Distributed Point Objects.
A New Concept for Parallel Finite Elements

Christian Wieners

Universität Karlsruhe, Institut für Praktische Mathematik
Englerstraße 2, 76128 Karlsruhe, Germany
(`http://www.mathematik.uni-karlsruhe.de/~ipm`)

Summary. We present a new concept for the realization of finite element computations on parallel machines with distributed memory. The parallel programming model is based on a dynamic data structure address by points. All geometric objects (cells, faces, edges) are referenced by its midpoint, and all algebraic data structures (vectors and matrices) are tied to the nodal points of the finite elements. The parallel distribution of all objects is determined by processor lists assigned to the reference points.
Based on this new model for Distributed Point Objects (DPO) a first application to a geotechnical application has been presented in Wieners et al. [2003]. Here, we consider the extension to parallel refinement, curved boundaries, and multigrid preconditioners. Finally, we present parallel results for a nonlinear model problem with isoparametric cubic elements.

1 Introduction

Many finite element applications require a fine mesh resolution or a huge number of time steps. Together with the increasing complexity of the considered models the solution of such problems are only possible with elaborated solvers such as domain decomposition methods or multigrid preconditioners. In order to obtain a reasonable computing time, very often this can be realized only on a parallel machine.

Here, we introduce a new parallel programming model which is specially designed for the support of nonlinear engineering finite element applications, and which provides a platform for the development of modern solvers and their adaptation to such problems. The main features of the concept are flexibility, transparence, and extensibility. It allows the realization of complex algorithms in a very compact form, and it reduces the implementation time for new applications.

Our concept is based on long experiences with parallel software development in Bastian et al. [1997, 1998, 1999, 2000, 2001] and parallel simulations for partial differential equations in Wieners [1999, 2000a,b], Lang et al. [2002].

It can be understood as an abstraction and simplification of our previous code. In terms of software engineering this study describes the underlying structure of a prototype implementation which provides optimal support for numerical experiments and numerical analysis in the investigation of new models, discretizations, and solvers.

This contribution is organized as follows. In Section 2, we introduce the programming model for Point Objects, which is enhanced in Section 3 to parallel distributed objects. In addition, we define a parallel refinement process leading to a hierarchical mesh structure. This is coupled in Section 4 with a multilevel parallel linear algebra, where parallel multigrid methods can be defined. In Section 5 this is combined with an abstract model for finite elements (including the requirements for isoparametric elements). Finally, in Section 6 the application to a nonlinear model problem is presented.

2 Point Objects

Our geometry model is based on a finite set of points $\mathcal{P} \subset \mathbf{R}^d$, where we consider different types of points: corner points, edge midpoints, face midpoints, cell midpoints, and the exception point $P = \infty$. A cell $C = (P_1, ..., P_N) \in \mathcal{P}^N$ is determined by a vector of N different corner points, and the convex hull of the corner points is denoted by $\mathrm{conv}(C) \subset \mathbf{R}^d$. A face $F = (P_{\mathrm{left}}, P_{\mathrm{right}}) \in \mathcal{P}^2$ is determined by a pair of points representing the midpoints of the cells C_{left} and C_{right} of the common face $\mathrm{conv}(C_{\mathrm{left}}) \cap \mathrm{conv}(C_{\mathrm{right}})$; boundary faces are characterized by $P_{\mathrm{right}} = \infty$. An edge $E = (P_{\mathrm{left}}, P_{\mathrm{right}}) \in \mathcal{P}^2$ represents the line from P_{left} to P_{right}.

Now, a mesh $\mathcal{M} = (\mathcal{C}, \mathcal{F}, \mathcal{E}, \mathcal{V}, \mathcal{B}, \mathcal{G})$ is given by

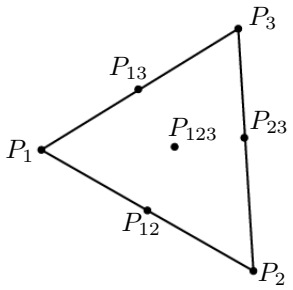
- a cell mapping $\mathcal{C} \colon \mathcal{P} \longrightarrow \bigcup_N \mathcal{P}^N$, which assigns every cell midpoint P_C the cell $C = \mathcal{C}(P_C)$ represented by the vector of N corner points;
- a face mapping $\mathcal{F} \colon \mathcal{P} \longrightarrow \mathcal{P}^2$, which assigns every face midpoint P_F the two adjacent element midpoints;
- an edge mapping $\mathcal{E} \colon \mathcal{P} \longrightarrow \mathcal{P}^2$, which assigns every edge midpoint P_E the two adjacent vertices;
- a vertex mapping $\mathcal{V}$ representing a list of vertices (and counting the number of cells with $P \in C$);
- a boundary mapping $\mathcal{B}$ representing a list of boundary faces for the assignment of segment numbers specifying boundary conditions;
- a geometry mapping $\mathcal{G} \colon \mathcal{P} \longrightarrow \mathbf{R}^d$, which assigns points P on the boundary of cells the projection onto the (possibly) curved boundary of the computational domain Ω.

All mappings return an empty object for points of wrong type. We use the notation $C \in \mathcal{C}$ if $C \in \mathcal{C}(\mathcal{P})$, and $P \in \mathcal{P}_{\mathcal{C}}$ for the cell midpoints, etc.

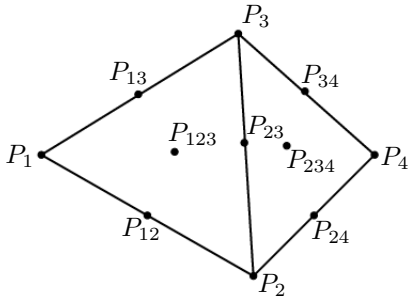
We consider only consistent meshes with admissible triangulations, i. e., we assume that the cells define a polygonal approximation $\bar{\Omega}_{\mathcal{C}} = \bigcup_{C \in \mathcal{C}} \mathrm{conv}(C)$

of a domain $\Omega \subset \mathbf{R}^d$, such that $\text{conv}(C) \cap \text{conv}(C') = \text{conv}(C \cap C')$ for two cells $C, C' \in \mathcal{C}$, i. e., the intersection is empty, a common vertex in $\mathcal{V}$, a common edge in $\mathcal{E}$, or a common face in $\mathcal{F}$. In general, we assume $\mathcal{G}(P) = P$ for the boundary vertices $P \in \mathcal{P}_\mathcal{V} \cap \partial\Omega$, and $\mathcal{G}(P) \in \partial\Omega$ for the $P \in \mathcal{P}_\mathcal{E} \cap \partial\Omega_\mathcal{C}$ and $P \in \mathcal{P}_\mathcal{F} \cap \partial\Omega_\mathcal{C}$. The geometry mapping is used for the realization of isoparametric elements and for the refinement algorithm (see below).

Example We illustrate the data structure for a mesh with two triangles in $\mathbf{R}^2$. Inserting the first triangle $C_1 = (P_1, P_2, P_3)$ in $\mathcal{M}$ results in the point set $\mathcal{P} = \{P_1, P_2, P_3, P_{12} = \frac{1}{2}(P_1 + P_2), P_{13} = \frac{1}{2}(P_1 + P_3), P_{23} = \frac{1}{2}(P_2 + P_3), P_{123} = \frac{1}{3}(P_1 + P_2 + P_3)\}$, the edges $\mathcal{E}(P_{12}) = (P_1, P_2)$, $\mathcal{E}(P_{23}) = (P_2, P_3)$, $\mathcal{E}(P_{13})$ $) = (P_{123}, \infty)$, $\mathcal{F}(P_{13})$



Now, the second triangle $C_2 = (P_2, P_4, P_3)$ is inserted, which adds the new points $P_{24} = \frac{1}{2}(P_2 + P_4)$, $P_{34} = \frac{1}{2}(P_3 + P_4)$, $P_{234} = \frac{1}{3}(P_2 + P_3 + P_4)$ to the points set $\mathcal{P}$, new edges $\mathcal{E}(P_{24}) = (P_2, P_4)$, $\mathcal{E}(P_{34}) = (P_3, P_4)$, and the new faces $\mathcal{F}$ $:= (P_{123}, P_{234})$ is now



After inserting all cells, we can identify the neighborhood relationship by the faces, where a boundary face can be identified by testing for $P_{\text{right}} = \infty$. Then, for curved boundaries the projection of edge midpoints and face midpoints onto the boundary can be computed. In general, this requires an additional data structure for the boundary definition. In our application, where the boundary is defined by a periodic cubic spline which is uniquely defined by the corner vertices, this can be realized without further geometry information.

3 Distributed Objects

A parallel distribution is determined by a partition map

$$\pi\colon \mathcal{P} \longrightarrow 2^{\{1,2,...,N_{\text{procs}}\}}$$

assigning to every point $P \in \mathcal{P}$ the subset $\pi(P) \subset \{1, 2, ..., N_{\text{procs}}\}$ of processors, where this point is represented. This defines also a unique master processor $\mu(P) = \min \pi(P)$ for every point, and it determines an overlapping partition

$$\mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_{N_{\text{procs}}}, \qquad \mathcal{P}_q = \{P \in \mathcal{P}\colon q \in \pi(P)\}.$$

We obtain the local mesh $\mathcal{M}_q = (\mathcal{C}_q, \mathcal{F}_q, \mathcal{E}_q, \mathcal{V}_q, \mathcal{B}_q, \pi_q)$ on processor q by restricting all mappings to $\mathcal{P}_q$. Then, the parallel distribution is completely determined by the partition map π. An admissible parallel distribution requires that every cell can be represented at least on one processor q, i. e., $C \in \bigcup_N \mathcal{P}_q^N$ for $C \in \mathcal{C}_q$. Moreover, for every face $F \in \mathcal{F}$ we require $F \in \bigcup_{p,q \in \pi(P_F)} \mathcal{P}_p \times \mathcal{P}_q$.

For the determination of an admissible distribution of the mesh $\mathcal{M}$ onto N_{procs} processors, we assign a destination processor $\text{dest}(C) \in \{1, 2, ..., N_{\text{procs}}\}$ to every cell $C \in \mathcal{C}$, defining a disjoint partition

$$\mathcal{C} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_{N_{\text{procs}}}, \qquad \mathcal{C}_q = \{C \in \mathcal{C}\colon \text{dest}(C) = q\}$$

and a domain decomposition

$$\Omega_{\mathcal{C}} = \Omega_1 \cup \cdots \cup \Omega_{N_{\text{procs}}}, \qquad \bar{\Omega}_q = \bigcup_{C \in \mathcal{C}_q} \text{conv}(C).$$

A corresponding compatible partition map is defined by

$$\begin{aligned}
\pi(P_C) &= \{\text{dest}(C)\}, && C \in \mathcal{C},\\
\pi(P_F) &= \{\text{dest}(C)\colon P_C \in F\}, && F \in \mathcal{F},\\
\pi(P_E) &= \{\text{dest}(C)\colon E \subset C\}, && E \in \mathcal{E},\\
\pi(P) &= \{\text{dest}(C)\colon P \in C\}, && P \in \mathcal{V}.
\end{aligned}$$

Thus, the partition map can be computed in advance before the realization of the parallel distribution.

Example (continued) The parallel distribution with $\text{dest}(C_1) = 1$ and $\text{dest}(C_2) = 2$ results in $\pi(P_1) = \pi(P_{12}) = \pi(P_{13}) = \pi(P_{123}) = \{1\}$, $\pi(P_2) = \pi(P_3) = \pi(P_{23}) = \{1, 2\}$, and $\pi(P_4) = \pi(P_{24}) = \pi(P_{34}) = \pi(P_{234}) = \{2\}$.

Parallel Refinement

A uniform refinement of a cell $C = (P_1, ..., P_N)$ in $\mathcal{M}$ is defined by a refinement rule $\mathcal{R} = \{r_{ij} : i = 1, ..., N,\ j = 1, ..., 2^d\}$: let $(P_1, ..., P_M) = (\mathcal{V}_C, \mathcal{E}_C, \mathcal{F}_C, P_C)$ be the vector of cell vertices, edge midpoints, face midpoints, and the cell midpoint. Then, insert the cells $C_j = (\mathcal{G}(P_{r_{1j}}), ..., \mathcal{G}(P_{r_{Nj}}))$, $j = 1, ..., 2^d$ in the new mesh $\mathcal{N}$.

The new partition map is determined independently and can be computed in advance before the refinement of the cells is realized:

$$\begin{aligned}
&\pi_{\mathcal{N}}(P) = \pi_{\mathcal{M}}(P) \quad \text{for all} \quad P \in \mathcal{P}_{\mathcal{M}},\\
&\pi_{\mathcal{N}}(\tfrac{1}{2}P_{\text{left}} + \tfrac{1}{2}\mathcal{G}(P_E)) = \pi_{\mathcal{N}}(\tfrac{1}{2}\mathcal{G}(P_E) + \tfrac{1}{2}P_{\text{right}}) = \pi_{\mathcal{M}}(P_E)\\
&\qquad\qquad \text{for} \quad E = (P_{\text{left}}, P_{\text{right}}) \in \mathcal{E}_{\mathcal{M}},\\
&\pi_{\mathcal{N}}(\mathcal{G}(P_{F_j})) = \pi_{\mathcal{M}}(P_F) \quad \text{for} \quad F \in \mathcal{F}_{\mathcal{M}},\ j = 1, 2^{d-1}\\
&\pi_{\mathcal{N}}(P_{C_j}) = \pi_{\mathcal{N}}(P_{E_k}) = \pi_{\mathcal{M}}(P_C) \text{ for } C \in \mathcal{C}_{\mathcal{M}},\ j = 1, 2^d \text{ and inner edges,}
\end{aligned}$$

where P_{F_j} and P_{C_j} are the midpoints of the refined face F or cell C.

Example (continued) The cell (P_1, P_2, P_3) is refined (in case of no boundary projections) to four cells $C_1 = (P_1, P_{12}, P_{31})$, $C_2 = (P_2, P_{23}, P_{12})$, $C_3 = (P_3, P_{31}, P_{23})$, $C_4 = (P_{12}, P_{23}, P_{31})$, and we set $\pi_{\mathcal{N}}(P_j) = \pi_{\mathcal{M}}(P_j)$, $\pi_{\mathcal{N}}(P_{jk}) = \pi_{\mathcal{N}}(\frac{1}{2}P_j + \frac{1}{2}P_{jk}) = \pi_{\mathcal{N}}(\frac{1}{2}P_{jk} + \frac{1}{2}P_k) = \pi_{\mathcal{M}}(P_{jk})$, and finally $\pi_{\mathcal{N}}(P_{C_1}) = \pi_{\mathcal{N}}(P_{C_2}) = \pi_{\mathcal{N}}(P_{C_3}) = \pi_{\mathcal{N}}(P_{C_4}) = \pi_{\mathcal{N}}(\frac{1}{2}P_{ij} + \frac{1}{2}P_{jk}) = \pi_{\mathcal{M}}(P_C)$.

4 Parallel Linear Algebra

We assign to every point $P \in \mathcal{P}$ the number of degrees of freedom $N_P \geq 0$, where the point set $\mathcal{P}$ may be extended by nodal points of the finite element discretization. Let $N_{\mathcal{P}} = \sum_{P \in \mathcal{P}} N_P$ be the total number of unknowns.
A vector $\underline{u} \in \mathbf{R}^{N_{\mathcal{P}}} \simeq \prod_{P \in \mathcal{P}} \mathbf{R}^{N_P}$ maps a point P to the vector $\underline{u}[P] \in \mathbf{R}^{N_P}$ of unknowns associated to the point $P \in \mathcal{P}$.
We use two different representation of distributed vectors (cf. Bastian [1996]):

- Solution vectors and correction vectors $\underline{u}$ are represented consistently in

$$V[\mathcal{M}] := \big\{(\underline{u}_1, ..., \underline{u}_{N_{\text{procs}}}) \in \textstyle\prod \mathbf{R}^{N_{\mathcal{P}_q}} : \underline{u}_p[P] = \underline{u}_q[P], \quad p, q \in \pi(P)\big\}.$$

  This defines a global vector $\underline{u}$ by $\underline{u}[P] = \underline{u}_q[P]$ for any $q \in \pi(P)$.
- Right-hand side vectors and residual vectors $\underline{r}$ are represented additively, i. e., any additive vector $(\underline{r}_q) \in \prod \mathbf{R}^{N_{\mathcal{P}_q}}$ represents at $P \in \mathcal{P}$ the value $r[P] = \sum_{q \in \pi(P)} \underline{r}_q[P]$. By collection the distributed values at the master

points and replacing $(\underline{r}_q)$ by $r_q[P] = r[P]$ for $q = \mu(P)$ and $\underline{r}_q[P] = 0$ else results in a unique representation in

$$V(\mathcal{M}) := \big\{(\underline{r}_1, ..., \underline{r}_{N_{\text{procs}}}) \in \prod \mathbf{R}^{N_{\mathcal{P}_q}} : \underline{r}_q[P] = \underline{0}, \quad q \neq \mu(P)\big\}.$$

This allows for the parallel evaluation of the norm $\sqrt{\underline{r}^T \underline{r}} = \sqrt{\sum \underline{r}_q^T \underline{r}_q}$.

In our programming model we define the following parallel operators:

- The stiffness matrix corresponds to an operator $\underline{A}\colon V[\mathcal{M}] \longrightarrow V(\mathcal{M})$. In particular, for the solution vector $(\underline{u}_q) \in V[\mathcal{M}]$ and the right-hand side $(\underline{b}_q) \in V(\mathcal{M})$ the parallel residual $(\underline{b}_q - \underline{A}_q \underline{u}_q) \in V(\mathcal{M})$ is independent of the distribution. For any additive matrix $(\underline{A}_q) \in \prod \mathbf{R}^{N_{\mathcal{P}_q}} \times \mathbf{R}^{N_{\mathcal{P}_q}}$ the application of the parallel operator consists of two steps: compute an additive representation $(\underline{A}_q \underline{u}_q) \in \prod \mathbf{R}^{N_{\mathcal{P}_q}}$ by local matrix-vector multiplication without parallel communication, and then collect the values on $\pi(P)$ for all points P to obtain a vector in $V(\mathcal{M})$.
- A parallel preconditioner corresponds to an operator $\underline{B}\colon V(\mathcal{M}) \longrightarrow V[\mathcal{M}]$. E. g., $\underline{B}[P,P] = \Big(\sum\limits_{q \in \pi(P)} \underline{A}_q[P,P] \in \mathbf{R}^{N_P \times N_P}\Big)^{-1}$ defines the block-Jacobi preconditioner. For the application, in the first step the local computation results in an additive result $(\underline{B}_q \underline{r}_q)$ without parallel communication, and then this is accumulated on $\pi(P)$ to obtain a vector in $V[\mathcal{M}]$.
- We consider transfer operators $\underline{I} = \underline{I}_{\mathcal{M}}^{\mathcal{N}}\colon V[\mathcal{M}] \longrightarrow V[\mathcal{N}]$ prolongating values on a coarse mesh $\mathcal{M}$ to a fine mesh $\mathcal{N}$. For the application, in the first step the local application results in $(\underline{I}_q \underline{u}_q)$ without parallel communication. For conforming discretizations and uniform parallel refinement described in the previous section, the result is consistent in $V[\mathcal{N}]$; in general, this requires local communication.
  The adjoined operator $\underline{I}^T\colon V(\mathcal{N}) \longrightarrow V(\mathcal{M})$ restricts values by first computing an additive result $(\underline{I}_q^T \underline{r}_q)$ without parallel communication and then collecting the values on $\pi(P)$ for all points P to obtain a vector in $V(\mathcal{M})$.

Together with the inner product in $V[\mathcal{M}] \times V(\mathcal{M})$, this allows for a general formulation of parallel iterative solvers such as Krylov methods with multigrid preconditioner, where communication is restricted to the inner products and the application of the parallel operators.

5 Parallel Finite Elements

Corresponding to cell based finite element discretizations we define the cell nodal points by $\mathcal{P}_C := \{P \in \mathcal{P} \cap \bar{\Omega}_C\colon N_P > 0\}$, the cell vector $\underline{u}[C] = \big(x[P]\big)_{P \in \mathcal{P}_C} \in \mathbf{R}^{N_C}$ with $N_C = \sum\limits_{P \in \mathcal{P}_C} N_P$, and the cell matrix $A[C] = \big(A[P,Q]\big)_{P,Q \in \mathcal{P}_C} \in \mathbf{R}^{N_C, N_C}$. In the case of cubic triangular elements we have $N_C = 10$. Again, the isoparametric transformation is determined by the application of the geometry mapping $\mathcal{G}$ to the nodal points.

A nonlinear finite element problem is given by the following assembling routines, running in parallel over all cells $C \in \mathcal{C}_q$ of a mesh $\mathcal{M}$:

- Essential boundary conditions are assigned by $D_C(\underline{u}[C])$ to the corresponding indices $i = 1, ..., N_P$, $P \in \mathcal{P}_C$; after the assembling it has to be guaranteed that this results in parallel consistent Dirichlet values respecting $(\underline{u}_q) \in V[\mathcal{M}]$.
- The additive residual $\underline{r}[C] = R_C(\underline{u}[C])$ is computed depending on the actual solution vector $\underline{u}[C]$; after the assembling the residual has to be collected to obtain a unique additive representation in $V(\mathcal{M})$.
- if the residual norm is not small enough, the Jacobian $A[C] = J_C(\underline{u}[C])$ is assembled additively; in every nonlinear step a consistent correction vector $\underline{c} \in V[\mathcal{M}]$ is computed by solving the linear equation $\underline{Ac} = \underline{r}$ iteratively up to a given accuracy, and the solution vector is updated by $\underline{u} := \underline{u} + \underline{c}$.

6 A numerical experiment

We present a numerical approximation of the quasi-linear nonlinear model problem problem (due to Gelfand and Bratu)

$$u \in H_0^1(\Omega): \qquad -\Delta u = \lambda \exp(u)$$

with cubic isoparametric finite elements. It is well known that this equation admits at least two solutions for $\lambda \in (0, \lambda^*)$, with a turning point at the critical parameter λ^* (see, e. g., Lions [1982]).

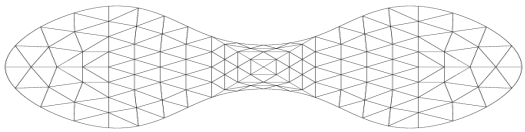


Fig. 1. A nonconvex domain Ω, where the boundary $\partial\Omega$ is of class $C^{2,1}$ (represented by a cubic spline).

While analytical tools provide the existence of only two solutions, it could be shown by a computer assisted existence proof in Plum and Wieners [2002] for a special nonconvex domain, that a further (symmetry breaking) solution branch exists. Such a solution is illustrated in Fig. 2. For this method (showing the existence of a continuous solution via Schauders fixpoint theorem), extremely smooth and accurate approximations are required.

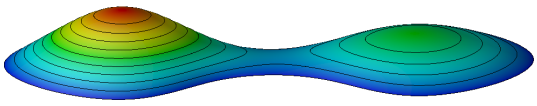


Fig. 2. Symmetry breaking solution of the Gelfand problem for $\lambda = 0.45$, computed in parallel with isoparametric cubic elements after 6 geometry preserving refinement steps of the mesh in Fig. 1.

References

P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Teubner Skripten zur Numerik. Teubner, Stuttgart, 1996.

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.

P. Bastian, K. Birken, K. Johannsen, S. Lang, V. Reichenberger, H. Rentz-Reichert, C. Wieners, and G. Wittum. A parallel software-platform for solving problems of partial differential equations using unstructured grids and adaptive multigrid methods. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering '98*, pages 326–339, Berlin, 1998. Springer-Verlag.

P. Bastian, K. Johannsen, S. Lang, S. Nägele, V. Reichenberger, C. Wieners, G. Wittum, and C. Wrobel. Advances in high-performance computing: Multigrid methods for partial differential equations and its applications. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering '99*, pages 506–519, Berlin, 1999. Springer-Verlag.

P. Bastian, K. Johannsen, S. Lang, V. Reichenberger, C. Wieners, and G. Wittum. High-accuracy simulation of density driven flow in porous media. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering '01*, pages 500–511, Berlin, 2001. Springer-Verlag.

P. Bastian, K. Johannsen, S. Lang, V. Reichenberger, C. Wieners, G. Wittum, and C. Wrobel. Parallel solutions of partial differential equations with adaptive multigrid methods on unstructured grids. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering '00*, pages 496–508, Berlin, 2000. Springer-Verlag.

S. Lang, C. Wieners, and G. Wittum. The application of adaptive parallel multigrid methods to problems in nonlinear solid mechanics. In E. Stein, editor, *Error-Controlled Adaptive Finite Element Methods in Solid Mechanics*, pages 347–384, New-York, 2002. Wiley.

P. L. Lions. On the existence of positive solutions of semilinear elliptic equations. *SIAM Review*, 24:441–467, 1982.

M. Plum and C. Wieners. New solutions of the Gelfand problem. *J. Math. Anal. Appl.*, 269:588–606, 2002.

C. Wieners. Multigrid methods for Prandtl-Reuß plasticity. *Numer. Lin. Alg. Appl.*, 6:457–478, 1999.

C. Wieners. Efficient elasto-plastic simulation. In A.-M. Sändig, W. Schiehlen, and W. L. Wendland, editors, *Multifield problems. State of the art*, pages 209–216, Berlin, 2000a. Springer-Verlag.

C. Wieners. *Theorie und Numerik der Prandtl-Reuß-Plastizität.* Universität Heidelberg, 2000b. Habilitationsschrift.

C. Wieners, M. Ammann, and W. Ehlers. Distributed point objects: A new concept for parallel finite elements applied to a geomechanical problem. 2003. Submitted to Future Generation Computer Systems.