Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2007
M. Papadrakakis, E. Oñate and B. Schrefler (Eds)
©CIMNE, Barcelona, 2007

EFFICIENCY ISSUES OF PARTITIONED SOLUTION IN FLUID-STRUCTURE INTERACTION

Malte von Scheven*, Sunil R. Tiyyagura†, Ekkehard Ramm*
and Manfred Bischoff*

*Institute of Structural Mechanics, University of Stuttgart
Pfaffenwaldring 7, 70550 Stuttgart, Germany
e-mail: mvs@ibb.uni-stuttgart.de, web page: http://www.ibb.uni-stuttgart.de

†High Performance Computing Center Stuttgart, University of Stuttgart
Nobelstraße 19, 70569 Stuttgart, Germany
e-mail: sunil@hlrs.de, web page: http://www.hlrs.de

Key words: Fluid-Structure Interaction; partitioned coupling schemes; nonlinear Block Gauss-Seidel Iteration; Newton-Krylov Iteration; application examples

1 INTRODUCTION

Numerical simulation of large scale computational fluid dynamics (CFD) and fluid-structure interaction (FSI) problems is still today a very challenging task. The correct modeling of fluid flow, governed by the instationary incompressible Navier-Stokes equations, and the nonlinear structural behavior are challenges of their own. In addition coupling of both physical fields introduces further requirements on stability and efficiency of the involved algorithms.

For this class of problems computing time is still a limiting factor for size and complexity of the problem. Especially for FSI simulations the necessary iterative coupling schemes dramatically increase the required computing time. Here, the use of advanced coupling strategies, reducing either the time needed for one iteration or the number of iterations, can considerably speed up the simulation. We will introduce a class of coupling schemes enhanced by a coarse grid solution to reduce calculation time and accelerate convergence.

In addition, the single fields – especially the fluid – demand high efficiency of the solution algorithm. Here, assembly of the element matrices and in particular the iterative solver for the global system of linear equations are the most time consuming parts of the computational process. Very often these algorithms only use a small fraction of the available computer power in scientific codes [1]. Therefore it is highly advisable to take a closer look at the efficiency of algorithms and improve them to make the best out of the available computer power. We will present approaches to significantly increase efficiency in the assembly and solution part of a finite element code. Here, the special features of vector super computers will be exploited.

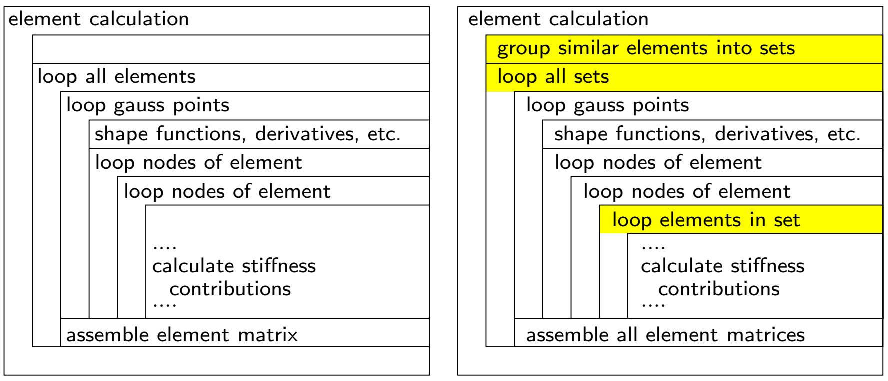

Figure 1: Old and new structure of the algorithm to evaluate element matrices.

2 COMPUTATIONAL EFFICIENCY

A major challenge facing computational scientists are facing today is the rapidly increasing gap between sustained and peak performance of high performance computing architectures [2]. The sustained computation to communication ratio and the computation to memory bandwidth ratio are much better for vector architectures when compared to clusters of commodity processors. This emphasizes the need to look towards vector computing as a future alternative for certain classes of applications. Vector processors like NEC SX-8 use a very different architectural approach than scalar processors. Vectorization exploits regularities in the computational structure to accelerate uniform operations on independent data sets.

2.1 VECTORIZATION CONCEPT FOR FINITE ELEMENTS

The major time consuming parts of FEM simulations are calculating the individual element contributions and solving the assembled global system of equations. The fine-grain data-parallelism inherent to finite element programs allows for vectorization in these parts of the code. In the following a straightforward concept, making use of this data-parallelism, to increase the performance of the integration of finite elements in arbitrary, unstructured meshes, is described.

The main idea is to group computationally similar elements into sets and then perform all calculations necessary to build the element matrices simultaneously for all elements in one set. *Computationally similar* in this context means, that all elements in one set require exactly the same operations to integrate the element matrix, i.e. they have e.g. the same topology and the same number of nodes and integration points.

The changes necessary to implement this concept are visualized in the structure charts in Figure 1. Instead of looping all elements and calculating the element matrix individually, now all sets of elements are processed. For every set the usual procedure to integrate the matrices is carried out, except on the lowest level, i.e. as the innermost loop, a new loop over all elements in the current set is introduced.

As some intermediate results now have to be stored for all elements in one set, the size of these sets is limited. The optimal size strongly depends on the hardware architecture.[3]

2.2 LINEAR ITERATIVE SOLVER

Most public domain solvers like AZTEC, PETSc, Trilinos, etc. do not perform as well on vector architecture as they do on superscalar architectures. The main reason for this is that their design considerations primarily target superscalar architectures. The design of most of the available solvers effects the following performance-critical features of vector systems.

The data structure used to store the sparse matrix plays an important role in the performance of such operators. Row based data structures used in the above mentioned solvers result in a low average vector length. Though, they are an easy and natural way of representing a sparse matrix object, they reduce performance of one of the critical operators on vector machines. A well known solution to this problem is to use pseudo diagonal data structure (JAD) to store the sparse matrix [4].

A major hurdle to performance in the sparse MVP kernels is not memory bandwidth but the latencies involved due to indirect memory addressing. Block based computations exploit the fact that many FEM problems have more than one physical variable per grid point to be solved. Thus, small blocks can be formed by grouping the equations at each grid point. Operating on such dense blocks considerably reduces the amount of indirect addressing required for sparse MVP [5]. This improves the performance of the key kernel dramatically on vector machines and also remarkably on superscalar architectures [6].

Taking into account these design aspects for a linear iterative solver, our in-house solver reaches a single CPU performance for sparse MVP of 7.2 GFlop/s (about 45% vector peak) on the NEC SX-8.

3 PARTITIONED SOLUTION OF FSI

The partitioned solution of FSI problems is particularly advantageous in terms of flexibility and code reuse. As staggered algorithms proved to be unstable in a wide range of applications, iterative schemes need to be employed. Their convergence rate is decisive for the overall efficiency of the algorithm.

We will introduce two different iteration procedures: a nonlinear Block Gauss-Seidel iteration [7] and a Newton-Krylov based approach [8, 9]. For both coupling strategies it is possible to introduce in various places a coarse grid solution to accelerate the iteration process. For the nonlinear Block Gauss-Seidel iteration the use of a coarse grid predictor reduces the number of iterations and the time to convergence significantly A coarse grid solution of the coupled problem can be used for the Newton-Krylov iteration to accelerate the approximation of the tangent operator.

The robustness and efficiency of these coupling schemes is shown on a variety of large scale numerical examples.

REFERENCES

[1] Behr, M., Pressel, D.M., Sturek, W.B. *Comments on CFD code performance on scalable architectures.* Computer Methods in Applied Mechanics and Engineering 190 (2000) pp. 263–277.

[2] Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S. *Scientific computations on modern parallel vector systems.* Proceedings of the ACM/IEEE Supercomputing Conference (SC 2004) Pittsburgh, USA (2004).

[3] Neumann, M., Tiyyagura, S.R., Wall, W.A., Ramm, E. *Robustness and efficiency aspects for computational fluid structure interaction.* In E. Krause, et al. (eds.): Computational Science and High Performance Computing II. NNFM 91, Springer (2006).

[4] Saad, Y. *Iterative methods for sparse linear systems, Second Edition.* SIAM, Philadelphia, PA (2003).

[5] Tiyyagura, S.R., Küster, U., Borowski, S. *Performance improvement of sparse matrix vector product on vector machines.* In Alexandrov, V et al. (eds.): Proc. of the Sixth Int. Conference on Computational Science (ICCS 2006). LNCS 3991, Springer (2006).

[6] Tuminaro, R.S., Shadid, J.N., Hutchinson, S.A. *Parallel sparse matrix vector multiply software for matrices with data locality.* Concurrency: Practice and Experience 3 (1998), pp. 229–247.

[7] Wall, W.A., Mok, D.P., Ramm, E. *Interactive substructuring schemes for fluid-structure-interaction.* In: 'Analysis and Simulation of Multifield Problems' (W.L. Wendland, M. Efendiev, eds.). Lecture Notes in Applied and Computational Mechanics, Springer (2003) pp. 349–360.

[8] Gerbeau, J.-F., Vidrascu, M., Frey, P. *Fluid-structure interaction in blood flows on geometries based on medical imaging* Computers and Structures 83 (2005) pp. 155–165.

[9] Michler, C., van Brummelen, E.H., de Borst, R. *An interface Newton-Krylov solver for fluid-structure interaction* International Journal for Numerical Methods in Fluids 47 (2005) pp. 1189–1195.