Numerical Simulation in Automotive Design

G. Lonsdale *C&C Research Laboratories, NEC Europe Ltd., St. Augustin, Germany*

Abstract

Focussing on those areas of simulation which dominate the large-scale computational requirements within the automotive industry, the initial aim of the paper is to give an insight into typical applications and algorithms being used. Beyond this, the main emphasis will be on the more recent developments related to the take-up of HPC technology in the form of the distributed-memory, message-passing programming paradigm. Examples will be taken from commercial applications codes involved in the ESPRIT porting action EUROPORT.

Keywords: Automotive, Simulation, Message-passing, Structural Mechanics, CFD

1 Introduction

Although numerical simulation has been included within the R&D activities of the automotive industry for some time, recent years have seen a real increase in the use of numerical simulation *in the design phase***.** This shift from research topic to design tool is an ongoing process: in some areas the latest High Performance Computing (HPC) developments are starting to allow design use; in other areas where numerical simulation is a well established design tool, the frontiers of what is possible with simulation are being pushed back. This trend is not restricted to the major automotive manufacturers alone: their suppliers, often small- or medium-sized enterprises, are using numerical simulation in their design processes. The range of applications areas is ever broadening, some examples being: simulation of the design of the tools to be used in manufacturing processes such as metal forming or extrusion blow-forming for plastic components; stress analyses of the car-body or component structures; crashworthiness and occupant safety simulation; fluid simulation of the external aerodynamics or in-cylinder combustion; electromagnetic compatibility analysis.

The main emphasis of the lecture is on the recent developments within automotive simulation codes related to the take up of HPC technology in the form of the distributed-memory, message-passing programming paradigm. Included within the presentation of parallelization strategies and attained performances will be overviews of typical applications and algorithms. Given the large range of applications areas, it is not possible to address all simulation codes or algorithms being used in the framework of a single lecture. Thus, this lecture focusses its attention in two ways:

- On two of those areas of numerical simulation which could be classed as "classical" applications within the automotive industry - Computational Fluid Dynamics (CFD); crash simulation; Linear Statics and Normal Modes analyses.
- On commercial applications codes which have recently moved to parallel, distributed computing and which were involved in the recent ESPRIT porting action EUROPORT [9, 13].

The above classical applications represent the major large-scale computational use within the automotive manufacturers: a recent presentation by Dr. Himeno from Nissan Motor Company Ltd. estimated the CPU use of those applications at approximately 90% of the main computer/supercomputer resources, with each area requiring/using more or less an equal share of that total. Since the accompanying paper “Parallel & Distributed Crashworthiness Simulation”

will deal in some detail with HPC aspects of the crashworthiness code PAM-CRASH, this paper will be restricted to the remaining two topics. Disregarding the acknowledged success of the EUROPORT action in promoting the benefits of parallel and distributed computing, the advantage of concentrating on codes ported within the Europort action is the availability of benchmarking results for real industrial applications[1].

2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) in the automobile industry covers a wide area of applications, including: external aerodynamics, in-cylinder flow and combustion, coolant flow, engine compartment and passenger compartment analyses. The general area exemplifies the varying levels at which HPC technology is helping to improve and enhance the design process: CFD for external aerodynamics has been a design tool since (approximately) the beginning of this decade, the latest HPC architectures are allowing both the extension of the features investigated by simulation - an example being the simulation of aerodynamic noise emitted from a door mirror - or the possibility of replacing wind tunnel testing by CFD (see [6]). In other areas such as in-cylinder computations or internal flow simulations (including detailed heat and mass transfer analyses) the employment of HPC technology is promoting the move from the research and development stage into the production and design stage.

The two code examples considered here are again taken from the EUROPORT project and, although they employ differing solution and discretization schemes, make use of the same basic parallelization aproach (as indeed did many of the EUROPORT codes) of *mesh partitioning*. With this approach, the computational domain (mesh) is partitioned and computational processes (usually process**ors**) are assigned the task of performing all computations for a particular partition of the mesh. The aim is to maintain as far as possible the execution behaviour of the sequential algorithm. To this end, communication between the partitions is introduced. In both cases below, mesh partitions are constructed, in advance of the solution procedure, so that inter-partition communication is minimised. For explicit time-marching algorithms it is quite possible to maintain the exact sequential execution pattern. For algorithms involving implicit components (as in the case of the codes below), algorithmic changes may be necessary if high parallel efficiency is to be achieved, which may mean that the numerical behaviour is altered. *A major point to note is that, typically, the complete solution phase of the code is parallelized - with the possible exception of initialization and I/O phases, no sequential sections remain.*

The N3S code from Simulog includes in fact two separate codes: the incompressible solver[2], N3S-EF, and the compressible solver, N3S-MUSCL. Both schemes employ mesh-partitioning of the 3-dimensional, unstructured meshes but the parallelization requirements of the two codes are somewhat different (for details see [3,4,12]).

The N3S-EF code uses a purely finite-element spatial discretization and an operator-splitting time-marching approach. The diffusion and propogation steps of the algorithm result in the necessity to solve sparse linear systems. In the parallel code this is performed using a diagonally preconditioned conjugate gradient algorithm. The distribution of the matrix components within this algorithm corresponds directly to the mesh-partitioning. The first order advection terms are calculated using a characteristics algorithm which introduces dynamically defined "data accesses" corresponding to dynamically changing communication constructs in the

[1] It should be pointed out here that for the EUROPORT benchmarking, only portable codes were used which had not been optimized for particular platforms, in contrast to the production or sequential supercomputer versions with which they were compared.

[2] In fact, a restricted class of *compressible* flow problems can also be solved using this code.

message-passing version: a trial-and-error procedure for following characteristics was developed. Despite the latter feature, which has the *potential* to introduce very high communication overheads, the success criterion, set by the end-user IFP, of 4 × the performance of a single-node Fujitsu VPP500 on a 16-node IBM SP2 was achieved for an incompressible, turbulent flow calculation in an engine cylinder head.

The compressible code, N3S-MUSCL, employs a mixed finite-volume/finite-element formulation. The major feature is that the linear systems arising from the nonlinear implicit scheme are solved by relaxation methods: either Jacobi or Gauss-Seidel. The use of such relaxation methods provides an illustration of the effect of the parallelization on the choice of numerical algorithm, alluded to above. While the Jacobi relaxation provides the best performances on a vector supercomputer, the example shown in Figure 1 demonstrates that the improved numerical convergence of the block Gauss-Seidel scheme pays off in the message-passing parallel implementation. The other aspect to note is that the *block* implementation necessary to maintain parallel efficiency is a modification of the serial Gauss-Seidel relaxation - whose convergence behaviour would deteriorate for cases with insufficient granularity, i.e. for "small" problems on "large" processor numbers.

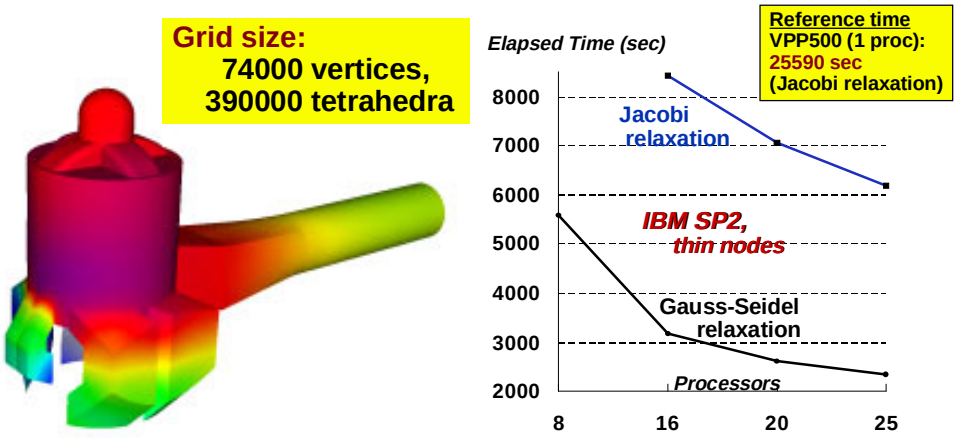

Figure 1: Flow inside the IFP 2-stroke engine (surface temperature displayed) and corresponding EUROPORT Benchmark results *Courtesy of IFP and the EUROPORT Project HPCN3S*

The STAR-HPC code is the parallel, message-passing version of the unstructured, finite-volume code STAR-CD, produced within the EUROPORT project. The implicit components of the solution method used, for both incompressible and compressible flow simulations, employs a pre-conditioned conjugate gradient algorithm. The parallel implementation developments, building on the mesh-partition approach, focussed on the design of efficient pre-conditioners and communication constructs to allow the distributed execution of the matrix-vector and vector-vector operations. As with all code porting carried out within the EUROPORT project, **portable** message-passing interfaces were used. One aspect of this portability was that the message-passing code could also be made available on *shared-memory* (or symmetric multiprocessor, SMP, machines), where the *message-passing interface exploited the shared-memory* of the architecture, not the application code. This allowed comparisons to be made between the message-passing code version and the shared-memory (complier directive cotrolled) version. Figure 2 demonstrates the performance gain of the message-passing version over the SMP version. This type of performance gain was also exhibited by other codes involved in the EUROPORT project, including PAM-CRASH as will be discussed in the aforementioned accompanying paper. However, a *caveat* to such comparisons should be added: the benchmarking was carried out on stand-alone systems and busy, multi-user exploitation may not always deliver similar gains (depnding on the operating system).

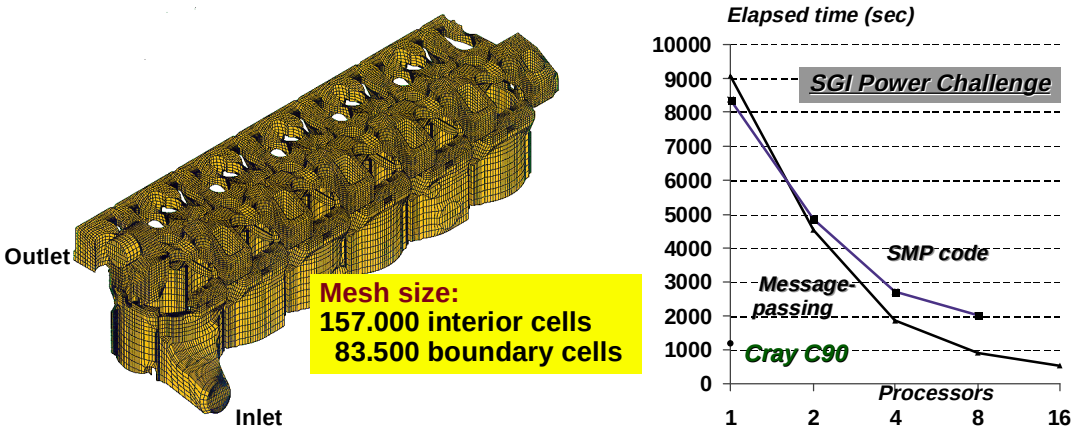

Figure 2: Mercedes Engine Coolant flow and correponding EUROPORT Benchmark results
Courtesy of Mercedes-Benz and the EUROPORT Project STAR-HPC

3 Linear Statics & Normal Modes Analyses

Although general purpose Finite Element (F.E.) codes may be employed for the numerical simulation of a range of mechanical and thermal processes of relevance to automotive design, the most heavily used options are for linear statics and nodal mode analyses (indeed, [8] states that such applications are estimated at between 70% and 80% of all industrial applications of the MSC/NASTRAN code). In this section, the HPC features of two such general purpose F.E. codes will be discussed: MSC/NASTRAN from MacNeal Schwendler and PERMAS from INTES.

In both the above analyses, the dominant feature resulting from the typical industrial automotive applications is the handling of very large, sparse matrices: for linear statics analysis, the solution of the linear system of equations; for normal modes analysis the solution of a related eigenvalue problem. The linear statics analysis allows the end-user to study the linear-elastic effects of static loading of both components and full vehicles, giving predictions of stress distributions and deformations. An example of a full vehicle model is shown in Figure 3. In addition to the analysis of vibrational damping and torsional behaviour, the normal modes analysis plays a major role in the determination of resonant frequencies which affect the passenger comfort.

For the linear statics analysis, either sparse direct solvers, involving a (triangular) matrix decomposition followed by forward-backward solution, or iterative solvers such as the pre-conditioned conjugate method are used. A typically used solution method for the eigenvalue problem is the Lanczos algorithm.The major computational tasks of the latter algorithm are matrix decomposition, forward-backward substitution and matrix multiplications. While such sparse algorithms belong to the classical areas of interest for numerical analysts, efficient parallel implementations for distributed-memory machines are not yet available in a general, standard way - indeed the provision of such libraries is a current research topic. For codes such as MSC/NASTRAN or PERMAS, the situation is significantly complicated by the fact that these solution algorithms are only a part of **very** large codes (developed over periods of over 25 years): of the order of 1.5 million lines of code.

The parallelization approaches adopted for the codes MSC/NASTRAN and PERMAS illustrate two quite different options for dealing with the task of parallelization of such large F.E. codes (for details see [8] and [1,11], respectively). Indeed, the approaches represent the typical

alternatives to the domain or mesh partitioning schemes employed by many structural or CFD codes (as exemplified by the applications in Section 2).

The approach adopted for MSC/NASTRAN is to parallelise the code modules corresponding to the most computationally intensively numerical parts. This provides a significant increase in performance with good efficiency on moderately large parallel systems: with those numerical solution modules typically requiring between 85% and 95% of total execution times, maximum theoretical speed-ups lie between 6 and 20. Current user requirements for reduced cost simulations or improved job throughput are met by this approach while the major advantage is that extensive modification to the code structures is avoided. Details of the parallelization can be found in [8], the salient features are as follows: The parallel iterative solver is based on a conjugate gradient type algorithm with diagonal scaling, Cholesky of Jacobi proconditioning and employs a distribution of matrix rows across processors. Parallel implementations of dot-products, vector updates and matrix-vector mulitplications were developed. The parallel direct solver uses a multi-frontal method and a two stage data distribution for the parallelization: distribution of fronts based on the elimination tree of the matrix combined with blocks of rows of the matrix for each front. The benchmark results of Figure 3 show the performance achieved with the direct solver for the BMW model. In addition to illustrating that the single-node supercomputer performance can be surpassed with a distributed-memory machine, it also highlights one of the open questions: I/O on parallel, distributed machines. The main gain in performance between the “thin-node” and “wide-node” IBM SP2 machines employed was the acceleration of the I/O enabled by the increased local memory of the wide-node being used for caching of local I/O.

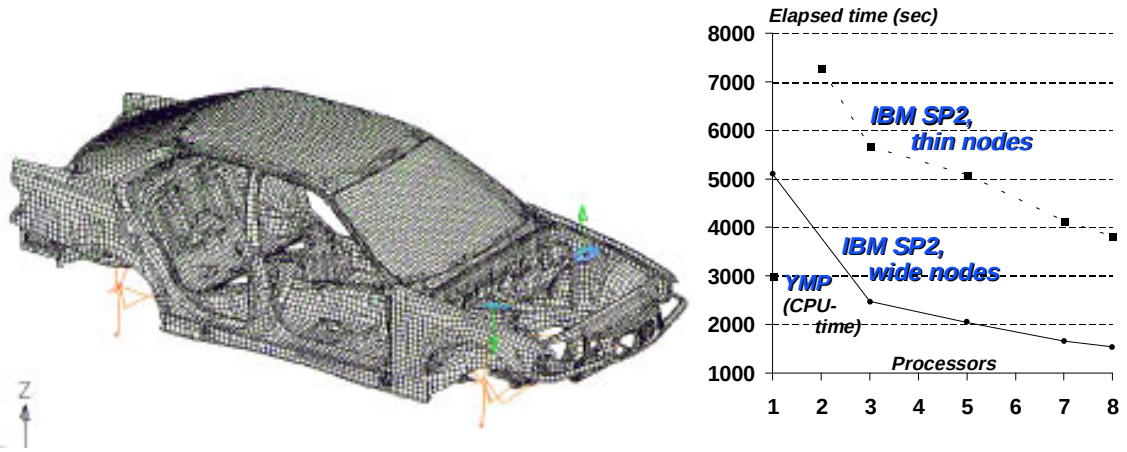

Figure 3: Static Analysis of a full car body, BMW Series-3 (239664 degrees of freedom) and EUROPORT Benchmark results *Courtesy of BMW & EUROPORT Project PMN*

Building on the basic data structures within the code, the parallelization of the PERMAS code uses the parallel task manager approach (for details see [11]). Within the PERMAS code, the large sparse matrices involved in the various numerical algorithms are stored and handled using three hierarchical levels, where only non-zero sub-matrices are stored or processed: the highest level (*hyper-matrix*) holds the structure of the division of the full matrix, related to the F.E. model, into sub-matrices corresponding to square blocks (typical sizes being 30×30 to 128×128); the next level is another matrix of pointers into (sub-) sub-matrices; the final level holds the real data of the F.E. matrix for the given sub-matrix. For the message-passing parallelization of PERMAS a new software layer was introduced: the Parallel Task Manager. Complemented by a distributed version of the data management system, this new layer enables the task parallel

(sometimes referred to “task farming”) approach in which the computational tasks corresponding to the final level sub-matrices are assigned to processors as they 'become available'. In fact, the implementation of the assignment is such that a task dependence graph is constructed, from which clusters of tasks (the clustering aimed at minimisation of sub-matrix communication) are assigned to the local queues of tasks to be performed by each slave processor.

With this task parallel approach, the basis for a complete code parallelization has been created without the need for a major re-design of its basic computational components. As part of the EUROPORT benchmarking, scalability tests with a 3-D model problem were carried out for the matrix decomposition[3]. With speed-ups of above 5 on 8 and 7 on 32 IBM SP2 processors (for approx. 150000 unknowns), a performance has been achieved which, when far away from *perfect parallel scaling,* demonstrates the advantage of its industrial exploitation.

4 Concluding Remarks

The recent demonstrations for industrially relevant applications, notable within the ESPRIT EUROPORT projects, have shown that HPC is a technology which can be exploited to enhance the design process. Automotive design, with a variety of numerical simulations needs (only touched on in this paper), is in a position to exploit the current code versions, but will also require the solution of outstanding problems if it is to continue to expand the areas within which simulation plays a decisive role. Future activity on parallel and distributed systems will need to address areas such as: efficient I/O on parallel systems; the development of highly efficient parallel direct methods; providing transparent, multi-user environments on both dedicated and networked parallel systems; the extension to multi-disciplinary simulation.

Acknowledgements

I am greatly indebted to the organisations (and EUROPORT projects) which provided both information and graphical material for this paper and the accompanying lecture. In particular, I would like to thank the following people: Prof. D. Gosman & Dr. R. Issa (Computational Dynamics), Dr. K. Stüben (GMD), Dr. R. Helfrich (Intes), Dr. H. Sippel (MacNeal-Schwendler), Dr. R. Himeno (Nissan Motor Co.), Dr. M. Loriot (Simulog).

References

1. Ast, M. *et.al.*, ‘A general approach for an automatic parallelization applied to the finite element code PERMAS’, pp. 844-849 in [5]
2. Bauer, W. *et.al.*, ‘Parallel Computing for CFD applications in the automotive industry - First experiences’, pp. 106-115 in [7]
3. Degrez, G. *et.al.*, ‘Parallel industrial CFD calculations with N3S’, pp. 820-825 in [5]
4. Giraud, L. *et.al.*,‘Parallel industrial incompressible CFD calculations with HPCN3S’, pp. 122-127 in [7]
5. Hertzberger, B. & Serazzi, G. (Eds), Proceedings of the HPCN '95 Conference, *Lecture Notes in Computer Science* ***919***, Springer-Verlag (1995).

6 Himeno, R.., ‘Car Aerodynamic Simulation on NEC SX-4 at Nissan Research Center’, *SX World*, **18**, NEC, Tokyo (1996)

[3] The industrial EUROPORT benchmark models used were from the shipbuilding industry.

7. Liddell, H., Colbrook, A., Hertzberger, B. & Sloot, P. (Eds.), Proceedings of the HPCN '96 Conference, *Lecture Notes in Computer Science* ***1067***, Springer-Verlag (1996).
8. Mayer. S. *et.al.,* 'Parallel MSC/NASTRAN on distributed memory computers', pp. 850-855 in [5]
9. Mierendorff, H. *et.al.,* 'Europort-1: Porting industrial codes to parallel architectures', pp. 806-812 in [5]
10. Robinson, D. *et.al. ,* 'Parallel STAR promises bright future', pp. 806-819 in [5]
11. Schulz, U. *et.al.,* 'Experiences and achievements with the parallelization of a large Finite Element system', pp.82-89 in [7]
12. Stoessel, A. *et.al.,*'Toward real CFD simulations on parallel computers in the aeronautic and automotive industry', pp. 99-105 in [7]
13. Stüben, K. *et.al.,* 'Parallel industrial Fluid Dynamics and Structural Mechanics codes", pp.90-98 in [7]