarsest mesh is reasonably small so that the coarse mesh problem can be solved with a direct solver.

The *restriction* $I_j' \colon V_j' \longrightarrow V_{j-1}'$ is represented by the transposed prolongation matrices

$$\underline{I}_j^T \colon \underline{V}_j' \longrightarrow \underline{\widetilde{V}}_{j-1}' ,$$

and the operation is defined by $\underline{I}_j^T\, \underline{f} = \text{collect}\,\big((I_j^p)^T f^p\big)$ for $\underline{f} \in \underline{V}_j'$.

## 5 Parallel multigrid methods

Based on the parallel operations described in the previous section we can now define the parallel multigrid preconditioner $\underline{B}_J^{\text{mg}}$. This is done recursively, starting with a solver $\underline{B}_0^{\text{mg}} \approx \underline{A}_0^{-1}$ on the coarse level. For small coarse problems this is done by collecting the matrix $\underline{A}_0$ on one processor, otherwise we use a Krylov method with a simple preconditioner, and we solve the coarse problem only approximately. For $j = 1, ..., J$, the multigrid preconditioner is a combination of smoothing $\underline{R}_j \colon \underline{V}_j' \longrightarrow \underline{V}_j$ and the coarse grid correction (using $\underline{B}_{j-1}^{\text{mg}}$). More precisely, the symmetric multigrid V-cycle consists of the successive applications of $M$ preconditioning steps with the smoother $\underline{R}_j$, followed by a preconditioning step with $\underline{I}_j \underline{B}_{j-1}^{\text{mg}} \underline{I}_j^T$, and finally again $M$ preconditioning steps

```
void Cycle (int j, Vector& u, Vector& f) {
    if (j==0) {
        u = B_0 * f;
        return;
    }
    for (int m=0; m<M; ++m) {
        w = R[j] * f;
        u += w;
        f -= A[j] * w;
    }
    d = f * I[j];
    c = 0;
    Cycle(j-1,c,d);
    w = I[j] * c;
    u += w;
    f -= A[j] * w;
    for (int m=0; m<M; ++m) {
        w = f * R[j];
        u += w;
        f -= A[j] * w;
    }
}
```

**Fig. 2** Source code for the parallel multigrid V-cycle, i.e., for a given additive vector $\underline{f}_j \in \underline{V}'_j$, the consistent multigrid correction $\underline{u}_j = \underline{B}_j^{\mathrm{mg}} \underline{f}_j \in \underline{V}_j$ is computed by `Cycle(j,u,f)`. In this code, `w` is an auxiliary vector in $\underline{V}_j$, `c` is a vector in $\underline{V}_{j-1}$, and `d` is a vector in $\underline{V}'_j$. The smoother is given by `R[j]` and the prolongation is given by `I[j]`. The restriction $\underline{I}_j^T \underline{f}$ is coded by `f * I[j]`, which applies the transposed prolongation matrix, and `f * R[j]` applies the transposed smoother.

with the transposed smoother $\underline{R}_j^T$ (see Fig. 2 for the realization of this cycle).

The corresponding error propagation matrix is defined by

$$\mathrm{id} - \underline{B}_j^{\mathrm{mg}} \underline{A}_j = \\ (\mathrm{id} - \underline{R}_j^T \underline{A}_j)^M (\mathrm{id} - \underline{I}_j \underline{B}_{j-1}^{\mathrm{mg}} \underline{I}_j^T \underline{A}_j)(\mathrm{id} - \underline{R}_j \underline{A}_j)^M \,.$$

Here, we use the symmetric variant since in our examples (linear elasticity) the operator $\underline{A}_j$ is symmetric and positive definite, so that we can use the multigrid preconditioner within a conjugate gradient method (see Fig. 3).

## *Condition number of the preconditioner*

The efficiency of the preconditioner within the cg method can be estimated by the spectral condition number of $\underline{B}_J \underline{A}_J$. Since this operator is self-adjoint in $\underline{V}_J$ with respect to the energy inner product

$$a_J(\underline{v}_J, \underline{w}_J) = \underline{v}_J \cdot \underline{A}_J \underline{w}_J \,, \tag{3}$$

we can approximate the condition number

$$\kappa(\underline{B}_J \underline{A}_J) = \frac{\lambda_{\max}(\underline{B}_J^{\mathrm{mg}} \underline{A}_J)}{\lambda_{\min}(\underline{B}_J^{\mathrm{mg}} \underline{A}_J)} \tag{4}$$

by the Lanczos method, cf. [15, Chap. 9]: starting from a given residual vector $\underline{r}_J \in \underline{V}'_J$, an orthogonal basis $\underline{v}_J^0, ..., \underline{v}_J^K$ (with respect to the inner product (3)) of

```
void CG (Vector& u, const Operator& A,
         const Operator& B, Vector& r) {
    double d = r.norm();
    double eps = Eps + Red * d;
    p = 0;
    double rho_0 = 1;
    for (iter=0; iter<max_iter; ++iter) {
        if (d < eps) break;
        c = B * r;
        double rho_1 = r * c;
        p *= (rho_1 / rho_0);
        p += c;
        rho_0 = rho_1;
        t = A * p;
        double alpha = rho_0 / (p * t);
        u += alpha * p;
        r -= alpha * t;
        d = r.norm();
    }
}
```

**Fig. 3** Source code for the parallel preconditioned conjugate gradient method, applied to the residual $\underline{r}_J = \underline{f}_J - \underline{A}_J \underline{u}_J^0$, where $\underline{u}_J^0$ is the start iterate (in this code, `c` and `p` are auxiliary vectors in $\underline{V}_j$, and `t` is a vector in $\underline{V}'_j$). Note that the parallel linear algebra guarantees the parallel consistency of this algorithm, so that no further parallel extensions are required.

Krylov space

$$\{\underline{B}_J \underline{r}_J, \underline{B}_J \underline{A}_J \underline{B}_J \underline{r}_J, ..., (\underline{B}_J \underline{A}_J)^K \underline{B}_J \underline{r}_J\}$$

is computed. Then, the Galerkin approximation $H \in \mathbb{R}^{K+1,K+1}$ of $\underline{B}_J \underline{A}_J$ with respect to this basis has Hessenberg form, and—in the symmetric case—is tridiagonal. See Fig. 4 for a prototype implementation of the Lanczos method (in our applications this can be integrated into the cg method).

Based on the condition number we can estimate the convergence of the preconditioned cg method: Let $\underline{u}_J$ be the solution of $\underline{A}_J \underline{u}_J = \underline{f}_J$, and let $\underline{u}_J^0$ be the start iterate. Then, we have in iteration step $k$

$$\|\underline{u}_J - \underline{u}_J^k\|_{\underline{A}_J} \le 2 \Big( \frac{\sqrt{\kappa(\underline{B}_J \underline{A}_J)} - 1}{\sqrt{\kappa(\underline{B}_J \underline{A}_J)} + 1} \Big)^k \, \|\underline{u}_J - \underline{u}_J^0\|_{\underline{A}_J}$$

with respect to the energy norm, cf. [19, Th. 9.4.12].

## *Parallel smoothing*

In our examples, we use in parallel an additive smoothing scheme $\underline{R}_j$ with (possibly overlapping) block-Gauss-Seidel relaxations on every processor, i.e., we have

$$\underline{R}_j = \theta \sum_{p \in \mathcal{P}} \underline{R}_j^p$$

with a damping factor $\theta \in (0, 1]$. The local parts $\underline{R}_j^p$ rely on an index decomposition $\mathcal{I}_j^p = \mathcal{I}_{j,1}^p \cup \cdots \cup \mathcal{I}_{j,K_j^p}^p$. Let

```
double Lanczos (const Operator& A,
                const Operator& B,
                Vector& r, int K) {
    v[0] = B * r;
    f[0] = A * v[0];
    Scalar s = sqrt(v[0]*f[0]);
    v[0] *= 1.0/s;
    f[0] *= 1.0/s;
    for (int k=0; k<K; ++k) {
        Vector w = B * f[k];
        v[k+1] = w;
        for (int i=0; i<=k; ++i) {
            H[i][k] = w * f[i];
            v[k+1] -= H[i][k] * v[i];
        }
        f[k+1] = A * v[k+1];
        H[k+1][k] = sqrt(f[k+1]*v[k+1]);
        v[k+1] *= 1.0/H[k+1][k];
        f[k+1] *= 1.0/H[k+1][k];
    }
    return SpectralConditionNumber(K,H);
}
```

**Fig. 4** Source code for the parallel Lanczos algorithm. Here, the vectors `v[k]` span the Krylov space in $\underline{V}_J$, and `f[k] = A * v[k]` are auxiliary vectors in $\underline{V}'_J$. If the operators `A` and `B` are symmetric and positive definite, `H[i][k]` is vanishing for $i > k+1$ (up to rounding errors). Note that `H[k+1][k]` is non-vanishing as long the Krylov space has full dimension. In our numerical tests, we use a combination of the cg method and the Lanczos method, simultaneously solving the linear system and computing the condition number of the iteration matrix.

$N^p_{j,k} = \#\mathcal{I}^p_{j,k}$, let $\underline{I}^p_{j,k} \in \mathbb{R}^{N^p_j \times N^p_{j,k}}$ the trivial prolongation matrix, let $\underline{A}^p_{j,k} = (\underline{I}^p_{j,k})^T \operatorname{accumulate}(\underline{A}^p_j)\underline{I}^p_{j,k}$ be the accumulated Galerkin restriction of the operator $\underline{A}_j$, and let

$$\underline{R}^p_{j,k} = \left(\underline{A}^p_{j,k}\right)^{-1}, \qquad k = 1, ..., K^p_j,$$

be the local subspace solver. Altogether, this defines for a given functional $\underline{f}_j \in \underline{V}'_j$ the hybrid smoothing operation $\underline{c}_j = \underline{R}_j \underline{f}_j$ as follows:

S1. In parallel, for every processor $p$ set $\underline{c}^{p,0}_j = 0$.

S2. On processor $p$, apply locally the successive subspace correction method: for $k = 1, ..., K^p_j$, compute the actual subspace residual

$$\underline{f}^p_{j,k} = (\underline{I}^p_{j,k})^T \left(\underline{f}^p_j - \operatorname{accumulate}(\underline{A}^p_j)\underline{c}^{p,k-1}_j\right)$$

and add the subspace correction

$$\underline{c}^{p,k}_j = \underline{c}^{p,k}_j + \underline{I}^p_{j,k}\underline{R}^k_{j,k}\underline{f}^p_{j,k}.$$

S3. Set $\underline{c}_j = \theta \operatorname{accumulate}(\underline{c}^{p,K^p_j}_j)$.

Note that in the evaluation of the subspace residual in Step S2 the accumulated matrix has to be used in order to avoid communication within the local successive subspace correction. Since in Step S2 for every $k$ only the indices in $\mathcal{I}^p_{j,k}$ of $\operatorname{accumulate}(\underline{A}^p_j)\underline{c}^{p,k-1}_j$ have to be computed, this step has only the complexity of the subspace size.

In our examples, we use three block decompositions:

(p) a (non-overlapping) point-block decomposition with

$$\mathcal{I}^p_{j,z} = \{i \in \mathcal{I}^p_j : z_i = z\},$$

(c) an overlapping cell-block decomposition with

$$\mathcal{I}^p_{j,c} = \{i \in \mathcal{I}^p_j : z_i \in \mathcal{N}_c\},$$

(r) an overlapping row-block decomposition with

$$\mathcal{I}^p_{j,k} = \{i \in \mathcal{I}^p_j : \underline{A}^p_{j,i,k} \neq 0\}.$$

## 6 The analysis of parallel multigrid methods with overlapping block smoothing

We shortly review the multigrid analysis adopted to parallel multigrid methods. In principle, we have two options for the analysis which can be summarized as follows.

For nested spaces and symmetric elliptic problems, the analysis of the subspace correction method provides a unified theory for multigrid methods and domain decomposition methods, see [36, 28]. This is based on

(D) a stable decomposition;

(C) a strengthened Cauchy-Schwarz inequality.

Mesh-independent convergence can be obtained without any regularity of the problem. The parallel multigrid method defined in the previous section is a hybrid subspace correction method with multiplicative multilevel corrections, multiplicative local smoothing, and additive parallel smoothing.

A more general multigrid theory (which can be applied also to non-nested spaces and non-conforming discretizations) is based on

(A) an approximation property;

(S) a smoothing property,

see [18, 6]. In this setting, mesh-independent convergence can be obtained only with some regularity of the problem.

Here, we show how the parallel multigrid method with local overlapping Block-Gauss-Seidel smoothing can be analyzed as subspace correction method by combining arguments from [8, Sect. 8] and [19, Chap. 11].

*Operators for the multigrid analysis*

In order to prove a level-independent bound for the condition number (4) we analyze the equivalent operator $T_J = B^{\mathrm{mg}}_J A_J$ in $V_J$ with $E_J \circ \underline{B}^{\mathrm{mg}}_J = B^{\mathrm{mg}}_J \circ E'_J$.

The parallel multigrid method corresponds to a hybrid additive/multiplicative subspace correction method based on the overlapping decomposition

$$V_J = V_0 + \sum_{j=1}^{J} \sum_{p\in\mathcal{P}} \sum_{k_p=1}^{K_j^p} V_{j,k_p}^p$$

with

$$V_{j,k}^p = \{\phi_{j,i} \colon i \in \mathcal{I}_{j,k}^p\}\,.$$

We define the embeddings

$$I_j \colon V_j \longrightarrow V_J\,,$$
$$I_{j,k}^p \colon V_{j,k}^p \longrightarrow V_J\,,$$

Galerkin approximations

$$A_j = (I_j)' A_J I_j \colon V_j \longrightarrow V_j'\,,$$
$$A_{j,k}^p = (I_{j,k}^p)' A_J I_{j,k}^p \colon V_{j,k}^p \longrightarrow (V_{j,k}^p)'\,,$$

and projections

$$P_0 = (A_0)^{-1}(I_0)' A_J \colon V_J \longrightarrow V_0\,,$$
$$P_{j,k}^p = (A_{j,k}^p)^{-1}(I_{j,k}^p)' A_J \colon V_J \longrightarrow V_{j,k}^p\,.$$

This gives for the local smoothing operator $R_j^p$ on processor $p$

$$\mathrm{id} - R_j^p A_J = \prod_{k=1}^{K_j^p} \left(\mathrm{id} - I_{j,k}^p P_{j,k}^p\right),$$

for the parallel smoother on level $j$

$$R_j = \theta \sum_{p\in P} R_j^p \colon V_J' \longrightarrow V_J\,,$$

and recursively for the multigrid preconditioner

$$\mathrm{id} - T_j = (\mathrm{id} - R_j' A_J)^M (\mathrm{id} - T_{j-1})(\mathrm{id} - R_j A_J)^M$$

with $T_0 = I_0 P_0 \colon V_J \longrightarrow V_J$. Thus, we have $T_J = B_J^{\mathrm{mg}} A_J$.

*Finite element setting for the multigrid analysis*

For the following analysis we assume (by changing the notation) that homogeneous Dirichlet boundary conditions are included into $V_j$, so that $\|\nabla v\|$ is a norm. We assume that the bilinear form $a(\cdot,\cdot)$ is spectrally equivalent to the Laplace problem, i.e.,

$$c_a \|\nabla v_J\|^2 \le a(v_J, v_J) \le C_a \|\nabla v_J\|^2\,, \quad v_J \in V_J\,. \quad (5)$$

Note that in our application to elasticity in $\mathbb{R}^3$ this corresponds to a system of three Laplace problems. The corresponding energy norm is denoted by

$$|\!|\!| v_J |\!|\!| = \sqrt{\langle A_J v_J, v_J\rangle}\,, \qquad v_J \in V_J\,.$$

We assume that the spaces $V_0 \subset V_1 \subset \cdots \subset V_J$ are nested and that $V_j$ is shape regular with size $h_j = 2^{-j} h_0$. This gives the inverse inequality

$$\langle A_j v_j, v_j\rangle \le C_{\mathrm{inv}} h_j^{-2} \|v_j\|^2\,, \qquad v_j \in V_j$$

and for the $L_2$ projection $Q_j \colon V_J \longrightarrow V_j$ defined by

$$(Q_j v_J, w_j) = (v_J, w_j)\,, \qquad v_J \in V_J,\ w_j \in V_j$$

we have the approximation result

$$\|v_j - Q_{j-1} v_j\| \le C_Q h_j \,\|\nabla v_j\|\,, \qquad v_j \in V_j$$

for $j > 0$.

*Analysis of the smoother*

The analysis of the parallel hybrid smoother combines the additive and the multiplicative smoothing analysis in [8, Sect. 8]. Therefore, we introduce the symmetrization

$$(\mathrm{id} - R_j' A_J)(\mathrm{id} - R_j A_J) = (\mathrm{id} - \bar R_j A_J)$$

with $\bar R_j = \sum_{p\in\mathcal{P}} \bar R_j^p$ and $\bar R_j^p = R_j^p + R_j^{p\prime} - R_j^{p\prime} A_J R_j^p$.

The smoother estimates depend on overlapping constants. Let $C_{\mathrm{loc}}, C_{\mathrm{glob}} > 0$ be given by

$$C_{\mathrm{loc}} = \max_{p\in\mathcal{P}} \max_{k\in\{1,\dots,K_j^p\}} \#\{l \in \{1,\dots,K_j^p\} \colon V_{j,l}^p \cap V_{j,k}^p \ne \emptyset\}$$

and

$$C_{\mathrm{glob}} = \max_{p\in\mathcal{P}} \#\{q \in \mathcal{P} \colon V_j^q \cap V_j^p \ne \emptyset\}\,.$$

The constant $C_{\mathrm{loc}}$ is bounded independent of the level $j$ for our examples (p), (c), (r). The constant $C_{\mathrm{glob}}$ depends on the load balancing and is bounded by

$$\max_{0\le j\le J} \max_{i\in\mathcal{I}_j} \#\pi_j(i)$$

independently of the number of processors $P$.

Moreover, we need splitting constants $C_{\mathrm{overlap}}, C_{\mathrm{par}}$. We assume that for all $v_j^p \in V_j^p$ a local decomposition

$$v_j^p = \sum_{k=1}^{K_p} v_{j,k}^p\,, \qquad v_{j,k}^p \in V_{j,k}^p$$

exists satisfying

$$\sum_{k=1}^{K_p} \|v_{j,k}^p\|^2 \le C_{\mathrm{overlap}} \|v_j^p\|^2\,,$$

and we assume that for all $v_j \in V_j$ a parallel decomposition

$$v_j = \sum_{p\in\mathcal{P}} v_j^p\,, \qquad v_j^p \in V_j^p$$

exists satisfying

$$\sum_{p\in\mathcal{P}} \|v_j^p\|^2 \le C_{\mathrm{par}} \|v_j\|^2\,.$$

**Lemma 1** *A damping factor $\theta \in (0,1)$ independent of $j$ exists such that the symmetrized smoothing operator $\bar R_j$ is invertible on $V_j$ and such that constants $C_R > 0$ and $\omega_R \in (0,2)$ independent of $j$ exists with*

$$\langle A_j R_j A_j v_j, R_j A_j v_j\rangle \le \omega_R \langle A_j v_j, R_j A_j v_j\rangle\,, \quad (6)$$
$$\langle \bar R_j^{-1} v_j, v_j\rangle \le C_R h^{-2} \|v_j\|^2\,, \quad v_j \in V_j\,. \quad (7)$$

*Proof* In the first step we show (using [8, Th. 8.2]) that the local parallel smoother $R_j^p$ satisfies the conditions in [8, Th. 8.3].

Locally, for $v_j^p \in V_j^p$ we define $w_j^p \in V_j^p$ by

$$(w_j^p, \phi_j^p) = \langle A_j v_j^p, \phi_j^p\rangle\,, \qquad \phi_j^p \in V_j^p\,. \quad (8)$$

Since $R_j^p$ is defined as a multiplicative subspace correction method with local exact solving, we obtain for the local overlapping Gauss-Seidel smoother

$$\langle A_j R_j^p A_j v_j^p, R_j^p A_j v_j^p \rangle \le \omega_{\mathrm{GS}} \langle A_j v_j^p, R_j^p A_j v_j^p \rangle\,, \qquad (9)$$
$$\|w_j^p\|^2 \le C_{\mathrm{GS}} h_j^{-2} (w_j^p, \bar{R}_j^p A_j v_j^p) \qquad (10)$$

with

$$\omega_{\mathrm{GS}} = \frac{2C_{\mathrm{loc}}}{C_{\mathrm{loc}}+1}\,,$$
$$C_{\mathrm{GS}} = C_{\mathrm{inv}} C_{\mathrm{overlap}} C_{\mathrm{loc}}^2$$

(cf. [8, Th. 8.2], where the result is adapted to our notation).

Since (10) corresponds to Hypothesis (iv) and (9) corresponds to Hypothesis (iii), we can apply [8, Th. 8.3]. This gives (6) for a sufficiently small damping parameter $\theta \in (0,1)$ satisfying

$$\omega_R := \theta \omega_{\mathrm{GS}} C_{\mathrm{glob}} < 2\,. \qquad (11)$$

For $v_j \in V_j$ define $w_j \in V_j$ by $(w_j, \phi_j) = \langle A_j v_j, \phi_j \rangle$, $\phi_j \in V_j$. Again using [8, Th. 8.3] we obtain

$$\|w_j\|^2 \le C_R h_j^{-2} (w_j, \bar{R}_j A_j v_j) \qquad (12)$$

with

$$C_R = \max\big\{1 + C_{\mathrm{glob}}(1-\theta C_{\mathrm{loc}}), 1\big\} \frac{C_{\mathrm{par}} C_{\mathrm{GS}}}{\theta}\,.$$

This shows that the operator $\bar{R}_j A_j$ is positive definite and that $\bar{R}_j^{-1} \colon V_j \longrightarrow V_j'$ is well defined. Moreover, introducing the $L_2$ duality map $D_j \colon V_j \longrightarrow V_j'$ by

$$\langle D_j \phi_j, \psi_j \rangle = (\phi_j, \psi_j)\,, \qquad \phi_j, \psi_j \in V_j\,,$$

we obtain $(w_j, \bar{R}_j A_j v_j) = (w_j, \bar{R}_j D_j w_j)$. Since $\bar{R}_j D_j$ is symmetric positive definite in $L_2$, this gives the identities

$$(w_j, \bar{R}_j D_j w_j) = \|(\bar{R}_j D_j)^{1/2} w_j\|^2$$
$$\langle \bar{R}_j^{-1} w_j, w_j \rangle = \|(\bar{R}_j D_j)^{-1/2} w_j\|^2\,.$$

Thus, (12) is equivalent to (7).

**Corollary 1** *We have*

$$\langle A_j v_j, v_j \rangle \le \langle \bar{R}_j^{-1} v_j, v_j \rangle\,, \qquad v_j \in V_j\,. \qquad (13)$$

*Proof* Since $\bar{R}_j A_j$ is positive definite (as a consequence of (10)) and symmetric with respect to $a(\cdot,\cdot)$, we have

$$\|(\bar{R}_j A_j)^{1/2} v_j\|^2 = \langle A_j v_j, \bar{R}_j A_j v_j \rangle$$
$$= \|v_j\|^2 - \|(\mathrm{id} - R_j A_j) v_j\|^2 \le \|v_j\|^2$$

and thus

$$\langle A_j v_j, v_j \rangle = \|v_j\|^2 \le \|(\bar{R}_j A_j)^{-1/2} v_j\|^2 = \langle \bar{R}_j^{-1} v_j, v_j \rangle\,.$$

*Analysis of the multigrid V-cycle*

The multigrid V(1,1)-cycle (with one pre- and one post-smoothing step and exact coarse problem solver) is a successive subspace correction method on

$$V_J = V_0 + \cdots + V_J$$

with approximate solution operators $R_j$ on the subspaces. The main observation in multigrid theory is that $R_j A_j \approx$ id on the subspaces $W_j = (Q_j - Q_{j-1}) V_J$ of "high frequencies". Analytically, the main property is the corresponding stable splitting

$$\|\nabla Q_0 v_J\|^2 + \sum_{j=1}^J h_j^{-2} \|(Q_j - Q_{j-1}) v_J\|^2 \le C_{\mathrm{stab}} \|\nabla v_J\|^2$$

for $v_J \in V_J$, and the strengthened Cauchy-Schwarz inequality

$$|(\nabla v_i, \nabla w_j)| \le C_{\mathrm{CS}} 2^{(i-j)/2} \|\nabla v_i\|\, \|\nabla w_j\|$$

for $v_i \in V_i$, $w_j \in W_j$, $i < j$ [5]. We now show that the analysis in [19, Chap. 11.6] applies also to the parallel hybrid smoother and to problems which are spectrally equivalent to the Laplace problem.

Since $T_0^2 = T_0$, we have

$$\|\mathrm{id} - T_J\| = \|(\mathrm{id} - T_0)(\mathrm{id} - R_1 A_J) \cdots (\mathrm{id} - R_J A_J)\|^2$$
$$= \|(\mathrm{id} - T_0)(\mathrm{id} - \bar{R}_1 A_J) \cdots (\mathrm{id} - \bar{R}_J A_J)\|\,.$$

**Lemma 2** *The condition number $\kappa(B_J^{\mathrm{mg}} A_J)$ for the parallel multigrid method with hybrid smoothing is bounded independently of $J$.*

*Proof* We define the smoother dependent norm

$$\|v\|_j = \sqrt{\langle \bar{R}_j^{-1} v_j, v_j \rangle}\,, \qquad v_j \in V_j\,,$$

where we set $\bar{R}_0 = A_0^{-1}$ for compatibility (cf. [26]). Then, Lemma 1 and the stable $L_2$ splitting gives

$$\sum_{j=0}^J \|(Q_j - Q_{j-1}) v_J\|_j^2 \le \frac{C_R C_{\mathrm{stab}}}{c_a} \|v_J\|^2\,.$$

Next, Corollary 1 and the strengthened Cauchy-Schwarz inequality yields for $v_i \in V_i$, $w_j \in W_j$, $i < j$,

$$|a(v_i, w_j)| \le \frac{C_a C_{\mathrm{CS}} 2^{(i-j)/2}}{c_a} \|v_i\|_j\, \|w_j\|_j\,.$$

Now, estimating $\|(2^{(i-j)/2})_{i,j=0,\dots,J}\|_\infty \le 2 + \sqrt{2}$ gives for the multiplicative subspace correction method

$$\|\mathrm{id} - B_J^{\mathrm{mg}} A_J\| \le 1 - \frac{1}{C_1 (1 + C_2)^2} =: \rho_{\mathrm{mg}} < 1$$

with

$$C_1 = \frac{C_R C_{\mathrm{stab}}}{c_a}\,, \qquad C_2 = \frac{C_a C_{\mathrm{CS}} (2 + \sqrt{2})}{c_a}$$

[19, Th. 11.4.3]. Since the operator $\mathrm{id} - B_J^{\mathrm{mg}} A_J$ is positive semi-definite by construction, we obtain

$$(1 - \rho_{\mathrm{mg}}) \|v_J\|^2 \le a(B_J^{\mathrm{mg}} A_J v_J, v_J) \le \|v_J\|^2$$

and thus $\kappa(B_J^{\mathrm{mg}} A_J) \le \dfrac{1}{1 - \rho_{\mathrm{mg}}}$ .

*Remark 1* Since $R_j$ satisfies (6), the analysis in [37] shows convergence also for the $V(1,0)$-cycle or the $V(0,1)$-cycle. For the symmetric cycle, (6) is replaced by (13) which is independent of the damping parameter. Nevertheless, sufficient damping is required in order to guarantee that $\bar{R}_j^{-1}$ exists, and indeed we observe in the numerical experiments that damping is necessary for the parallel smoother: without damping, the Lanczos method shows that $B_J^{\mathrm{mg}} A_J$ is not positive definite and thus the cg method is not convergent.
On the other hand, strong damping should be avoided, since this increases $C_R$ and thus the estimate for the condition number of the multigrid preconditioner. This can be also observed in the experiments.

*Spectrally equivalent preconditioner*

The norm equivalence (5) suggests to use for preconditioning $A_J$ a multigrid preconditioner $B_J^{\mathrm{mg,lap}}$ for the Laplace problem $A_J^{\mathrm{lap}}$, i.e., $\langle A_J^{\mathrm{lap}} v_J, w_J\rangle = (\nabla v_J, \nabla w_J)$ for $v_J, w \in V_J$. The condition number of $\kappa(B_J^{\mathrm{mg,lap}} A_J^{\mathrm{lap}})$ can be estimated from spectral bounds

$$c_{\mathrm{lap}}\|\nabla v_J\|^2 \le (\nabla B_J^{\mathrm{mg,lap}} A_J^{\mathrm{lap}} v_J, \nabla v_J) \le C_{\mathrm{lap}}\|\nabla v_J\|^2$$

which gives

$$\kappa(B_J^{\mathrm{mg,lap}} A_J^{\mathrm{lap}}) \le \frac{C_{\mathrm{lap}}}{c_{\mathrm{lap}}}\,.$$

Then, the condition number of $B_J^{\mathrm{mg,lap}} A_J$ can be estimated from Korn's inequality

$$c_{\mathrm{K}}\|\nabla v_J\|^2 \le \|\operatorname{sym}(\nabla v_J)\|^2 \le 2\,\|\nabla v_J\|^2\,, \qquad v_J \in V_J$$

and the Lamé parameters $\mu, \lambda$: from

$$4\mu^2\|\operatorname{sym}(\nabla v_J)\|^2 \le a(v_J, v_J) \le (2\mu + 3\lambda)^2\|\operatorname{sym}(\nabla v_J)\|^2$$

we obtain

$$\kappa(B_J^{\mathrm{mg,lap}} A_J) \le \frac{C_{\mathrm{lap}}}{c_{\mathrm{lap}}}\frac{2\mu + 3\lambda}{2\mu}\sqrt{\frac{2}{c_{\mathrm{K}}}}\,. \qquad (14)$$

In the special case of pure Dirichlet boundary conditions on $\Gamma = \partial\Omega$ we can estimate $c_{\mathrm{K}} \ge 1/2$ [6, Chap. 5.3], which yields

$$\kappa(B_J^{\mathrm{mg,lap}} A_J) \le \frac{C_{\mathrm{lap}}}{c_{\mathrm{lap}}}\frac{2\mu + 3\lambda}{\mu}\,. \qquad (15)$$

We will see in the numerical experiments below that this estimate provides a resonable prediction of the convergence.

## 7 Parallel multigrid performance

We present results for the parallel multigrid solution of problems in linear elasticity and plasticity. The material parameters for our tests are given in Tab. 1. For the parallel experiments we use a Linux cluster with dual board QuadCore nodes and Infiniband switches. All tests

| parameter | symbol | value |
|---|---|---|
| Young's modulus | $E$ | $206\,900.0\,\mathrm{kN/m^2}$ |
| Poisson number | $\nu$ | 0.29 |
| Lamé parameter | $\mu = \frac{E}{2(1+\nu)}$ | $80\,193.8\,\mathrm{kN/m^2}$ |
| | $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ | $110\,743.8\,\mathrm{kN/m^2}$ |
| yield stress | $y_0$ | $450\ \mathrm{kN/m^2}$ |
| hardening modulus | $H_0$ | $0.0001\mu$ |

**Table 1** Material parameters for linear elasticity and infinitesimal plasticity with kinematic hardening.

are realized in our research code M++. For the coarse problem the matrix $\underline{A}_0$ is collected on one processor, and for $\underline{B}_0$ we use the sparse direct solver [9].

For the quantitative evaluation of the parallel multigrid method we consider four different test configurations:

1. In the first test configuration, $\Omega$ is the unit cube, and on $\partial\Omega$ Dirichlet boundary conditions are imposed. We use a structured mesh with $8^J$ hexahedral cells (i.e., the vertices build a regular Cartesian grid of mesh size $2^{-J}$). For this example, we have full regularity, i.e., the solution of the continuous problem is at least in $H^2$. As a consequence, the multigrid analysis based on approximation property and smoothing property estimates the spectral radius of the error propagation operator $\mathrm{id} - B_J^{\mathrm{mg}} A_J$ by $O(1/M)$, where $M$ is the number of smoothing steps.
2. For the second test configuration we only change the boundary condition: we use Dirichlet boundary condition on the bottom of the cube only, and Neumann boundary conditions on the remaining 5 faces. Thus, we also have full regularity, but the Poincaré-Friedrichs constant $C_{\mathrm{PF}}$
$$\|u\|_{H^1(\Omega)} \le C_{\mathrm{PF}}\|\nabla u\|_{L_2(\Omega)}$$
is much larger.
3. In the third test configuration, $\Omega$ is a union of regular cubes but non-convex with reentrant corners. Thus, we have very limited regularity, and the multigrid analysis based on approximation property and smoothing property only gives poor results, i.e., a convergence estimate $O(1/M^\alpha)$ for multiple smoothing with $\alpha$ close to 0. Nevertheless, since the vertices are a subset of the regular grid $2^{-J}\mathbb{Z}^3$, we have optimal shape regularity of the cells.
4. Finally, we consider an example with an unstructured mesh with isoparametric hexahedral cells. In particular, some angles between edges are larger than $90°$, but the aspect ratio is moderate. The underlying domain $\Omega$ is a thick plate with a cylindric hole (by symmetry, we consider only a part of the configuration), so that we have full regularity (up

to the corners induced by the piecewise linear approximation of the hole). We use Neumann boundary conditions outside of the full configuration of a plate with a cylindrical hole, and on the boundaries which arise from the symmetric reduction, we use homogeneous Dirichlet boundary conditions in normal direction of the symmetry planes.

The condition number of the operator $B_J^{\mathrm{mg}} A_J$ is estimated by the Lanczos method, see Fig. 5. We test on two successive levels the point block smoother (p) with $M$ smoothing steps, and the overlapping smoothers (c) and (r).

For comparison, we also test the Laplace multigrid preconditioner $B_J^{\mathrm{mg,lap}}$ applied to the elasticity problem $A_J$. The condition number $\kappa(B_J^{\mathrm{mg,lap}} A_J)$ estimates the norm equivalence factor between the elasticity system and the Laplace problem.

From the performance results in Fig. 5 of the four test cases we make the following observations:

- ▶ The condition number of the point block smoother with $M = 1$ and the overlapping smoothers (c) and (r) are nearly independent from the boundary condition, the regularity, and from the refinement level in the first three test cases, and the condition number is significantly larger for the forth example. This indicates that multigrid performance with only a single smoothing step behaves as predicted by the subspace correction method and mainly depends on the shape regularity (and is best on regular grids).
- ▶ Multiple smoothing improves the multigrid performance only in case of sufficient regularity in examples 1, 2, and 4. Asymptotically, we observe for the spectral radius
$$\rho(\mathrm{id} - B_J^{\mathrm{mg}} A_J) \approx 1 - 1/\kappa(B_J^{\mathrm{mg}} A_J) \approx O(1/M)$$
as predicted by the multigrid analysis based on approximation and smoothing property.
- ▶ Multigrid performance seems to be uncorrelated to the norm equivalence factor in Korn's inequality which is estimated by the condition number $\kappa(B_J^{\mathrm{mg,lap}} A_J)$. We see in the first test example that the estimate (15) provides a realistic prediction for this condition number. On the other hand, since $\kappa(B_J^{\mathrm{mg,lap}} A_J^{\mathrm{lap}})$ is close to one in our test, the estimate (14) can be used to estimate the constant in Korn's inequality, i.e., we have
$$c_{\mathrm{K}} \approx \Big(\frac{2\mu + 3\lambda}{2\mu}\Big)^2 \frac{2}{\kappa(B_J^{\mathrm{mg,lap}} A_J)} \,.$$
- ▶ Comparing the overlapping smoothers (c) and (r) we observe that in parallel increasing overlap does not automatically improve the performance of the multigrid method. This may be explained by the requirement of stronger damping for the larger overlap in the additive part of the parallel smoothing scheme.

Next, we consider the multigrid performance for the linearized problem in plasticity, see Fig. 6. Here, we use the configuration of test example 4, and by increasing the external load more and more of the domain is plastified. For the linearization this leads to nearly singular systems close to the limit load of perfect plasticity (where the linearization is singular). For the time discretization we use the backward Euler scheme, and for the nonlinearity and its linearization in the incremental problem we use the standard radial return algorithm [27,30]. We can use this test as a robustness test for multigrid. From the results we clearly observe that multiple smoothing as well as overlapping smoothing improves the robustness of the multigrid performance.

Finally, as a snapshot for the current performance on production machines, the computing times for a full time-dependent elasto-plastic simulation (using the same configuration as above) are presented in Tab. 2. We observe a good scaling for more processors but due to the nonlinearity we cannot expect stable computing times in this parallel experiment.

*Acknowledgement*

The author gratefully acknowledges the help of Wolfgang Müller, Martin Sauter, Antje Sydow, Alexander Bulovyatov, and Daniel Maurer for parts of the implementation within the presented code M++.

## References

1. P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Teubner Skripten zur Numerik. Teubner Stuttgart, Stuttgart, 1996.
2. P. Bastian, K. Birken, K. Johannsen, S. Lang, N. Neuß, H. Rentz-Reichert, and C. Wieners. UG – a flexible software toolbox for solving partial differential equations. *Comp. Vis. Sci.*, 1:27–40, 1997.
3. P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework. *Computing*, 82:103–119, 2008.
4. P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in DUNE. *Computing*, 82:121–138, 2008.
5. F. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numerische Mathematik*, 64:455–476, 1993.
6. D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics.* Cambridge University Press, 1997.
7. J. H. Bramble. *Multigrid Methods.* Longman Scientific & Technical, Essex, 1993.

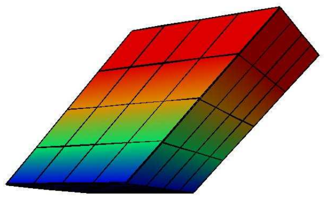
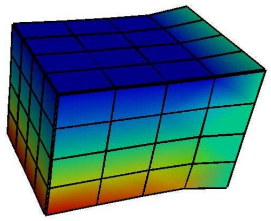
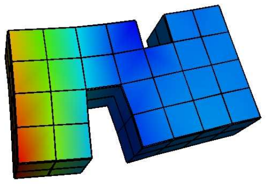
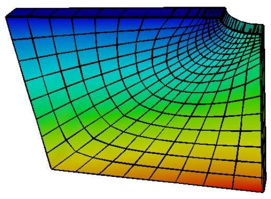
| Example 1 | Example 2 | Example 3 | Example 4 |
|---|---|---|---|
| full regularity | full regularity | reduced regularity | some regularity |
| Dirichlet b.c. | mixed b.c. | mixed b.c. | mixed b.c. |
| structured | structured | structured | unstructured |
|  |  |  |  |
| 823 875 d.o.f. | 823 875 d.o.f. | 655 875 d.o.f. | 449 307 d.o.f. |
| 262 144 cells | 262 144 cells | 204 800 cells | 131 072 cells |
| $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 3.45$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 29.29$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 459.3$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 31.2$ |
| $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 2.09$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 2.01$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 2.06$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 49.7$ |
| $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.22$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.31$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.37$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 23.5$ |
| $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.07$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.15$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.28$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 11.9$ |
| $\kappa(B_J^{\mathrm{mg,p,8}}A_J) = 1.03$ | $\kappa(B_J^{\mathrm{mg,p,8}}A_J) = 1.09$ | $\kappa(B_J^{\mathrm{mg,p,8}}A_J) = 1.24$ | $\kappa(B_J^{\mathrm{mg,p,8}}A_J) = 6.2$ |
| $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.87$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.85$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.79$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 19.5$ |
| $\kappa(B_J^{\mathrm{mg,r,1}}A_J) = 1.86$ | $\kappa(B_J^{\mathrm{mg,r,1}}A_J) = 1.92$ | $\kappa(B_J^{\mathrm{mg,r,1}}A_J) = 1.99$ | $\kappa(B_J^{\mathrm{mg,r,1}}A_J) = 19.3$ |
| 6 440 067 d.o.f. | 6 440 067 d.o.f. | 5 079 555 d.o.f. | 3 368 499 d.o.f. |
| 2 097 152 cells | 2 097 152 cells | 1 638 400 cells | 1 048 576 cells |
| $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 3.46$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 29.66$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 382.7$ | $\kappa(B_J^{\mathrm{mg,lap}}A_J) = 35.1$ |
| $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 2.08$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 1.98$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 1.97$ | $\kappa(B_J^{\mathrm{mg,p,1}}A_J) = 54.3$ |
| $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.08$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.31$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 1.40$ | $\kappa(B_J^{\mathrm{mg,p,2}}A_J) = 26.2$ |
| $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.04$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.16$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 1.31$ | $\kappa(B_J^{\mathrm{mg,p,4}}A_J) = 13.3$ |
| $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.60$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.61$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 1.72$ | $\kappa(B_J^{\mathrm{mg,c,1}}A_J) = 15.7$ |

**Fig. 5** Numerical approximation of the condition number of the iteration matrix for different V-cycle multigrid preconditioners: the (non-overlapping) point-block smoother $B_J^{\mathrm{mg,p},M}$ with $M$ pre- and postsmoothing steps, and the (overlapping) cell/row block smoothers $B_J^{\mathrm{mg,c,1}}$, $B_J^{\mathrm{mg,r,1}}$. In all cases, we use 64 processors, and we use a fixed damping factor $\theta = 0.9$ for the smoother. We perform the cg method up to an error reduction of $10^{-10}$, and the condition number of the iteration matrix is simultaneously determined from the corresponding Krylov space with the Lanczos method. For comparison, we also test the Laplace multigrid V-cycle $B_J^{\mathrm{mg,lap}}$ as a preconditioner for the elasticity problem.

8. J. H. Bramble and X. Zhang. The analysis of multigrid methods. In P. Ciarlet and J. Lions, editors, *Numerical Methods in Scientific Computing*, volume VII of *Handbook of Numerical Analysis*, pages 173–416. Elsevier Science B.V., 2000.
9. J. W. Demmel, J. R. Gilbert, and X. S. Li. An asynchronous parallel supernodal algorithm for sparse Gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 20(4):915–952, 1999.
10. K. D. Devine, E. G. Boman, R. T. Heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, J. E. Flaherty, and L. G. Gervasio. New challenges in dynamic load balancing. *Appl. Numer. Math.*, 52(2-3):133–152, 2005.
11. C. C. Douglas. A review of numerous parallel multigrid methods. Astfalk, Greg (ed.), Applications on advanced architecture computers. Philadelphia, PA: SIAM. Software - Environments - Tools, 1996.
12. F. Durst and M. Schäfer. A parallel block-structured multigrid method for the prediction of incompressible flows. *Int. J. Numer. Methods Fluids*, 22(6):549–565, 1996.
13. U. Rüde (ed.). Special issue on multigrid computing and finite element methods. *Computing in Science & Engineering*, 8, 2006.
14. J. E. Flaherty, R. M. Loy, C. Özturan, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz. Parallel structures and dynamic load balancing for adaptive finite element computation. *Appl. Numer. Math.*, 26(1-2):241–263, 1998.
15. G. Golub and C. F. Van Loan. *Matrix computations.* Texts and Readings in Mathematics 43. New Delhi: Hindustan Book Agency., 2007.
16. M. Griebel and G. Zumbusch. Parallel multigrid in an adaptive PDE solver based on hashing and space-filling curves. *Parallel Comput.*, 25(7):827–843, 1999.
17. G. Haase, M. Kuhn, and U. Langer. Parallel multigrid 3D Maxwell solvers. *Parallel Comput.*, 27(6):761–775, 2001.

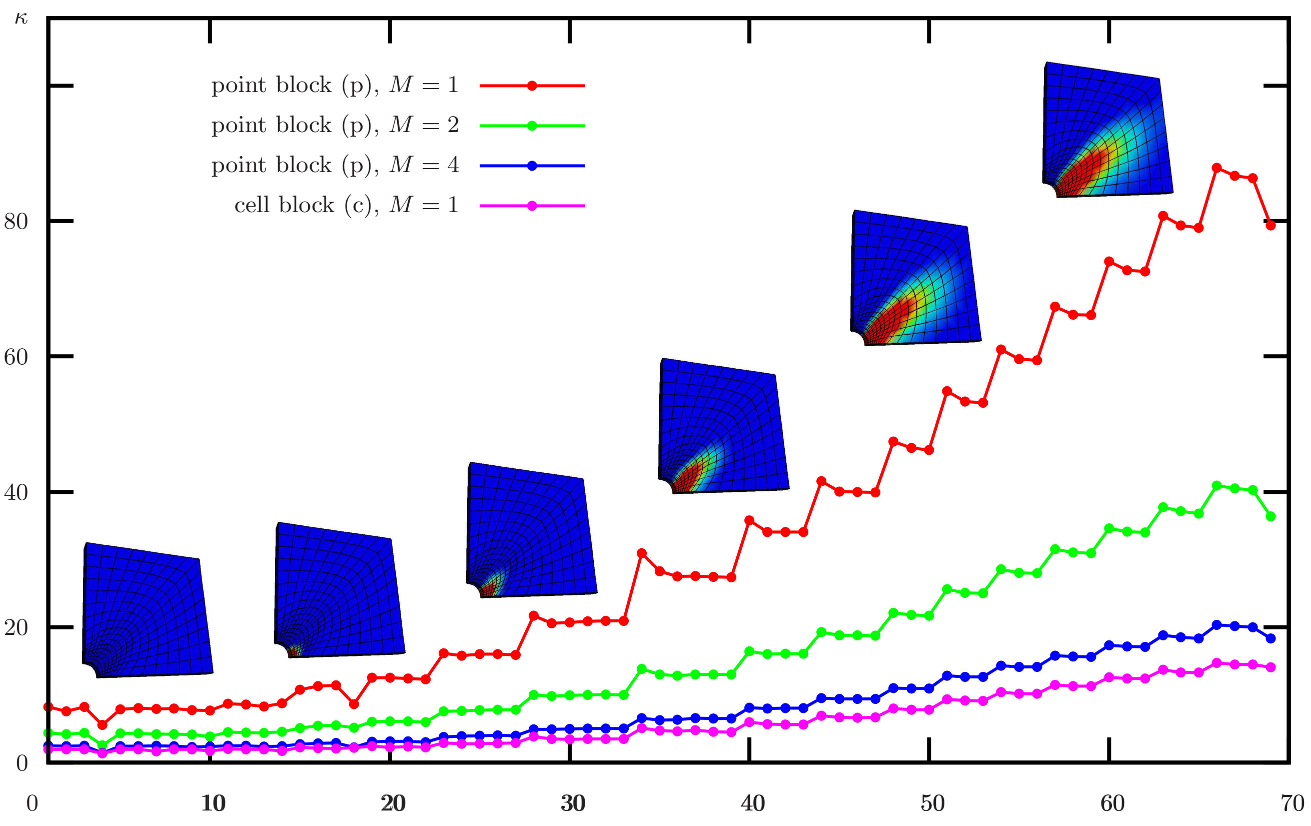


**Fig. 6** Performance of block Gauss-Seidel smoothing in plasticity. The condition number $\kappa = \kappa(B_J A_J)$ of the elasto-plastic linearization for 70 Newton steps (in 20 time steps) is displayed. Multiple smoothing and overlapping block smoothing increases the robustness of the multigrid method with respect to the increasing plastic region.

| level | d.o.f. | internal variables | number of processor | memory requirement | computing time |
|---|---|---|---|---|---|
| 0 | 10 845 | 148 736 | | | |
| 1 | 74 871 | 1 189 888 | 1 | $\leq$ 1 GB | 5 min. |
| 2 | 553 707 | 9 519 104 | 2 | $\leq$ 8 GB | 21 min. |
| 3 | 4 253 139 | 76 152 832 | 16 | $\leq$ 64 GB | 35 min. |
| 4 | 33 328 035 | 609 222 656 | 128 | $\leq$ 512 GB | 51 min. |

**Table 2** Parallel experiment with fixed load for a full elasto-plastic simulation with 7 linear elastic steps and 4 nonlinear plastic steps. On every refinement level the problem size and the number of processors are increased by a factor of 8. Since we fix the number of time steps, the incremental nonlinear problem gets more difficult on finer meshes and thus we need more Newton steps on higher levels.

18. W. Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag Berlin, 1985.
19. W. Hackbusch. *Iterative solution of large sparse systems of equations*. Springer-Verlag Berlin, 1994.
20. V. Hoang, M. Plum, and C. Wieners. A computer-assisted proof for photonic band gaps. *Z. angew. Math. Phys. (ZAMP)*, 60:1–18, 2009.
21. M. Jung. On the parallelization of multi-grid methods using a non-overlapping domain decomposition data structure. *Appl. Numer. Math.*, 23(1):119–137, 1997.
22. S. Lang. *Parallele Numerische Simulation instationärer Probleme mit adaptiven Methoden auf unstrukturierten Gittern*. PhD thesis, Universität Stuttgart, 2001.
23. W. F. Mitchell. A parallel multigrid method using the full domain partition. *ETNA, Electron. Trans. Numer. Anal.*, 6:224–233, 1997.
24. P. Neff, K. Chełmiński, W. Müller, and C. Wieners. A numerical solution method for an inifinitesimal elasto-plastic Cosserat model. *Mathematical Models and Methods in Applied Sciences (M3AS)*, 17:1211–1240, 2007.
25. P. Neff, A. Sydow, and C. Wieners. Numerical approximation of incremental infinitesimal gradient plasticity. *Int. J. Numer. Meth. Eng.*, 77:414–436, 2009.
26. N. Neuss. V-cycle convergence with unsymmetric smoothers and application to an anisotropic model problem. *SIAM J. Numer. Anal.*, 35:1201–1212, 1998.

27. J. C. Simo and T. J. R. Hughes. *Computational Inelasticity*. Springer-Verlag Berlin, 1998.
28. A. Toselli and O. Widlund. *Domain decomposition methods – Algorithms and theory*. Springer-Verlag Berlin, 2005.
29. C. Wieners. Distributed point objects. A new concept for parallel finite elements. In R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering*, volume 40 of *Lecture Notes in Computational Science and Engineering*, pages 175–183. Springer-Verlag Berlin, 2004.
30. C. Wieners. Nonlinear solution methods for infinitesimal perfect plasticity. *Z. Angew. Math. Mech. (ZAMM)*, 87:643–660, 2007.
31. C. Wieners. SQP methods for incremental plasticity with kinematic hardening. In D. Reddy, editor, *IUTAM Symposium on Theoretical, Modelling and Computational Aspects of Inelastic Media*, pages 143–154. Springer-Verlag Berlin, 2008.
32. C. Wieners, M. Ammann, and W. Ehlers. Distributed point objects: A new concept for parallel finite elements applied to a geomechanical problem. *Future Generation Computer Systems*, 22:532–545, 2006.
33. C. Wieners, M. Ammann, W. Ehlers, and T. Graf. Parallel Krylov methods and the application to 3-d simulations of a tri-phasic porous media model in soil mechanics. *Comput. Mech.*, 36:409–420, 2005.
34. C. Wieners, N. Karajan, and W. Ehlers. Parallel overlapping domain decomposition preconditioner and application to a porous media model in biomechanics. *Comput. Mech.*, 2009.
35. C. Wieners, N. Karajan, T. Graf, and W. Ehlers. Parallel solution methods for porous media models in biomechanics. *PAMM*, 5:35–38, 2005.
36. J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1992.
37. J. Xu and L. Zikatanov. The method of alternating projections and the method of subspace corrections in Hilbert space. *J. Amer. Math Soc.*, 15:573–597, 2002.
38. G. Zumbusch. *Parallel multilevel methods. Adaptive mesh refinement and load balancing.* Advances in Numerical Mathematics. Wiesbaden: Teubner, 2003.