Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

On a coarse-grained parallel code to simulate reactive flows on an IBM RS/6000 workstation-cluster

F.-K. Hebeker
IBM Scientific Center, P.O. Box 103068, D-69020 Heidelberg, Germany

INTRODUCTION

The present note is a continuation based on previous work [3.]) in a joint research project with the Daimler-Benz AG, Stuttgart. Our aim is to get insight into physical reasons of knock damage in internal combustion engines. We invoke numerical simulation methods and take advantage of modern computer architecture. This work combines engineering-mathematical modelling with algorithmic and code development. Parallel processing is used on an IBM RS/6000 high-end workstation cluster.

Prohibiting knock damage in the spark ignited Otto engine forms severe restrictions concerning the efficiency of the engine. Whereas laboratory experiments pertaining to material thermal or mechanical loading are expensive or even impossible, numerical techniques were provided here as a helpful tool but have been, however, employed severely in the field of erosive damage analysis in fluids. A mathematical model essentially has to take into account complex interaction of diffusive effects with shock waves and reaction fronts (propagating and interacting mutually

with each other and with walls) and, on the other hand, the presence of nonequilibrium combustion phenomena. We invoke the compressible Navier-Stokes equations and global exothermic reaction chemistry (according to a two-substances model of burnt and unburnt mixture). Our numerical approach combines (via operator splitting) a recent shock-capturing finite-volume scheme for the compressible flow with semi-implicit treatment of the chemical source terms.

By careful and sensible algorithmic development a code was written optimally adapted to parallel computers. This algorithm (originally conceived to run on an IBM ES/3090 vector multiprocessor) has been implemented on an IBM RS/6000 high-performance workstation cluster. Coarse-grained parallelism has been achieved by employing the domain splitting techniques. For message passing we used the PVM programming environment.

In this note we present some performance measurements and illustrate the potential of the code by advanced physical details of the knock damage problem, namely by comparison between calculation results and experiment.

THE MATHEMATICAL MODEL

The underlying mathematical model takes advantage of the nonstationary Thin-Layer Navier Stokes equations extended by global exothermic reaction chemistry, which read in divergence form

$$\frac{\partial}{\partial t} U + \operatorname{div} F(U) = D(U) + S(U)$$

where $U = (\rho, \rho u, \rho v, e, Y)^T$ denotes the vector of unknown flow quantities, F the wellknown flux dyadic function,

$$D = \begin{pmatrix} 0 \\ \mu u_{yy} \\ \mu v_{yy} \\ \lambda \Theta_{yy} + \mu(u_y^2 + v_y^2) \\ 0 \end{pmatrix}, \quad S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ q\sigma \\ -\sigma \end{pmatrix}$$

and

$$p = (\kappa - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right), \quad \Theta = \frac{p}{\rho R}$$
$$\sigma = Y \exp\left(\beta\left(1 - \frac{1}{\Theta}\right)\right)$$

Here $\rho, u, v, e, Y, p, \Theta$ denote density, velocity in $x-$, resp. $y-$direction, total energy, and reaction progress parameter (according to a

two-substances model, here: volume concentration of fuel), thermodynamic pressure, temperature in the flow field, and $\kappa, \lambda, \mu, q, \beta$ denote the isentropic exponent, thermal conductivity, dynamic viscosity, heat release, and global activation energy of the explosive gas (all quantities assumed as constants).

These differential equations are investigated in an L-shaped plane domain where appropriate boundary conditions based on engine conditions after high compression at top drall center are posed. For some details of the mathematical model and the subsequent numerical algorithm we refer to Hebeker, Maly, Schoeffel [3].

THE NUMERICAL ALGORITHM

Our algorithm is substantially based on the splitting idea, namely combining domain splitting with dimensional and operator splitting within a second-order numerical scheme (both in time and space).

First of all, the L-shaped domain is split into a total number of d rectangular subdomains G_i. Each two neighboring subdomains share four rows of cells in order to allow proper modelling of appropriate boundary conditions on the intermediate boundaries between the subdomains (overlapping). Thus, one time step (of size Δt) is reduced to solve several related subtasks of the same kind on all of these rectangles:

$$L(\Delta t) = L_1(\frac{\Delta t}{2}) \ldots L_{d-1}(\frac{\Delta t}{2}) L_d(\Delta t) L_{d-1}(\frac{\Delta t}{2}) \ldots L_1(\frac{\Delta t}{2})$$

where L_i denotes the solution operator on subdomain G_i. All these subtasks are independent, so that the following procedure is done in parallel for all subdomains.

For any subdomain we use dimensional splitting to advance the solution. Thus the problem is further reduced to one-dimensional counterparts, where each of them may be solved numerically with respect to $x-$, resp. $y-$direction on short time intervals.

Finally we use operator splitting to separate the physico-chemical effects for the pure flow phenomena. The one-dimensional subproblems are split into a convection-diffusion part and a chemo-kinetic part. The convection-diffusion problem is solved by adopting novel shock-capturing explicit schemes for the convective part (namely vanLeer's MUSCL scheme, employing Roe's approximate Riemann solver, see [4]) with explicit treatment of the diffusive terms. This "Rapid-Solver Strategy" (MacCormack) is applicable for high Reynolds number flow.

In detail, a predictor-corrector scheme has been used to update the flow quantities over one time step within a second-order scheme.

The chemokinetic part is a nonlinear 2×2 –system of ordinary differential equations for the temperature and the unburnt fuel concentration, whereas the other flow quantities are assumed as "frozen". Since the system is stiff in general, we invoked a semi-implicit trapezoidal rule to update the unknowns, thus preserving second-order accuracy. Latter lead to unlimited stability. Consequently, the time step restrictions are posed by the convection-diffusion problem.

In the following we will show that the present algorithm is appropriate to run on the specific architecture of a high-performance workstation cluster. This is essentially due to the fact that "coarse-grained" parallelism (that means: much arithmetic, low communication) has been adopted by introducing the domain splitting procedure as early as possible. Thus, all other parts of the algorithm run without further data exchange between the processors.

NETWORK COMPUTING

In recent years network computing has been finding increasing acceptance in a wide range of computer applications as a low-cost/high-performance alternative to traditional computing centers. We shortly introduce our system platform (see Altevogt-Linke [1] for some more details).

At the IBM Scientific Center Heidelberg an IBM RS/6000 high-end workstation cluster is available to run multitasking jobs. The core of the cluster currently (March 1993) consists of one Model 560 workstation (100 MFlops peak performance) and seven Model 550 workstations (82 MFlops), equipped with 128 MB central memory each. The workstations are connected by different networks. Among them the Serial Optical Channel Converter (SOCC, with 220 Mb/s transfer rate) is the most efficient one via an NSC DX Router to provide for point-to-point connection between the processors. All machines run AIX 3.2 as the operating system.

Because of relatively high latency times ($\sim 1\,ms$) for a high-performance workstation cluster design and implemention of a coarse-grained parallel algorithm, including the initiation, communication and synchronization of the various processes on the physical processors is required.

For message passing on a workstation cluster we use a parallel programming extension of FORTRAN, e.g. PVM (Parallel Virtual Machine) [1]. These programming environments provide as basic

functionality a library of routines to create processes on the workstations, to communicate between theses processes (data exchange), to synchronize these processes, etc.

For programming our parallel application we use the client/server programming model. Here two levels of processes are distinguished. The job starts with the client process which handles administative and preprocessing tasks and opens the so-called server processes. These server processes run in parallel and typically serve to compute the time consuming information. Before expiring they return their results to the client process which finishes its job with postprocessing. Logically, these two kinds of processes are realized by employing two computer codes: the client process is governed by the client program, whereas all server processes are running the only server program but with different numerical parameters.

PARALLELIZATION AND TIME MEASUREMENTS

In detail the client and server processes handle the following tasks and communicate via the PVM FORTRAN statements of message passing. Each server process governs exactly one of the subregions. Preprocessing by the client process: it computes some relations of neighborhood between the subdomains, some physical and numerical constants, all initial data for the array STATE (containing most of the physical information), and it sends required information to the server processes. The time loop is essentially processed by the server processes: they receive information from the client process, exchange boundary data between server processes governing neighboring subdomains, carry out the core of the numerical algorithm, they update the subarray for those subdomains they treat and finally return the data to the client process. Postprocessing is done by the client process, receiving the data (collecting the subarrays to reestablish the array STATE , e.g.), doing I/O operations, etc.

Fig. 1 shows the results of some time measurements run on a dedicated workstation cluster, connected by SOCC. The L-shaped domain is split into six overlapping subdomains, each being treated by exactly one parallel process. Each subdomain is further decomposed into a total of $nx \times ny$ cells. The figures show the computing time required for one time step of the algorithm, where the code contains all message passing statements required to communicate between the six processes (this gives communication losses even when employing one processor!). The continuous line shows the total elapsed time measured by the client process, hence including all loads required for communication and synchronization between all processes. The broken line shows the pure CPU time required to update the flow quantities on each server, meas-

ured (averaged) locally on all servers. The difference between these values is shown in the dotted line. The relatively little losses caused by message passing and synchronization in case of (physically relevant!) parameters is easily recognized.

Fig. 1 shows the computing times depending on the number p of employed physical processors (Model 550). The expected timing model $T(p) = (A + C)/p$ (with A = arithmetic load, C = communication load) is clearly verified (where $nx = 160$, $ny = 80$ according to those realistic parameters used for the physical examples). A parallel speedup of 5.5 results when employing 6 processors.

To summarize, the most important result of these test computations is the following: the losses caused by parallelization (communications etc.) are negligible in case of realistic discretization parameters, namely those used for the physical examples. This confirms our previously stated hypothesis that the present algorithm is well suited to be run on the specific parallel environment of a cluster of high-end workstations.

PHYSICAL DETAILS

Figs. 2 and 3 show instructive physical details of the knock damage problem, in comparison between theory (numerical simulation) and experiment. The experiments have been carried out in the diploma thesis [2.] using a knock simulator and Cranz-Schardin camera.

Fig. 2 shows the pressure evolution due to an incoming explosion wave (here: detonation wave at Mach number 3.8) at the reentrant corner of the cylinder-piston cross section in the knock damage simulator. Here the comparison between theory (upper picture) and experiment (neglect here the nonphysical oscillations in case of the experiment!) shows excellent qualitative and even quantitative agreement and serves to validate our code.

In Fig. 3 we investigate the complex phenomenon of pseudo-shockwaves leaving the crevice. Let two shock waves enter successively the channel, namely in a way that the second wave enters in just that moment where the first one is reflected at the end wall of the channel. Both waves are then interacting, and they produce (besides of a shock wave turning again towards the end of the channel) a wave train, so-called *pseudo-shock waves* (which are due to massive viscous-inviscid interaction between the reflected shock wave and its boundary layer). Rigorous quantitative investigation of this kind of phenomena has been started only recently (see e.g. [6.]). Fig. 3 shows the isopycnics of multiple pseudo-shock waves shedding from the gap into the inlet combustion chamber, a phenomenon particularly observed for small gap

widths of the order of 20 to 30 μm. This computation takes several hours of computing time.

Concerning the influence of exothermic chemical reactions, further numerical results show a reduction of shock-wave-boundary-layer interaction due to local separation and vortex formation that substantiates a damping effect of heat release in nonequilibrium flow.

CONCLUSIONS

A new algorithm has been presented to simulate the knocking of spark ignited Otto engines. Based on the compressible Navier-Stokes equations as the mathematical model, the splitting technique serves to reduce the problem numerically. Domain splitting leads to coarse-grained parallelism which fits favorably with the specific architecture of a high-performance workstation-cluster. Some advanced physical details illustrate the potential of our new code, namely by comparison between calculation results and experiment.

REFERENCES AND FIGURES

1. P. Altevogt, A. Linke, "Parallelization of the two-dimensional Ising model on a cluster of IBM RISC System/6000 Workstations", *Parallel Computing*, in press.

2. J.O. Beismann, "Experimentelle Simulation der Klopfschaedigung im Kolbenringspalt", Diploma thesis, Daimler-Benz AG (1991)

3. F.K. Hebeker, R.R. Maly, S.U. Schoeffel, "Numerical simulation of reactive flow on the IBM ES/3090 vector multiprocessor", *IBM Systems J.* **31** , 788-797 (1992).

4. C. Hirsch, *Numerical Computation of External and Internal Flows*, Vol. 2, New York etc. (1990).

5. J.J. Kloeker, "Shock induced self-ignition of a reactive gas mixture in a L-shaped duct", *Numerical Combustion*, Proc. of 3rd Intl. Conf. at Antibes (1989).

6. H. Sugiyama, H. Takeda, Z. Zhang, F. Abe, "Multiple shock wave and turbulent boundary layer interaction in a rectangular duct", in: H. Groenig (ed.), *Shock Tubes and Waves*, Proc. of 16th Intl. Symp. at Aachen 1987, pp. 185-191, Weinheim (1988).

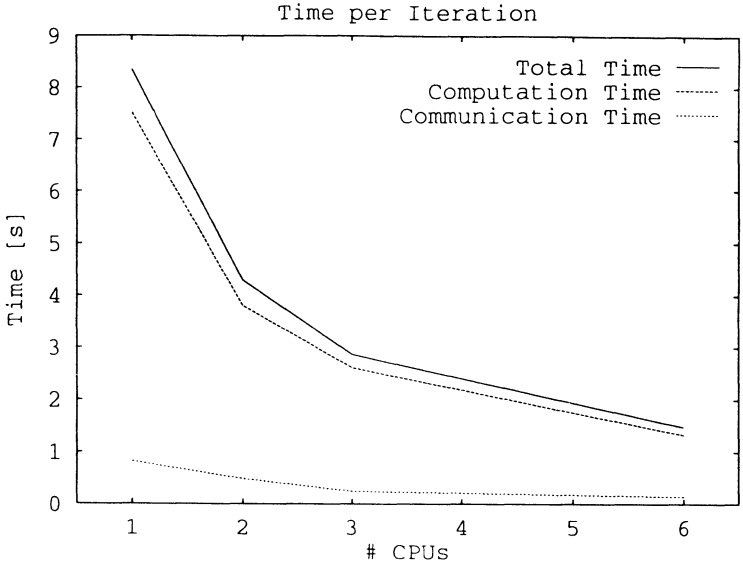

Figure 1. Time measurement: total code distributed over different number of processors

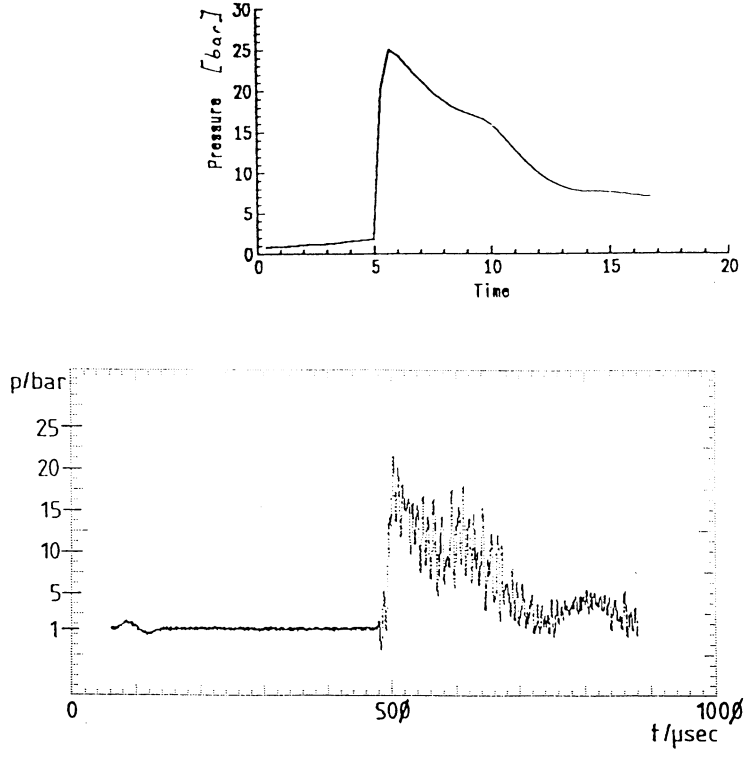

Figure 2. Pressure history due to incoming detonation wave just below reentrant corner: numerical simulation (upper) vs. experiment (lower fig.)

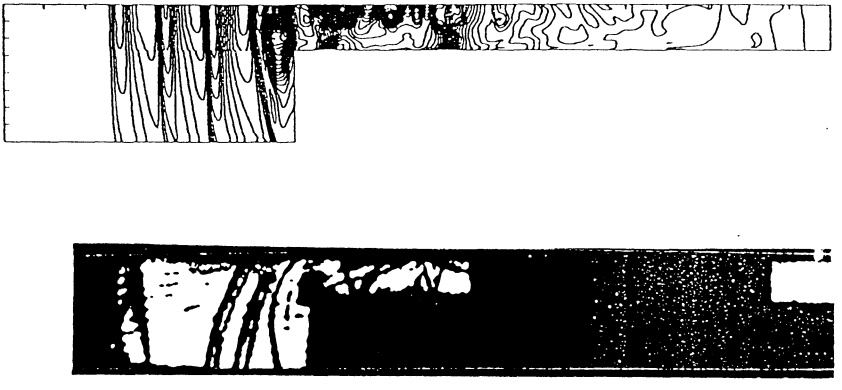

Figure 3. Pseudo-Shockwaves caused by massive viscous-inviscid interaction which leave the gap: numerical simulation (upper) vs. experiment (lower fig.)