14th Finite Element Workshop — Ulm, July 2007

Finite Element Simulation of Bone Microstructures

Tobias Preusser [*] Martin Rumpf [†] Lars Ole Schwen[†]

The geometric construction of finite element spaces suitable for complicated shapes or microstructured materials is discussed. As an application, the efficient computation of linearized elasticity is considered on them. Geometries are supposed to be implicitly described via 3D voxel data (e. g. µCT scans) associated with a cubic grid. We place degrees of freedom only at the grid nodes and incorporate the complexity of the domain in the hierarchy of finite element basis functions, i. e. constructed by cut off operations at the reconstructed domain boundary. Thus, our method inherits the nestedness of uniform hexahedral grids while still being able to resolve complicated structures. In particular, the canonical coarse scales on hexahedral grid hierarchies can be used in multigrid methods.

AMS Subject Classifications: 65N30, 65N55, 65N50.

1 Introduction

Three-dimensional objects with complicated microstructure are a challenge in FE simulations. For classical FE, the two main difficulties are to find a suitable mesh (typically a non structured mesh) and to provide sufficient computational resources for setting up and solving the system of equations obtained by discretizing the partial differential equations (PDEs) associated to the physical process under consideration.

High quality tetrahedral mesh generation has been worked on for many years and still is a nontrivial problem. We refer to [10, 49] for an overview on methods for mesh generation and still challenging problems, [42] for an overview of mesh quality measures and [3] for an efficient, recent method.

The main disadvantage of unstructured FE meshes is that (unlike for uniform, e. g. cubic meshes) explicit storage of the location of grid points and the connectivity structure is necessary and that there are no canonical coarse versions of the mesh. The corresponding systems of equations are sparse but the sparsity structure also needs to be stored.

For uniform cubic meshes, geometric locations and connectivity information of grid points and sparsity structure of systems of equations are known implicitly which saves a big amount of memory, moreover canonical coarse scales allow to use multigrid (MG) methods based on the geometric hierarchy. Multigrid methods are solvers for linear systems of equations that are of optimal computational complexity and have proved to be very efficient in many applications. In case of

[*]Center of Complex Systems and Visualization, University of Bremen, tp@cevis.uni-bremen.de
[†]Institute for Numerical Simulation, University of Bonn, {martin.rumpf,ole.schwen}@ins.uni-bonn.de

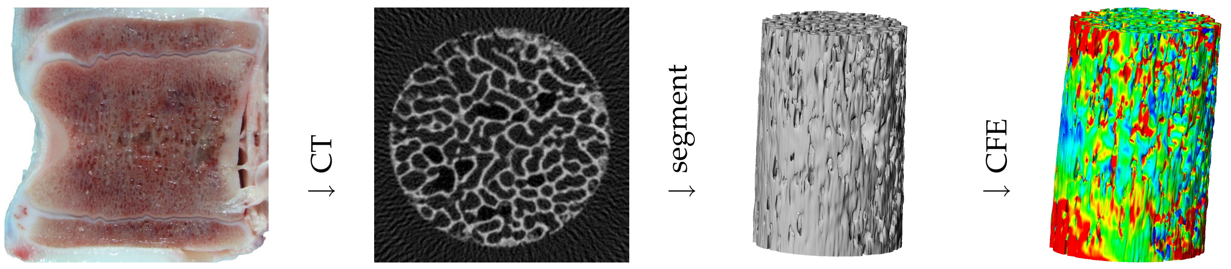


Figure 1: The image based computing pipeline is depicted: From the anatomical structure of interest, a sample is extracted and scanned in μ-CT. A three dimensional model for the boundary of the object is obtained by image-segmentation. Finally, a physical simulation is performed on the perviously segmented object.

unstructured meshes, where purely geometric coarsening is impossible, so-called algebraic multigrid methods (AMG) can be used. AMG only makes use of the sparsity structure in the matrix, which frequently is not the optimal approach if geometric information is available. We refer to [12] where MG was introduced and [13] for an introduction to AMG, [53, 47, 14] are overviews of different methods and applications of (A)MG.

Here, we make use of the composite finite element (CFE) concept first introduced in [22, 23]. The main idea is to incorporate the complexity of the microstructure in the FE basis functions (rather than in the mesh) and use an underlying structured cubic mesh. This way, we are able to use efficient data structures for structured meshes and corresponding geometric multigrid solvers.

After a review of related methods, we explain the CFE method and a corresponding MG method presented in [35] and show some results obtained the implementation of linearized elasticity, where many technical details were originally implemented in [34].

2 Review of related methods

The idea of using regular grids for complicated domains has been used in a number of different methods. We present a few of these, not claiming this list to be exhaustive.

Many of those methods deal with the case of a two-phase material with complicated interface and different material parameters. This is a more general setting because the case of a single object of complicated shape can be viewed as a special case (the limiting case) of an object embedded in an "infinitely soft" or "infinitely insulating" material. However, we consider CFE for objects with complex shape as a problem in its own right and treat discontinuous coefficients across complicated interfaces as a separate problem [41].

A similar method called the "Immersed Interface Method" (IIM) was developed by LeVeque, Li and others starting in the 1990s. It was first used for finite difference computations where a Cartesian grid for complicated interfaces requires adapting the finite difference stencils near the interface. The IIM for discontinuous coefficients and possibly singular sources on the interface can be found in [11, 27] in 1D / 2D and summarized in Li's PhD dissertation [28], he extends it to 3D in [29] and presents an efficient solver in [30].

Adams and Li present an MG solver for the IIM [1, 2], however they do not deal with the type of problems we encountered in our multigrid scheme. An overview of the applications of IIM can be found in [32]. Calhoun and LeVeque combine the IIM with a finite volume method using "capacity functions" for partially filled cells in [16, 15]. Wiegmann and Bube apply the IIM to nonlinear problems in 1D [52] and modify the method to the "Explicit Jump IIM" considering not only the discontinuities in the coefficient but also the expected singularities in the solution. Li builds the bridge to the finite element world by using "Immersed Finite Elements" in 1D and 2D in [31, 33].

Babuška and others start with the "Partition of Unity Method" [37, 7] (PUM) combining the partition of unity subordinate to a finite cover of the object (classically supports of FE basis functions) with a priori knowledge about the solution (discontinuities at interfaces) to obtain special PUM finite element spaces. In the "Generalized Finite Element Method" (GFEM) [36, 45, 44, 46, 6, 43], the PUM and classical FEM basis functions are used together to improve approximation.

Belytschko and others use so-called "Extended Finite Element Methods" (XFEM) [20, 9] starting from classical FEM and "enriching" the FE spaces by additional basis functions to incorporate discontinuities. Their meshes do not depend on the location of the discontinuities, but the enrichment does introduce additional unknowns. An important application of XFEM is the simulation of crack growth [38, 19, 48, 26] where one tries to avoid repeated remeshing which is necessary in classical approaches. Other applications of XFEM are found in [51, 18, 50].

Finally, we would like to mention the "Finite Cell Method" [39] which is based on the idea of extending the PDE outside the actual object domain such that a domain-independent mesh for FEM can be used.

3 Composite finite elements

In this Section, we discuss the method of [35] in the one– and two–dimensional case which most easily explains the underlying ideas. For the actual application in 3D, we discuss the issue of condition numbers. For a more detailed and algorithmic description of the three-dimensional case, we refer to [35].

Consider an object with complicated boundary[1] in 1D as the red line segments shown in Fig. 2. The construction of CFE basis functions works as follows:

Step 1. We define an equidistant (and in particular domain-independent) grid (black dots) on which we are going to place our unknowns (green circles). Unknowns are placed at all nodes inside the object and on one layer of nodes outside the object.

Step 2. Next, we add the "virtual nodes" (not separately shown) at the object boundary to obtain the "virtual grid". On this grid, we imagine piecewise linear basis functions (cyan hat functions) forming the "virtual basis".

Step 3. As weighted sums of the virtual basis functions with zero weight outside the object, we finally obtain the CFE basis functions (blue hat functions). The weights are chosen such that the basis functions coincide with the standard piecewise linear basis inside the object.

3.1 Construction of basis functions in 1D

By construction, these basis functions satisfy the following properties: They are piecewise linear; they are "nodal", i. e. at each grid point in the interior, exactly one basis function has value 1 whereas all others have value 0; and they form a "partition of unity", i. e. at each point in the interior (not necessarily a grid point), all basis functions sum up to 1.

A hierarchy of coarsened grids (starting on the finest equidistant grid) is defined in the obvious way: two fine cells form one coarse cell. This process introduces new degrees of freedom: "one layer outside the object" on the coarse grid may be wider than one layer on the fine grid, see Fig. 3.

[1]Note that literally there is not much space in 1D for geometric complexity and that disconnectedness is not the generic case in higher space dimension.

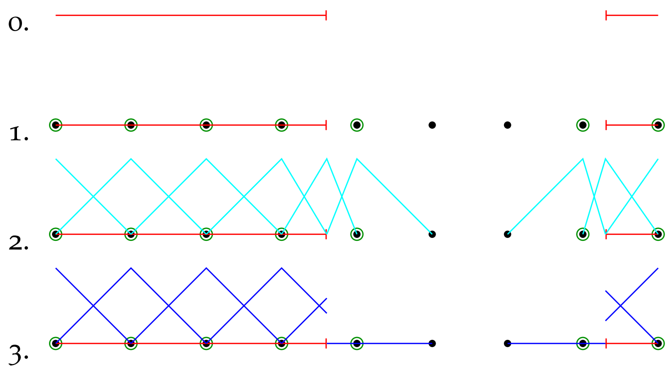


Figure 2: CFE in 1D: (from top to bottom) a "complicated" object, degrees of freedom associated to nodes on an equidistant grid, the "virtual basis" and the CFE basis functions.

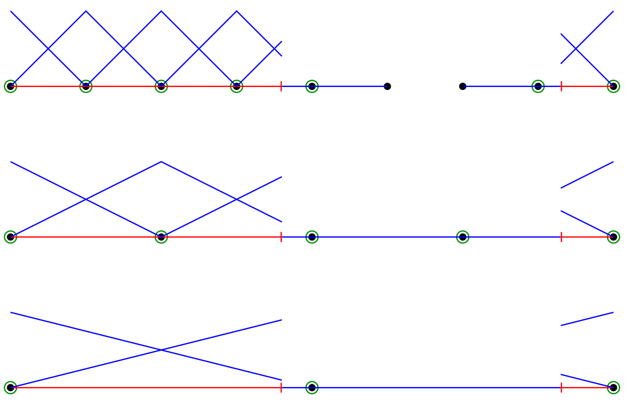


Figure 3: Coarsening in 1D: Canonical coarsening of the grid introduces new DOFs.

This figure also shows the coarse-grid basis functions obtained by the method described later in Sect. 4.

3.2 Construction of basis functions in 2D

Computational domains extracted from images. In our applications, we do not have an exact analytic description of our object and/or its boundary, but only a discrete (CT) image. The uniform quadrilateral grid is immediately given by the pixel resolution of the image.

As input for our CFE method, we suppose the boundary of a physical domain to be given as the zero level set of a function $\Phi : \mathbb{R}^d \to \mathbb{R}$. This function shall be continuous and strictly negative in the interior of the domain. In explicit, the preimage of 0 under Φ is assumed to be a strictly lower-dimensional subset of $\mathbb{R}^d$.

In general, Φ is obtained from the image data via some pre-processing and segmentation steps. In our computations (cf. Sect. 5) we first denoise the data applying a couple of time steps of an edge-perserving anisotropic diffusion [40]. Given a suitable threshold value, this is substracted from the function Φ, i. e. we subtract a threshold from the gray-values of the smoothed CT-image. Finally, we usually invert the resulting gray values, because the original CT image intensity is high inside interesting anatomic structures due to x-ray attenuation.

To illustrate the segmentation process, Fig. 4 shows an exemplary object (brown), a (noisy) pixel

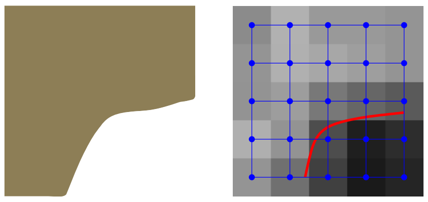


Figure 4: An object in 2D and its boundary recovered in a pixel image of the object.

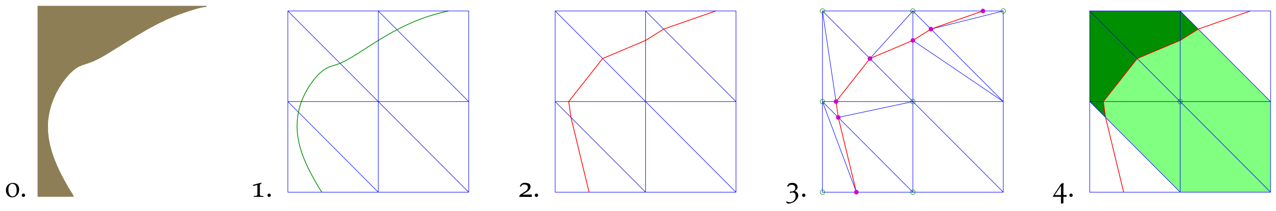


Figure 5: 2D CFE construction: the actual object, a uniform grid underlying standard piecewise linear FE, the recovered object boundary, the two-dimensional "virtual grid" and the support of a CFE basis function.

image of the object, the corresponding uniform quadrilateral grid (blue) and the recovered object boundary (red line).

Basis functions in 2D. Generalizing the construction of CFEs in 1D above, we now deal with piecewise linear basis functions in 2D. An example is shown in Fig. 5:

Step 1. For piecewise linear FE in 2D, we need to subdivide the equidistant quadrilateral grid in "regular triangles" (blue lines). Here, this is done in such a way that we do not cut through the node with smallest global index for inverse lexicographical ordering. The object boundary is shown as the green line and the object lies on the top left.

Step 2. On the edges of the regular triangles, we determine the zero-crossings of Φ to obtain the approximate object boundary (red polygonal line).

Step 3. Adding those intersection points ("virtual nodes", magenta dots) and subdividing the resulting quadrilaterals, we obtain the virtual grid on which we imagine the "virtual basis functions".

Step 4. Weighted sums of the virtual basis functions again define the CFE basis functions. The figure shows the support of the CFE basis function for the central node in dark green, whereas in the light green region (the remaining support of the standard piecewise linear FE basis function at that point) the CFE basis function is zero.

Note that the virtual grid in step 3 could also be used as an unstructured FE grid. The generation is fully automatic and the resulting grid is well-structured inside the object away from the boundary. However, as the image suggests, triangles in the virtual grid can become arbitrarily bad in almost any quality measure.

Again, a grid hierarchy can be defined in the canonical way: four fine level cells form one coarse cell, or, in terms of basis functions, seven fine basis functions (one central and six surrounding ones) are combined to one coarse basis function.

Let us emphasize that we are only interested in coarsening and not in refinement because our finest computational grid is given by the image data which do not want to super sample.

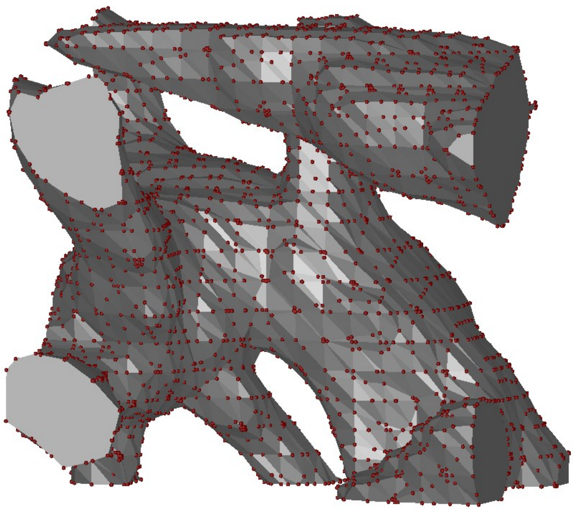


Figure 6: Virtual nodes (red dots) for a complicated 3D object. Flat shading and the location of the virtual nodes show the hexahedral structure of the underlying grid.

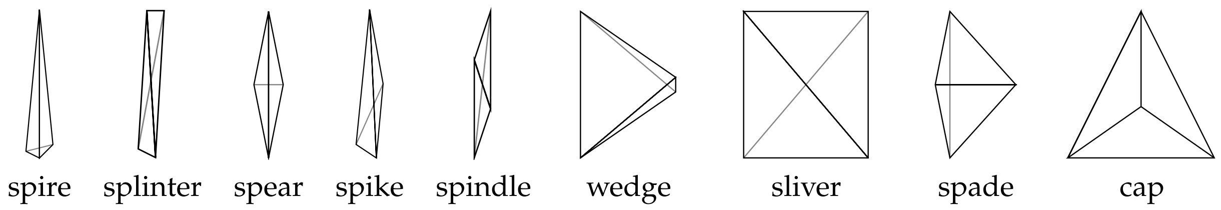


Figure 7: The "zoo" of badly shaped tetrahedra in [17].

3.3 Construction of basis functions in 3D

In three space dimensions, the situation is more technical but not significantly more complicated. We point out some analogies to the two-dimensional case and refer to [35] for more details.

Image data. All slices of the CT image are used as one 3D voxel data set. Denoising and segmentation are performed on the 3D data set.

Regular grid. Our degrees of freedom are now placed on an equidistant cubic grid. Each cube consists of six regular tetrahedra and the union of such regular tetrahedra incident to a given node is the support of the corresponding standard piecewise linear basis function.

Virtual grid. The location of the interface is reconstructed on the edges of the regular tetrahedra, again resulting in virtual nodes. An example is shown in Fig. 6.

3.4 Condition numbers

In our sketches of the virtual grids in 2D, one notices that badly shaped triangles can occur. In fact, in the 3D application, there is no lower bound on the aspect ratio of the virtual tetrahedra. Following the classification of [17], the geometric construction of the virtual nodes allows spires and splinters (but no spears, spikes and spindles) as well as wedges, slivers and spades (but no caps), see Fig. 7.

So if we used the (unstructured) virtual grid (with more unknowns than for the CFE grid), we expect arbitrarily large condition numbers of the matrices arising. The condition number κ of a

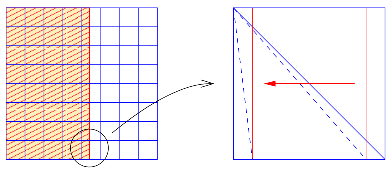


Figure 8: Brick Object for which we determine condition numbers of the corresponding CFE systems of equations.

Matrix A, being defined as

$$\kappa(A) := \frac{\text{largest absolute eigenvalue of } A}{\text{smallest absolute eigenvalue of } A} \tag{1}$$

is an indicator how fast iterative solvers converge. For the Conjugate Gradient (CG) method [25], we have

$$\|e_{k+1}\| \leq \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\|e_k\|, \tag{2}$$

hence the relative decrease η of the residual in one step of the iterative solver (= convergence rate) is high for large condition numbers. A convergence rate $\eta \gtrsim 0$ means fast convergence, $\eta \lesssim 1$ means slow convergence and $\eta > 1$ means divergence. Note that for the CG method, the estimate (2) gives an upper bound and that the speed of convergence in practice also depends on the distribution of the eigenvalues. Techniques like preconditioning [5, 8] can help improve convergence, but only to a certain extent.

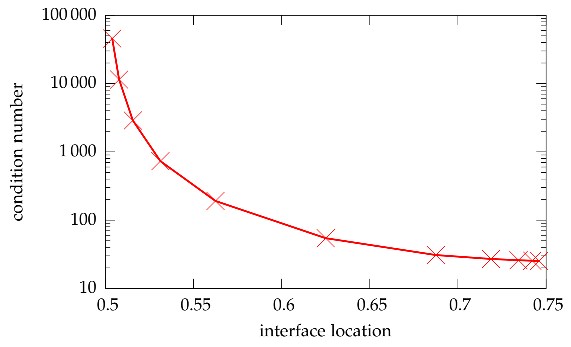
A case study of the condition. To see the influence of bad tetrahedra for the CFE systems of equations, let us consider a brick-shaped object that has size 1 in y and z direction and variable size in x direction, see Fig. 8. We shift the interface in x direction from covering almost one full layer of cubic cells to only a very small part of that layer. We refer to this part as the fractional part of the width of the domain measured in grid cells or fractional width. Fig. 8 shows that the size and the shape of the tetrahedra changes with the fractional width.

For this test we consider a diffusion problem $\partial_t - \Delta u = f$ with homogeneous Neumann boundary condition. We use grids with 5^3 and 17^3 nodes corresponding to grid widths $h = \frac{1}{4}$ and $h = \frac{1}{16}$, respectively. Furthermore, we apply a implicit Euler scheme in time with a time step $\tau = h$.

First, we set up the corresponding CFE matrices and use the numerical computing system OCTAVE [21] to compute the condition numbers of these matrices. Second, we solve the systems of equations to an accuracy of 10^{-8} (relative to the initial residual) using a CG solver and count the number of iterations. Each iterations requires $O(\#\text{DOF})$ flops, so the number of iterations is proportional to the cputime.

In Fig. 9 we show the condition number and the number of CG iterations versus the fractional width. We see from the figures that for fractional width close to 1, small virtual tetrahedra in the exterior and badly shaped interior virtual tetrahedra do not affect the condition number of the systems. Furthermore, for small fractional width close to 0, small and badly shaped interior virtual tetrahedra lead to big condition numbers. As expected, the CG method turns slower for bigger condition numbers. Note that the increase of CG iterations is slower than Eq. (2) might suggest.

For these simple geometries, (diagonal) preconditioning does actually help, it can almost compensate the influence of increasing badness of the tetrahedra on the number of PCG iterations. However, for more complicated objects, simple preconditioning techniques like diagonal, block diagonal, ILU0 and SSOR preconditioning have not proven to be effective.



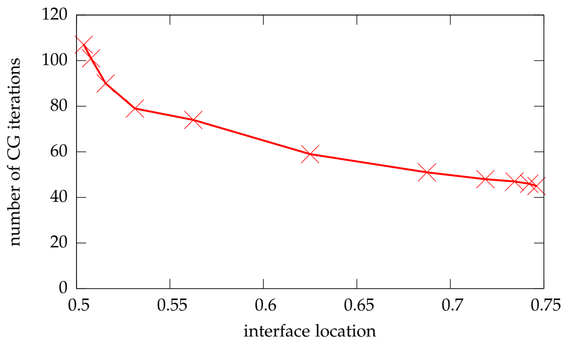


Figure 9: Condition numbers and number of CG iterations for the time discrete diffusion problem on the brick geometries shown in Fig. 8.

4 A multigrid solver for CFE matrices

Multigrid methods are renown as efficient methods to solve PDE problem and in particular to resolve bad condition of the corresponding linear systems. Their construction is based on the following observation [12]: simple iterative methods like Jacobi and Gauß-Seidel iterations quickly reduce high frequency components of the error between the iterates and true solution whereas low frequency components are damped very slowly. But these components could be reduced more quickly by the same type of method for an approximation on a coarser grid where their relative frequency is high.

Let us recall the basic structure of a two-grid method for solving a system $Ax = b$ with starting point x^0. One iterates

Step 1. take a fixed number of iterations on the original system which gives an approximation x^k ("presmoothing")

Step 2. Compute the residual $r^k = b - Ax^k$

Step 3. Restrict the residual r^k on a coarser grid by $\tilde{r} = \mathcal{R}(r)$

Step 4. Solve $\tilde{A}\tilde{e} = -\tilde{r}$ for the discretization of the coarser problem

Step 5. Prolongate (interpolate) $\tilde{e}$ on the finer grid to obtain $e^k = \mathcal{P}(\tilde{e})$, a correcting term for x^k ("coarse grid correction")

Step 6. Take $x^k - e^k$ as a starting point for more iterations of the iterative method on the finest grid ("postsmoothing")

until the residual is sufficiently small. Again taking a two-grid method in step 4, we obtain a multigrid method.

So we need the following building blocks:

Iterative method. In our computations, standard (block) Gauß-Seidel iterations, two for pre- and postsmoothing, respectively, turned out to be a good choice.

Restriction operator $\mathcal{R}$. We take the transpose of the prolongation operator (see below) as the restriction operator.

Coarsening of the problem $\tilde{A}$. To obtain the coarsened problem matrix, we use the standard Galerkin product $\tilde{A} = \mathcal{R}A\mathcal{P}$ [24].

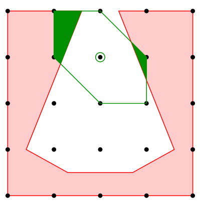


Figure 10: The support of this coarse grid basis function has two disconnected components lying in physically weakly related parts of the object (“horseshoe”-like geometry).

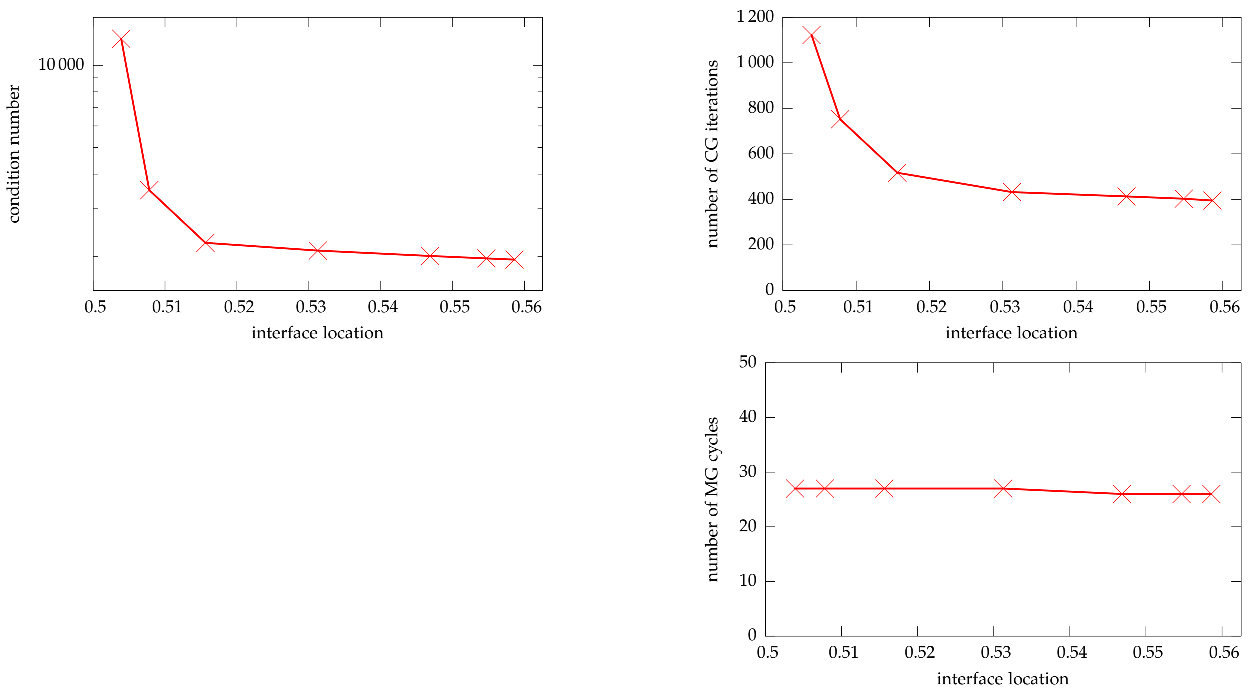


Figure 11: Condition numbers, number of CG iterations and number of MG cycles for the elasticity problem on the brick geometries shown in Fig. 8.

Prolongation operator $\mathcal{P}$. The prolongation weights are obtained by evaluating the coarse grid basis functions at the fine grid nodes. As we are dealing with underlying piecewise linear FE, these weights are $1, \frac{1}{2}$ and 0.

The restriction and prolongation explained above also determine coarse grid basis functions. Even though these are not needed and not computed explicitely, we state some properties. Being weighted sums of fine grid basis functions, coarse grid basis functions are again piecewise linear, and the weights are such that the coarse grid basis functions are nodal and form a partition of unity.

Individual coarse grid basis functions may have disconnected support because coarse grids in general cannot resolve the structure or topology of our complicated object. This is problematic if those components correspond to parts of the object that are physically only very weakly related, as shown in the “horseshoe”-like geometry in Fig. 10. This leads to poor coarse grid corrections in the multigrid scheme and to slow convergence. An improvement of the coarsening scheme is currently investigated to avoid such artificial numerical coupling and to significantly speed up convergence.

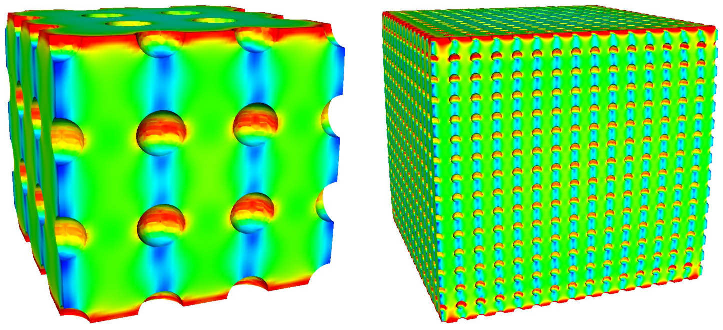


Figure 12: Compression of two "cheese"-like datasets. The color shading from blue to red in the visualization indicates von Mises surface stresses.

5 Applications and results

On a domain $\Omega \subset \mathbb{R}^3$, let us consider linear elasticity modeled by the Lamé-Navier-Equations with Dirichlet boundary conditions and without volume forces. Given the elastic energy

$$\begin{aligned} E_{\text{elast}}[\mathbf{u}] &= \frac{1}{2}\int_\Omega \mathcal{E}(\mathbf{u}) : \mathcal{C}\mathcal{E}(\mathbf{u}) \quad \text{with} \\ \mathcal{E}(\mathbf{u}) &= \frac{1}{2}[\nabla \mathbf{u} + \nabla \mathbf{u}^T] \quad \text{and} \\ \mathcal{C}_{ijmn} &= \lambda \delta_{ij}\delta_{mn} + \mu[\delta_{im}\delta_{jn} + \delta_{in}\delta_{jm}], \end{aligned} \tag{3}$$

we search for displacement fields $\mathbf{u} : \Omega \to \mathbb{R}^3$ which minimize the elastic energy. Solutions to this minimization problem lie in the Sobolev space $H^1(\Omega, \mathbb{R}^3)$ [4] and obey the equation

$$\int_\Omega \lambda \operatorname{div}\mathbf{u} \operatorname{div}\mathbf{v} + 2\mu\, \mathcal{E}(\mathbf{u}) : \mathcal{E}(\mathbf{v}) = 0 \qquad \forall \mathbf{v} \in H^1(\Omega, \mathbb{R}^3). \tag{4}$$

First we consider the same test geometry as in the last section. Again we assemble the CFE system matrix and compute the condition number using the numerical computing system OCTAVE. In Fig. 11 we depict the condition number of the system-matrix as well as the iteration numbers for the CG- and the multigrid-method versus the fractional width. As before, we see an increase of the iteration count for the CG-method whereas the number of MG cycles merely changes from 28 to 29 in this example. The number of MG cycles is proportional to the cputime needed by the multigrid-solver.

In a second example, we consider the compression of a "cheese"-like domain resolved on a 65^3 grid, using $3 \times 258\,064$ DOFs. Requiring about 560 MB of memory, this simulation can be run on a regular desktop PC. The multigrid convergence is fairly good in this example (convergence rate 0.548) and on an Intel P4 3.6 GHz, the computation took 193 seconds for a solver accuracy of 10^{-8}. The resulting von Mises surface stresses are shown in Fig. 12. Fig. 12 also shows the same type of geometry with significantly finer structures resolved on a 257^3 grid, using $3 \times 16\,458\,648$ DOFs. Requiring about 30 GB of memory, this simulation cannot be run on a standard PC. The multigrid convergence rate of 0.552 was equally good and the solver took 10 411 seconds, less than three hours, on an Opteron 1.8 GHz processor.

Further examples are the compression of an array of $20 \times 20 \times 20$ rods, resolved on a 257^3 grid, using $3 \times 5\,028\,836$ DOFs and the same geometry with a random 10 percent of the connections removed, using $3 \times 4\,653\,815$ DOFs. These simulations require about 30 GB of memory and show poor convergence rates (0.977, 0.996, resulting in approximately 1.4 and 4 days of cputime on an Opteron 1.8 GHz processor. The resulting von Mises stresses are shown in Fig. 13.

Finally, we consider the shearing of a bone dataset (cylindrical sample of a porcine T1 vertebral bone), again resolved on a 257^3 grid using $3 \times 3\,374\,720$ DOFs and with a memory requirement of about 30 GB. The multigrid convergence is even worse (0.999) and the computation took about 7.6 days. The results of this computation are shown in Fig. 14.

6 Summary and Outlook

We have demonstrated that composite finite elements allow the effective simulation of elastic microstructured bone material. Computational efficiency, however, still suffers from slow convergence of the multigrid solver. This is currently being investigated. As for the two geometries in Fig. 13, a more detailed parameter study is planned.

In cooperation with Uwe Wolfram at the Institute for Biomechanics and Orthopaedic Research at University of Ulm, the experimental validation of our elasticity simulations of Aluminum foams and trabecular bone specimens is currently investigated.

Furthermore, the CFE construction and implementation is going to be generalized to the case of two-phase materials with discontinuous material coefficient across a complicated interface.

Acknowledgments

The authors would like to thank Annette Kettler (Institute of Biomechanics and Orthopaedic Research, University of Ulm) for the photo in Fig. 1, and Uwe Wolfram (Institute of Biomechanics and Orthopaedic Research, University of Ulm) for the CT scan of the trabecular sample in Fig. 14. Furthermore, we thank Stefan Sauter (Institute of Mathematics, University of Zurich) for his continuous advice concerning composite finite element methodology.

References

[1] Loyce Adams. A multigrid algorithm for immersed interface problems. In *Proceedings of the Seventh Copper Mountain Conference on Multigrid Methods*, pages 1–14, 1996. SEE N97-13750 01-64.

[2] Loyce Adams and Zhilin Li. The immersed interface / multigrid methods for interface problems. *SIAM J. Sci. Comput.*, 24(2):463–479, 2002.

[3] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, July 2005.

[4] Hans Wilhelm Alt. *Lineare Funktionalanalysis*. Springer, 2002.

[5] O. Axelsson. A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT Numerical Mathematics*, 25(1):165–187, March 1985.

[6] A. O. Ayhan and H. F. Nied. Stress intensity factors for three-dimensional surface cracks using enriched finite elements. *International Journal for Numerical Methods in Engineering*, 54:899–921, 2002.

[7] I. Babuška and J. Melenk. The partition of unity method. *Int. J. Numer. Meths. Eng.*, 40:727–758, 1997.

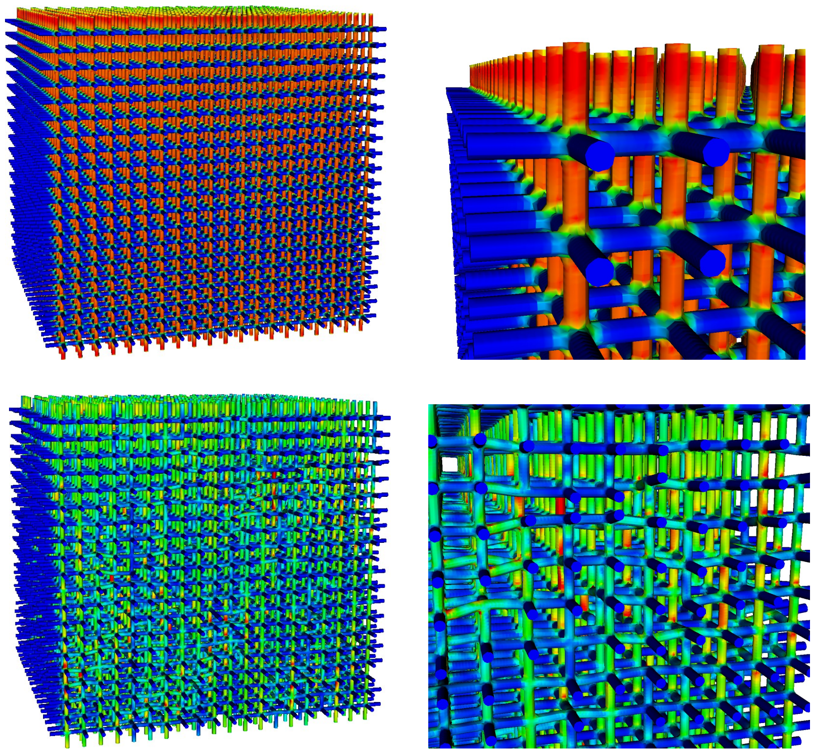


Figure 13: Compression of $20 \times 20 \times 20$ rods and zoom to the top left corner (top row) and the same geometry with ten percent of the connections removed and zoom to an "interesting region" (bottom row).

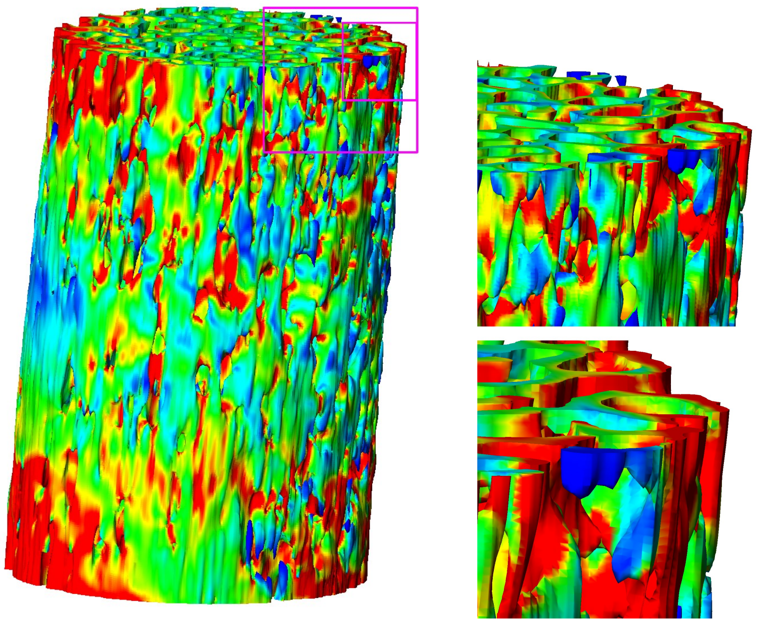


Figure 14: Shearing of a bone dataset and zoom to the top right corner.

[8] Zhong-Zhi Bai. A class of modified block SSOR preconditioners for symmetric positive definite systems of linear equations. *Advances in Computational Mathematics*, 10:169–186, 1999.

[9] T. Belytschko, N. Moës, S. Usui, and C. Parimi. Arbitrary discontinuities in finite elements. *International Journal for Numerical Methods in Engineering*, 50(4):993–1013, 2001.

[10] Marshall Bern and David Eppstein. *Mesh generation and optimal triangulation*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, Singapore, 1992.

[11] R. P. Beyer and R. J. LeVeque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM J. Numer. Anal.*, 29(2):332–364, 1992.

[12] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[13] Achi Brandt. General highly accurate algebraic coarsening. *Electronic Transaction on Numerical Analysis*, 10:1–20, 2000.

[14] Achi Brandt and Dorit Ron. *Multigrid Solvers and Multilevel Optimization Strategies*, volume 14 of *Combinatorial Optimization*, chapter 1, pages 1–69. Kluwer Academic Publishers, 2002.

[15] Donna Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of Computational Physics*, 176:231–275, 2002.

[16] Donna Calhoun and Randall J. LeVeque. A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries. *Journal of Computational Physics*, 157:143–180, 2000.

[17] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver exudation. *Journal of the ACM*, 47(5):883–904, September 2000.

[18] Jack Chessa, Patrick Smolinski, and Ted Belytschko. The extended finite element method (XFEM) for solidification problems. *International Journal for Numerical Methods in Engineering*, 53:1959–1977, 2002.

[19] Christophe Daux, Nicolas Moës, John Dolbow, Natarjan Sukumar, and Ted Belytschko. Arbitrary branched and intersecting cracks with the extended finite element method. *International Journal for Numerical Methods in Engineering*, 48:1741:1760, 2000.

[20] John Dolbow. *An Extended Finite Element Method with Discontinuous Enrichment for Applied Mechanics*. Dissertation, Northwestern University, 1999.

[21] John W. Eaton et al. GNU Octave, version 2.9.12. http://www.octave.org, 2007.

[22] W. Hackbusch and S. Sauter. Composite finite elements for the approximation of PDEs on domains with complicated micro-structures. *Numerische Mathematik*, 75:447–472, 1997.

[23] W. Hackbusch and S. A. Sauter. Composite finite elements for problems containing small geometric details. Part II: Implementation and numerical results. *Comput. Visual. Sci.*, 1(1):15–25, 1997.

[24] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer, 1985.

[25] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[26] R. Huang, N. Sukumar, and J.-H. Prévost. Modeling quasi-static crack growth with the extended finite element method. Part II: Numerical applications. *International Journal of Solids and Structures*, 40(26):7539–7552, 2003.

[27] Randall J. LeVeque and Zhi Lin Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31(4):1019–1044, 1994.

[28] Zhilin Li. *The Immersed Interface Method - A Numerical Approach for Partial Differential Equations with Interfaces*. Dissertation, University of Washington, Seattle, WA, USA, 1994.

[29] Zhilin Li. A note on immersed interface method for three-dimensional elliptic equations. *Computers and Mathematics with Applications*, 31(3):9–17, February 1996.

[30] Zhilin Li. A fast iterative algorithm for elliptic interface problems. *SIAM Journal on Numerical Analysis*, 35(1):230–254, 1998.

[31] Zhilin Li. The immersed interface method using a finite element formulation. *Applied Numerical Mathematics*, 27:253–267, 1998.

[32] Zhilin Li. An overview of the immersed interface method and its applications. *Taiwanese Journal of Mathematics*, 7(1):1–49, March 2003.

[33] Zhilin Li, Tao Lin, and Xiaohui Wu. New Cartesian grid methods for interface problems using the finite element formulation. *Numerische Mathematik*, 1996(1):61–98, November 2003.

[34] Florian Liehr. Ein effizienter Löser für elastische Mikrostrukturen. Diploma thesis, University Duisburg, 2004.

[35] Florian Liehr, Tobias Preusser, Martin Rumpf, Stefan Sauter, and Lars Ole Schwen. Composite finite elements for 3D image based computing. *Submitted to Computing and Visualization in Science*, 2007.

[36] Jens Markus Melenk. *On Generalized Finite Element Methods*. Dissertation, University of Maryland, 1995.

[37] Jens Markus Melenk and Ivo Babuška. The partition of unity finite element method. Research Report 96-01, Eidgenössische Technische Hochschule Zürich, Seminar für angewandte Mathematik, January 1996.

[38] Nicolas Moës, John Dolbow, and Ted Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46:131–150, 1999.

[39] Jamshid Parvizian, Alexander Düster, and Ernst Rank. Finite cell method. *Computational Mechanics*, Online First, 2007.

[40] T. Preußer and M. Rumpf. An adaptive finite element method for large scale image processing. *Journal of Visual Comm. and Image Repres.*, 11:183–195, 2000.

[41] Tobias Preusser, Martin Rumpf, Stefan Sauter, Lars Ole Schwen, et al. Three-dimensional composite finite elements for scalar problems with jumping coefficients. 2007. in preparation.

[42] Jonathan Richard Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proceedings of the 11th International Meshing Roundtable*, pages 115–126. Sandia National Laboratories, September 2002.

[43] F. L. Stazi, E. Budyn, J. Chessa, and T. Belytschko. An extended finite element method with higher-order elements for curved cracks. *Computational Mechanics*, 31:38–48, 2003.

[44] M. Stolarska, D. L. Chopp, N. Moës, and T. Belytschko. Modelling crack growth by level sets in the extended finite element method. *International Journal for Numerical Methods in Engineering*, 51:943–960, 2001.

[45] T. Strouboulis, I. Babuška, and K. Copps. The design andd analysis of the Generalized Finite Element Method. *Comput. Methods Appl. Mech. Engrg.*, 181:43–69, 2000.

[46] T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method. *Comput. Methods Appl. Mech. Engrg.*, 190:4081–4193, 2001.

[47] Klaus Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128:281–309, 2001.

[48] N Sukumar and J.-H. Prévost. Modeling quasi-static crack growth with the extended finite element method. Part I: Computer implementation. *International Journal of Solids and Structures*, 40(26):7513–7537, 2003.

[49] Shang-Hua Teng and Chi Wai Wong. Unstructured mesh generation: Theory, practice and applications. *International Journal of Computational Geometry & Applications*, 10(3):227–266, 2000.

[50] Lara M. Vigneron, Jaques G. Verly, and Simon K. Warfield. *On Extended Finite Element Method (XFEM) for Modelling of Organ Deformations Associated with Surgical Cuts*, volume 3078 of *Lecture Notes in Computer Science*, pages 134–143. Springer, Berlin/Heidelberg, 2004.

[51] G. J. Wagner, N. Moës, K. W. Liu, and T. Belytschko. The extended finite element method for rigid particles in Stokes flow. *International Journal for Numerical Methods in Engineering*, 51(3):293–313, 2001.

[52] Andreas Wiegmann and Kenneth P. Bube. The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 35(1):177–200, Feb. 1998.

[53] Jinchao Xu. *Theory of Multilevel Methods*. PhD dissertation, Cornell University, May 1989.