High Performance FEM Simulation

Prof. Dr. Stefan Turek, Dipl. Math. Sven Buijssen, Dipl. Math. Matthias Grajewski, Dipl. Math. Hilmar Wobker

1. Motivation

Processor technology is still dramatically advancing and promises further enormous improvements in *processing data* for the next decade. On the other hand, much lower advances in *moving data* are expected such that the efficiency of many numerical software tools for Partial Differential Equations (PDEs) is restricted by the cost for memory access. In this article, we briefly discuss concepts for special discretization techniques (*adaptive error control* for Finite Element (FEM) methods [7]) and corresponding solvers (ScaRC [7] as generalization of MG/domain decomposition [DD]) for the numerical treatment of PDEs, and we illustrate the influence of processor technology on FEM simulation tools, particularly in computational structural mechanics (CSM) and computational fluid dynamics (CFD). While the numerical and computational results in this paper are mainly based on our own studies, we refer to [8] and related literature if we discuss the recent technological development:

- Not *data processing*, but *data moving* is costly.
- Employ *cache-oriented* techniques based on data locality and pipelining.
- Exploit *locally structured* data.

Therefore, the aim of this article is to illustrate how these 'paradigms' may influence the design of modern Numerics and of corresponding FEM software for PDEs.

2. Numerical concepts and related PDE software

The examples in [3] indicate that many of today's numerical simulation tools – based on the standard *sparse* MV techniques – are not able to achieve a significant percentage of the high performance on recent processors which is in the range of more than 1 GFLOP/s; in the case of *fully adaptive FEM codes* our measurements show that we better talk about performance rates of about 1 – 10 MFLOP/s and less for matrix-vector multiplications which are already the fastest components in numerical codes, while complete multigrid solvers often perform much slower. However, *adaptive mesh refinement* and *a posteriori error control* are believed to be absolutely necessary for realistic problem configurations in future. So, how to combine these adaptive, implicit, hierarchical numerical approaches with optimized implementation techniques for exploiting high performance computing?

2.1. Adaptive FEM approaches

Most of the recent numerical approaches for adaptive grid refinement (or coarsening) are working in a hierarchical way: One starts with a 'coarse' mesh which gets recursively refined. We call it in the following the *macro mesh*, and each triangle/quadrilateral/etc. is a *macro*. Depending on the *error indicator or error estimator*, certain global refinement steps are performed, or mesh refinement is applied only locally (see [7] for such grid structures which can lead to 'optimal' locally refined meshes with hanging nodes). Such meshes are highly adapted with very different local grid spacing. However, in most parts of the computational domain, very often the use of local tensorproduct grids is possible: fully adaptive local mesh refinement including hanging nodes is necessary only in very few regions, particularly near the boundaries. While the resulting grids usually contain a lot of highly structured features, at least in a local sense, the corresponding matrix structure of *sparse* type neglects all these nice structures: The typical *sparse* CSR technique, which is representative for most others, is *globally* defined and cannot exploit *local* tensorproduct meshes, resulting in poor MFLOP/s rates. Therefore, we exemplarily show an alternative approach which is based on patchwise adaptive techniques: The resulting mesh is locally refined, including hanging nodes and anisotropic mesh deformation, but it consists of many generalized (local) tensorproduct meshes such that *sparse banded* MV techniques can be locally applied. Hence, the aim of this 'macro-oriented' concept is to explicitly exploit such locally structured parts such that

- the local use of the highly efficient *sparse banded* techniques is possible
- local superconvergence results through locally orthogonal meshes can be exploited
- storage costs can be (partially) diminished through the use of matrix stencils only.

First step: Macro-oriented adaptivity

In the first step, we adapt the *macro mesh* (left picture, 25 macros), hereby performing completion techniques which lead to new conforming *macro meshes*: The resulting *macro mesh* (105 macros) is now semi-adapted towards the cylinder. This new macro mesh will be the basis for all subsequent local refinement steps which are applied in the corresponding single macros only. This means that each of these 105 macros will lead to (local) generalized tensorproduct meshes during the following (global) refinements.

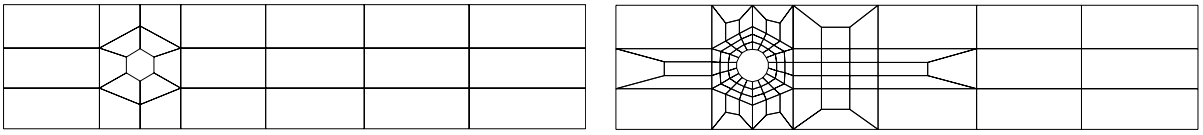

Figure 1.
Example of macro-oriented conforming local refinement of a given macro mesh.

Second step: Patchwise adaptivity

Having adapted the initial *macro mesh*, we proceed with local refinements inside of the single *macros*. Here, we apply the following two techniques which both lead to generalized tensorproduct meshes in each *macro* such that the use of optimized *sparse banded* techniques (see [3]) is allowed:

1. Use generalized tensorproduct meshes – rowwise numbering, but locally different mesh widths – with $M \times M$ vertices (or analogously in 3D), M may be different in each macro. As typical for fully adaptive approaches, we allow for technical and computational reasons only 1-irregular hanging (also called blind or improper) nodes on the macro edges.
2. Move mesh points in an arbitrary way as long as the local tensorproduct structure is preserved, for instance towards boundary layers or due to strong (interior) gradients.

Since the implementation is in the framework of generalized tensorproduct meshes, the corresponding data structures are quite easy to handle, and the high performance of the *sparse banded* techniques (see [3]) can be exploited such that there is no major reason for decreased performance of this special hanging node technique. Figure 2 shows the effect of local refinement in a layer of macro elements which are centered around the circle. Further global refinement will lead to tensorproduct meshes with $M \times M$ and $2M \times 2M$ gridpoints respectively.

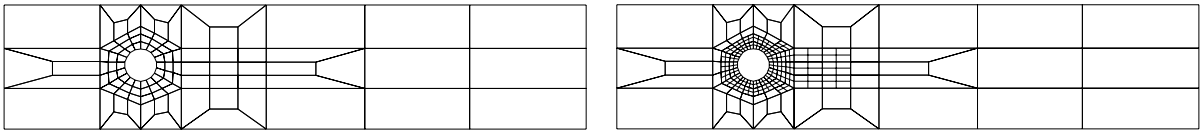

Figure 2.
Example of one step of macro-oriented local refinement in certain macros.

Next, in Figure 3, we apply another local refinement in those macros which are directly attached to the circle such that – in those macros – the additional refinement level is of size 2, resulting in local tensorproduct meshes with $M \times M$ up to $4M \times 4M$ gridpoints. Roughly speaking, the result is a mesh with many local tensorproduct grids, with respect to each *macro*, hereby allowing a certain degree of nonconformity due to blind nodes on the *macro edges* such that the local refinement can be different for each underlying macro element.

Finally, we apply local mesh deformation techniques which use additional local grid alignement, in this configuration towards the cylinder. We do not show the corresponding figures since these small mesh distances are very hard to demonstrate graphically. As the subsequent examples will show, the local mesh width h_{min} can be significantly decreased while at the same time the mesh aspect ratio, that means the grade of grid deformation, is dramatically increased.

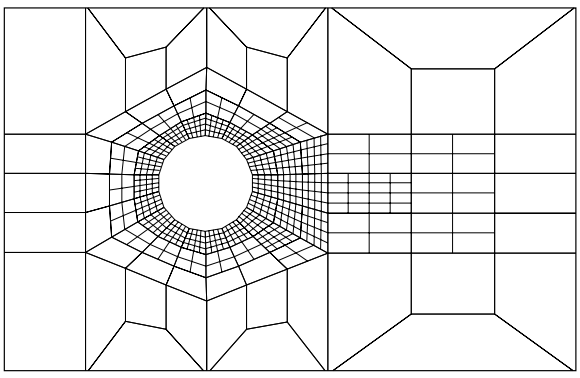

Figure 3.
Example of another macro-oriented local refinement step (zoomed region).

Third step: Fully local adaptivity

In the final step, we may perform the typical fully local adaptive refinement techniques, but only in very specific *macros*, and only then if the previous macro/patch-oriented approaches have not led to satisfying results. Since locally the tensorproduct structures are lost, the typical *sparse* techniques for such 'unstructured' grids have to be used, but only in single *macros*. Hence, the goal must be to work as often as possible with both purely macro/patch-oriented techniques, and to minimize the number of *sparse macros*, i. e., macros with (locally) unstructured grids.

Next, we consider the problem of how to incorporate such locally structured FEM discretization features in corresponding (parallel) *multigrid* and *domain decomposition* solvers which shall have 'optimal' complexity with respect to numerical as well as computational efficiency. This means that we do not only refer to the efficiency of a numerical method, but also to the efficiency of implementations for this method.

2.2. Generalized MG/DD solvers of SCARC type

In view of their typically excellent convergence rates, multigrid methods seem to be most suited for the solution of many PDEs. However, as the examples in [3] have shown, multigrid on general domains has often poor computational efficiency, at least if the implementation is based on the standard sparse techniques. As a result from our performance measurements in [3], the realistic MFLOP/s rates for complete multigrid codes are often in the range of 1 – 10 MFLOP/s only, even on very modern high performance workstations. Moreover, the linear relationship between problem size and CPU time may sometimes get hardly realizable, due to problem-size dependent performance rates of the sparse components. Additionally, the robust treatment of complex mesh structures with locally varying details is often hard to satisfy by typical 'Black Box' components, even for ILU smoothing, and particularly on parallel systems. Motivated by these facts, we developed a more general strategy for solving discretized PDEs, particularly in a parallel framework, which satisfies several conditions:

- The parallel efficiency shall be high due to a non-overlapping decomposition and a low communication overhead.
- The convergence rates ρ are supposed to be independent of the mesh size h, the complexity of the domain and the number of subdomains N, and they shall be in the range of typical multigrid rates (as $\rho \sim 0.1$).
- The method shall be easily implementable and use only existing standard methods.
- The approach shall guarantee the treatment of complicated geometries with local anisotropies (large aspect ratios), without impairment of the overall (parallel) convergence rates.

The underlying idea is to 'hide recursively all anisotropies in single subdomains' combined with an outer 'block Jacobi/Gauß-Seidel smoothing' within standard multigrid. This approach is based on the numerical experience [5] that these 'simple' block-oriented schemes perform well as soon as all occurring anisotropies are locally hidden, that means if the local problems on each block are solved (more or less) exactly. These ideas are combined with corresponding hierarchical data and matrix structures, which exploit the described tensorproduct-like meshes on each macro to achieve the high performance rates for the necessary Numerical Linear Algebra components in the local (multigrid) solvers. Consequently, all solution processes are recursively performed via sequences of more 'local' steps until the lowest level, for instance a single macro with the described generalized tensorproduct mesh, is reached. Consequently, the complete SCARC approach (see [1], [5], [6] for more details) can be characterized as:

- **Scalable** (with respect to 'quality and number of local solution steps' at each stage)
- **Recursive** ('independently' for each stage in the hierarchy of partitioning)
- **Clustering** (for building blocks via 'fixed or adaptive blocking strategies')

Again, the main philosophy is to recursively split the global problem into a sequence of local problems, preferably on macros which allow the robust and efficient solution of the auxiliary problems at very high MFLOP/s rates. The number of arithmetic operations tends to be larger than for the standard approach, but these local operations are cheap: Data moving, not data processing is costly!

The following Figure shows parallel convergence rates of SCARC for a sequence of adaptively refined meshes (indicated by 'MESH') including a global refinement strategy due to hanging nodes and some additional grid alignment. 'r' denotes the convergence rate of SCARC and '#IT' gives the required number of global iteration steps to gain 6 digits. 'AR' denotes the maximum aspect ratio, 'h_{min}' the minimum mesh width, and '#NEQ' shows the total number of unknowns of the problem. The applied SCARC scheme is additionally

embedded into an outer CG ('conjugate gradient') solver; we perform $l = 2$ (global) smoothing steps (V-cycle) in each ScaRC step and solve the local problems on each macro via local multigrid (F-cycle, 2 smoothing steps). The local multigrid uses a 'linewise Gauß-Seidel smoother' (MGTRI) to handle the strong local anisotropies in the mesh. Due to the anisotropic refinement in direction towards the circle, a finest mesh size $h_{min} \approx 10^{-12}$ is obtained. Nevertheless, the resulting parallel convergence rates are almost the same for all kinds of local refinement, even being independent of the amount of local anisotropy which can be of order AR $\approx 10^7$: They all are in the range of $\rho_{Scarc} \sim 0.1$.

Mesh Name	#NEQ	AR	h_{min}	ρ (#IT)
25-3	1,704	3	0.10-1	0.03 (5)
25-4	6,608	3	0.50-2	0.03 (4)
25-5	26,016	3	0.25-2	0.03 (4)
25-5-ad1	44,544	3	0.12-2	0.03 (5)
25-6-ad1	177,152	3	0.62-3	0.04 (5)
25-5-ad1-an	177,152	10^5	0.90-8	0.09 (6)
105-6	431,317	10	0.43-3	0.09 (6)
105-7	1,722,773	10	0.22-3	0.08 (6)
105-7-ad1	1,034,581	10	0.11-3	0.09 (6)
105-7-ad2	9,344,533	10	0.54-4	0.09 (6)
105-7-ad2-an	9,344,533	10^6	0.29-10	0.09 (6)
105-8-ad5-an	37,366,805	10^7	0.30-11	0.09 (6)

Tab. 1: ScaRC results

#NVT	MV-V/C	MGTRI-V/C
652	788/1301	558/706
2572	396/1255	434/672
10252	158/542	163/357

Tab. 2: DEC 21264 (833 MHz) 'ES40'

#NVT	MV-V/C	MGTRI-V/C
65^2	258/1275	299/706
257^2	187/638	173/382
1025^2	179/630	146/288

Tab. 3: AMD K7 (1333 MHz) 'XP+'

Figure 4.
ScaRC results for different configurations and for a sequence of locally refined meshes (Tab. 1); obtained MFLOP/s rates for matrix-vector multiplication (MV) and complete multigrid (MGTRI) based on Sparse Banded BLAS on the single subdomains (Tab. 2 & Tab. 3)

It is obvious that parallel load balancing is quite complex for this ScaRC approach and cannot be determined via standard *a priori* strategies, for instance by equilibrating the number of unknowns on each processor. Instead, we have to perform *a posteriori* load balancing techniques, analogously to the adaptive approaches, which might be based on the numerical and computational run-time behaviour of the last iteration to equilibrate better the total required CPU time on each processor. More information and computational studies for such implementations of ScaRC in the framework of Feast are part of the recent work of Chr. Becker [1] and S. Kilian [5].

[1] Chr. Becker. FEAST – The realization of Finite Element software for high-performance applications. PhD Thesis, University of Dortmund, 2004 (to be published)

[2] Chr. Becker, S. Kilian, S. Turek. Consequences of modern hardware design for numerical simulations and their realization in FEAST. Proceedings „Euro-Par'99 Parallel Processing" (P. Amesto, P. Berger, M. Dayde, I. Duff, V. Fraysse, L. Giraud, D. Ruiz eds.), Toulouse, France, August/September 1999, 643–650

[3] Chr. Becker, A. Runge, S. Turek. The FEAST INDICES – Realistic evaluation of modern software components and processor technologies. Computer and Mathematics with Applications, 41, 1431–1464, 2001

[4] S. Buijssen, S. Turek. Sources of parallel inefficiency for incompressible CFD simulation. 8th International Euro-Par Conference Paderborn, Germany, August 27–30, 2002 Proceedings (B. Monien, R. Feldmann, eds.), LNCS, Springer, 701–704, 2002

[5] S. Kilian. SCARC als verallgemeinerter Mehrgitter- und Gebietszerlegungsansatz für parallele Rechnerplattformen. PhD Thesis, Logos Verlag, Berlin, 2002, ISBN 3-8325-0092-8

[6] S. Kilian, S. Turek. An example for parallel SCARC and its application to the incompressible Navier-Stokes equations. Proc. ENUMATH-97, Heidelberg, October 1997, World Science Publ., 1998

[7] R. Rannacher. Error control in finite element computations. In H. Bulgak and C. Zenger, eds., Proc. Summer School Error Control and Adaptivity in Scientific Computing, Kluwer, Academic Publishers, 247–278, 1998

[8] U. Rüde. Technological trends and their impact on the future of supercomputers, High Performance Scientific and Engineering Computing (H.-J. Bungartz, F. Durst, Chr. Zenger, eds.), LNCSE 8, Springer-Verlag, 1999