PadFEM: A Portable Parallel FEM-Tool *

Ralf Diekmann, Uwe Dralle, Friedhelm Neugebauer, Thomas Römke
Paderborn Center for Parallel Computing, PC^2,
University of Paderborn, 33095 Paderborn, Germany †
email: {diek, dralle, freddy, tomtom}@uni-paderborn.de
Phone: +49 5251 60 6690, Fax: +49 5251 60 6697

Abstract

We present a modular toolbox for parallel finite element simulations on distributed memory systems. The library named *PadFEM* includes a graphical editor for specifying domains with boundary conditions, automatic mesh generation, automatic mesh partitioning and mapping onto the processors of a MIMD-system. The parallel FEM-simulation uses the preconditioned conjugate gradient method. The parallelization is based on PVM which guarantees portable parallel code. The current implementation is able to solve the two-dimensional Poisson equation in arbitrary domains with mixed Dirichlet- and Neumann-type boundary conditions using triangular meshes. The modular structure based on the idea of *programming frames* allows an easy extension to other problems.

1 Introduction

The solution or approximation of partial differential equations using finite element, finite volume, finite difference, or boundary element techniques is one of the most important tasks in the field of scientific computing for engineering, physics, chemistry biology and many other disciplines. 95% of all stability proofs in engine production use FEM. Simulations of heat conduction, fluid dynamics, diffusion, and weather and climate developments use finite difference or finite element methods.
Common to all of these methods are their demand for high performance floating point power and large memory requirements. To solve complex problems with sufficient accuracy (e.g. 3D fluid dynamics or crash simulation), the mesh can reach sizes of several millions of elements and the computation times grow up to weeks even on the most powerful vector supercomputers.
Massively parallel systems embody the potential to overcome these disadvantages, provided that highly efficient parallel solvers and techniques to effectively utilize distributed memory architectures are available. Currently, the most efficient parallel solvers are Multigrid methods (MG) or Preconditioned Conjugate Gradient methods (PCG). The parallelization is done by data distribution using Domain Decomposition (DD) techniques [1, 3, 5, 7] splitting the domain of interest into as many subdomains as there are numbers of processors.

*This work is supported by the DFG Sonderforschungsbereich 376 "Massive Parallelität", by the Heinz Nixdorf Institut Paderborn (Project "Netze"), by the EC Esprit Basic Research Action Nr. 7141 (ALCOM II) and the EC HC&M Project MAP

†WWW: http://www.uni-paderborn.de/pcpc/pcpc.html

A number of tools have been designed to support and facilitate the use of numerical methods [1, 4]. Some of them use the potential of parallel systems, some of them include newest and most efficient numerical methods. The support for automatic pre- and post-processing is differing. Often, a lot of hand-work has to be done.

Aim of this work is to design a flexible toolbox combining the most advanced methods from mathematics and computer science to support highly efficient parallel numerical simulation including comfortable pre- and post-processing.
The toolbox is designed in a modular way with small and clearly defined interfaces, mainly on file-basis. Tcl/Tk is used to develop a portable prototype user-interface with a two-dimensional domain editor which allows the specification of arbitrary domains including boundary conditions, modification of meshes, modification of mesh-partitions and visualization of results.
Mesh *generation* is often a critical task in numerical simulation where a lot of manual work has to be done. The toolbox includes an automatic mesh generator on quad-tree basis able to generate optimal two-dimensional triangular meshes in arbitrary domains with holes [2].
For parallel FEM simulations, the mesh has to be partitioned into subdomains which can be assigned to processors. The toolbox includes implementations of the Inertial partitioning method (see e.g. [14]) together with a modified version of the KL-algorithm [12]. The file-interface to the partitioning module allows to include other existing partitioning algorithms and libraries such as HS [8], PAR^2 [7], Chaco [9],[1] Metis [11] or Jostle [15]. The mapping of subdomains to processors is determined using a combination of GEOM MAP and Bokhari's local improvement algorithm [7].
The parallel numerical simulation is based on an abstract data-type describing the mesh and including all relevant information such as non-zero matrix entries, right hand sides and boundary values. Storing the matrix together with the mesh avoids the need for additional sparse storage schemes and allows an easy extension to adaptive methods. The current implementation is able to solve the Poisson equation with mixed Dirichlet- and Neumann-boundary conditions in two dimensions using triangular finite elements. Conjugate Gradient, Preconditioned Conjugate Gradient and Multigrid iteration algorithms are implemented to solve the resulting system of linear equations. Basis for the parallel implementation is PVM running on workstation clusters, shared memory systems (SparcCenter 1000, PowerChallenge) and MPPs (Parsytec GC/PP).
The overall design of the toolbox follows the idea of *programming frames* which are skeletons with problem dependent parameters to be provided by the users. Programming frames focus on re-usability and portability as well as on small and easy-to-learn interfaces. Thus, non-expert users will be provided with tools to program and exploit parallel machines efficiently. The modular structure of our toolkit allows the easy exchange of certain parts of the library (the frame parameters). The parallel abstract data type hides all details of the parallel systems from the user. The update of inner boundaries between subdomains is done automatically, the numerical code does not differ from sequential implementations using the same abstract data-type. All parts of the library can easily be exchanged, prototype developments can be done using a sequential version.

[1] © Sandia National Laboratories. Use granted by License Agreement No. 93-N00053-021.

2 PadFEM-Lib

2.1 The Frontend

One of our objectives was to provide a unique and easy-to-use user-interface. We also wanted to be able to run certain parts of the software on different machines, e.g. to use better graphical facilities or to show demonstrator applications. However, we did not want to develop the user-interface repeatedly for all window systems involved. Therefore, we decided to take the Tcl/Tk toolkit [13]. This toolkit offers a general purpose and extensible script language that was originally designed to develop X-window applications with a Motif like look and feel. Fortunately, MSDOS, Windows, and Macintosh ports are also available. This ensures that almost all reasonable window systems can run Tcl/Tk applications provided that additional (non Tcl/Tk) code is compiled for the corresponding architecture.
The graphical user interface (GUI) was implemented by exploiting the rapid development facilities of Tcl/Tk. Complex data structures, e.g. graphs, were implemented in C with corresponding access and modify functions. These functions are then integrated into the Tcl/Tk interpreter as new commands, and can be used from within the script language.
The GUI assembles the four stages in the FEM computation mentioned in the Introduction: editing and generation of polygonal bound domains, mesh generation, partitioning of the mesh, and finally the computation of the FEM algorithm on several parallel systems. The communication between the stages, although also possible with Tcl/Tk constructs, is file-oriented. That has two main advantages. First, files can be shared between different systems and can be computed independently. And secondly, one is able to restart computations at a certain stage, whilst this is not possible when using piping mechanisms or internal data structures. The GUI also provides additional information, help screens, and sophisticated editing/viewing facilities.

2.2 Mesh Generation

The generation of adequate meshes is a crucial step in numerical simulation. Very few commercial or academic tools include high quality automatic mesh generators. In many industrial applications such as car crash for example, the generation of meshes is done by very expensive *hand-work* taking often several weeks. It is estimated that the US industry would be able to save up to 75% of their budget spent for FEM simulations if automatic mesh generators were available [6].
Quality criterions for FE-meshes are first of all the shape of elements, as this determines the convergence behavior of iterative solution methods. The shape of elements is usually measured in terms of *aspect ratio* (i.e. the ratio between the radius of the smallest outer circle and the radius of the largest inner circle) or smallest or largest angle of the mesh. Of course, the size of the mesh (i.e. its number of elements) is also very important, especially if MG-methods are used. Comparisons of different mesh generators can be found in [10].
Within this toolbox, an automatic generator for triangular meshes in arbitrary, reasonable, two-dimensional domains with holes is provided. The implementation follows the quadtree-based algorithms given by Bern, Eppstein, and Gilbert [2]. The method is able to handle arbitrary domains with holes and guarantees optimal meshes in terms of size and aspect ratio. Bern, Eppstein, and Gilbert proved that no other method can generate smaller meshes with the same aspect ratio. They guarantee smallest angles of 18.4°.

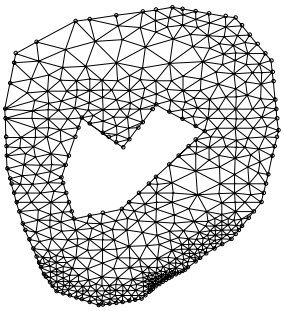

Figure 1: Automatically generated mesh.

Figure 1 shows the result of the mesh generator in an arbitrary artificial domain with hole. A Laplace-smoothing step is applied after the quadtree-based mesh generation.

2.3 Mesh Decomposition

Basis for a parallel numerical simulation is the decomposition of the mesh into as many subdomains as numbers of processors. This graph partitioning problem is itself *NP*-complete, thus only approximately solvable.
A number of powerful and efficient graph partitioning heuristics do exist, mostly based on recursive bisection [7, 8, 9, 11, 12, 14, 15]. They are usually divided into *global*- or *construction*-heuristics and *local*- or *improvement*-heuristics [8].
Within PadFEM we decided to implement *direct* partitioning heuristics which split the graph directly into a number of subdomains. The main advantage is the fact that such methods are no longer tailored to numbers of processors which are powers of two as is usually the case with recursive bisection.
We implemented a modified version of the Inertial partitioning method (see e.g. [14]) together with the Kernighan-Lin algorithm (KL, [12]) – applied to direct k-partitioning – as local smoother. The Inertial method performs a coordinate transformation of the domain such that its main axis corresponds to the x- or y-direction. Afterwards, a simple coordinate sorting delivers, in most cases, reasonable results in a short amount of time.
The KL-algorithm is a local improvement heuristic [12]. It chooses subsets of nodes to be exchanged between subdomains in a way that the cut size, i.e. the number of cut edges between subdomains, is decreased. While searching for subsets, temporary increases in cut size are allowed.

2.4 Parallel FEM

We consider the Poisson problem with mixed boundary conditions as shown in the left part of Fig. 2 on a two-dimensional arbitrary domain Ω:

$$-\Delta u(x,y) = f(x,y) \qquad \forall (x,y) \in \Omega \tag{1}$$

$$u(x,y) = g(x,y) \qquad \forall (x,y) \in \ , 1 \tag{2}$$

$$\frac{\partial u}{\partial n} = h(x,y) \qquad \forall (x,y) \in \ , 2 \tag{3}$$

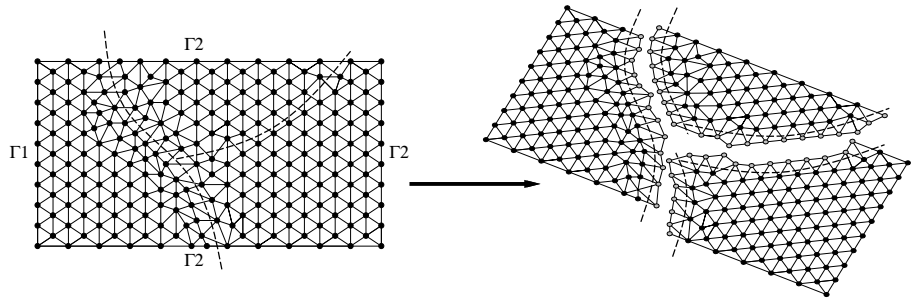

Figure 2: Overlapping Domain Decomposition with mixed boundary conditions.

This kind of elliptic partial differential equation is typical for a large class of physical problems. To solve the large systems of linear equations we implemented non-preconditioned as well as preconditioned Conjugate Gradient methods and a v-cycle Multigrid method [16].
The parallelization of the problem is actually done by domain decomposition techniques with one element wide overlapping at the artificial boundaries (cf. Fig. 2). This storage scheme allows communication latency hiding, as the boundary exchange and the computation of the interior matrix-vector product after the computation on the artificial boundaries can be done in parallel. On symmetric matrices the CG-method still requires three global scalar products, realized as local scalar products on the subdomains followed by a global sum. These are the only global synchronization points of the CG-algorithm.
Furthermore, a preconditioning CG-method is implemented. The construction of the implemented preconditioner is described in detail in [5]. For the calculation of the preconditioning matrix C^{-1} only the exchange of the information on the artificial boundaries is necessary. For the three global scalar products the same implementation as in the non-preconditioned cg-algorithm is used. Numerical results and various implementation techniques are also given in [3]. In general, the use of preconditioning leads to a better computation communication ratio compared to conventional CG-methods, but it requires a more sophisticate load balancing. The chosen decomposition for the CG-method achieves the required good load balancing, because the computation load of each processor can be accurately estimated by the number of assigned data points in each subdomain. The amount of communication is proportional to the cut size. Optimal problem decomposition and mapping leads to highly efficient programs even on large parallel systems like the GCel/1024 with 1024 processors. The time needed for global communication is negligible small. Because of optimal domain decomposition, mapping and parallel execution of computation and communication during matrix-vector products only little overhead of the parallelization appears. For the multigrid-method we use the same decomposition technique as for the CG algorithms. The same considerations as for the CG-algorithm, just for different grid-levels are necessary to get good performance for the multigrid algorithm.

3 Conclusions

The toolbox PadFEM is designed to combine the most advanced methods from computer science and mathematics to support efficient and scalable massively parallel numerical simulation including comfortable pre- and post-processing. Its modular structure follows the idea of programming frames which allows an easy exchange of certain parts of the code and a

simple extension to other methods and problems in the area of numerical simulation.
We plan to extend the toolbox to problems in the area of 3D instationary CFD using adaptive PCG methods. This includes the development of 3D mesh generators and efficient load balancing techniques.

Acknowledgments

We thank the 15 students of the project group ParFem for implementing most of the code. Thanks are also to Wolfgang Borchers and Burkhard Monien (Paderborn) for stimulating discussions.

References

[1] P. Bastian: *Parallel Adaptive Multigrid Methods.* Techn. Rep., IWR, Univ. of Heidelberg, 1993.

[2] M. Bern, D. Eppstein, J. Gilbert: *Provably Good Mesh Generation.* Proc. 31st Symp. on Foundations of Computer Science (FOCS), 1990, pp. 231-241

[3] S. Blazy, U. Dralle: *Comparison of numerical Methods for the Solution of elliptical Boundary-Value Problems on large Massively Parallel Systems.* Technical Report, Univ. of Paderborn, 1995.

[4] G.E. Blelloch, A. Feldmann, O. Ghattas, J.R. Gilbert, G.L. Miller, D.R. O'Hallaron, E.J. Schwabe, J.R. Shewchuk, S.-H. Teng: *Automated Parallel Solution of Unstructured PDE problems.* CACM, to appear

[5] J.H. Bramble, J.E. Pasciac, A.H. Schatz: *The construction of preconditioners for elliptic problems by substructering I.+II.*, Math. Comp. 47+49, 1986+87.

[6] *The CUBIT Mesh Generation Research Project,* Sandia National Lab, 1995,
WWW: `http://www.cs.sandia.gov/HPCCIT/cubit.html`

[7] R. Diekmann, D. Meyer, B. Monien: *Parallel Decomposition of Unstructured FEM-Meshes.* Proc. of IRREGULAR '95, Springer LNCS 980, 1995, pp. 199-215

[8] R. Diekmann, B. Monien, R. Preis: *Using Helpful Sets to Improve Graph Bisections.*
DIMACS Series in Discrete Math. and Theor. Computer Science, Vol. 21, AMS 1995, pp. 57-73

[9] B. Hendrickson, R. Leland: *The Chaco User's Guide.*
Technical Report SAND93-2339, Sandia National Laboratories, Nov. 1993

[10] K. Ho-Le: *Finite Element Mesh Generation Methods: A Review and Classification.* Computer Aided Design, Vol. 20, No. 1, 1988, S. 27-38

[11] G. Karypis, V. Kumar: *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs.* Techn. Rep. 95-035, Dept. of Comp. Science, Univ. of Minnesota, 1995.

[12] B.W. Kernighan, S. Lin: *An Effective Heuristic Procedure for Partitioning Graphs.*
The Bell Systems Technical Journal, Feb. 1970, pp. 291-308

[13] J.K. Ousterhout: *Tcl and the Tk Toolkit.* Addison Wesley, 1994.

[14] H.D. Simon: *Partitioning of unstructured problems for parallel processing.* Proc. Conf. on Par. Methods on Large Scale Structural Analysis and Physics Applications, Pergamon Press, 1991

[15] C. Walshaw, M. Berzins: *Dynamic Load-balancing for PDE solvers on adaptive unstructured meshes.* Concurrency: Practice and Experience 7(1), 1995, pp. 17-28

[16] O.C. Zienkiewicz: *The finite element method.* McGraw-Hill, 1989.