Zeszyty Naukowe
Wyższej Szkoły Informatyki i Zarządzania w Rzeszowie

7/1998 Rok II (8)

ANNA KUCABA-PIĘTAL, MARCIN PAPRZYCKI

PARALLEL COMPUTING FOR COMPUTATIONAL FLUID DYNAMICS

RÓWNOLEGŁE OBLICZENIA DLA OBLICZENIOWEJ DYNAMIKI PŁYNÓW

INTRODUCTION

The main goal of this paper is to summarize the possibilities brought by parallel computers to the area of fluid mechanics. However, many of the issues raised here can be also applied to a broadly understood area of scientific computing. Fluid mechanics is one of the domains of physics substantially benefiting from the development of new numerical algorithms and a rapid increase of computational power of modern computers. The equations of fluid dynamics (Navier-Stokes equations) are partial differential equations containing both, first and second derivatives in spatial coordinates and first derivative in time. The time derivative appears as a linear term while the spatial derivatives quite often appear nonlinearly. Also, except for very limited special cases, the analytical solutions of these complicated systems of equations are not available. Only application of numerical approximations and computer generated solutions allowed solution of many real-life problems involving: forces on bodies (skin friction, drag, lift), efficiencies (turbine, diffuser), heat transfer, weather prediction and so on. These solutions lead to the development of practical applications (for more details see [2], [7]). Nowadays, problems concerned with computation of the boundary layer, turbulence, separation, vorticity and general unsteady flows for complex geometries of the flow fields are of particular interest to the researchers in the field. All these problems involve the solution of highly nonlinear Navier-Stokes equations. Progress in finding solutions is slow and involves adapting the well-known methods to modern high-performance computers or construction of new algorithms targeting the new computer architectures directly.

FEATURES OF NUMERICAL APPROXIMATIONS

The invention of the digital computer helped to develop a new meaning of the concept of approximation. This new understanding concerns the numerical approximation to the set of equations representing a mathematical model of a physical system. Approximate numerical solutions are sought for real-world problems, for which no analytical solutions exist. Figure 1. shows a diagram of the interaction between the real world, the mathematical model and the numerical approximation.

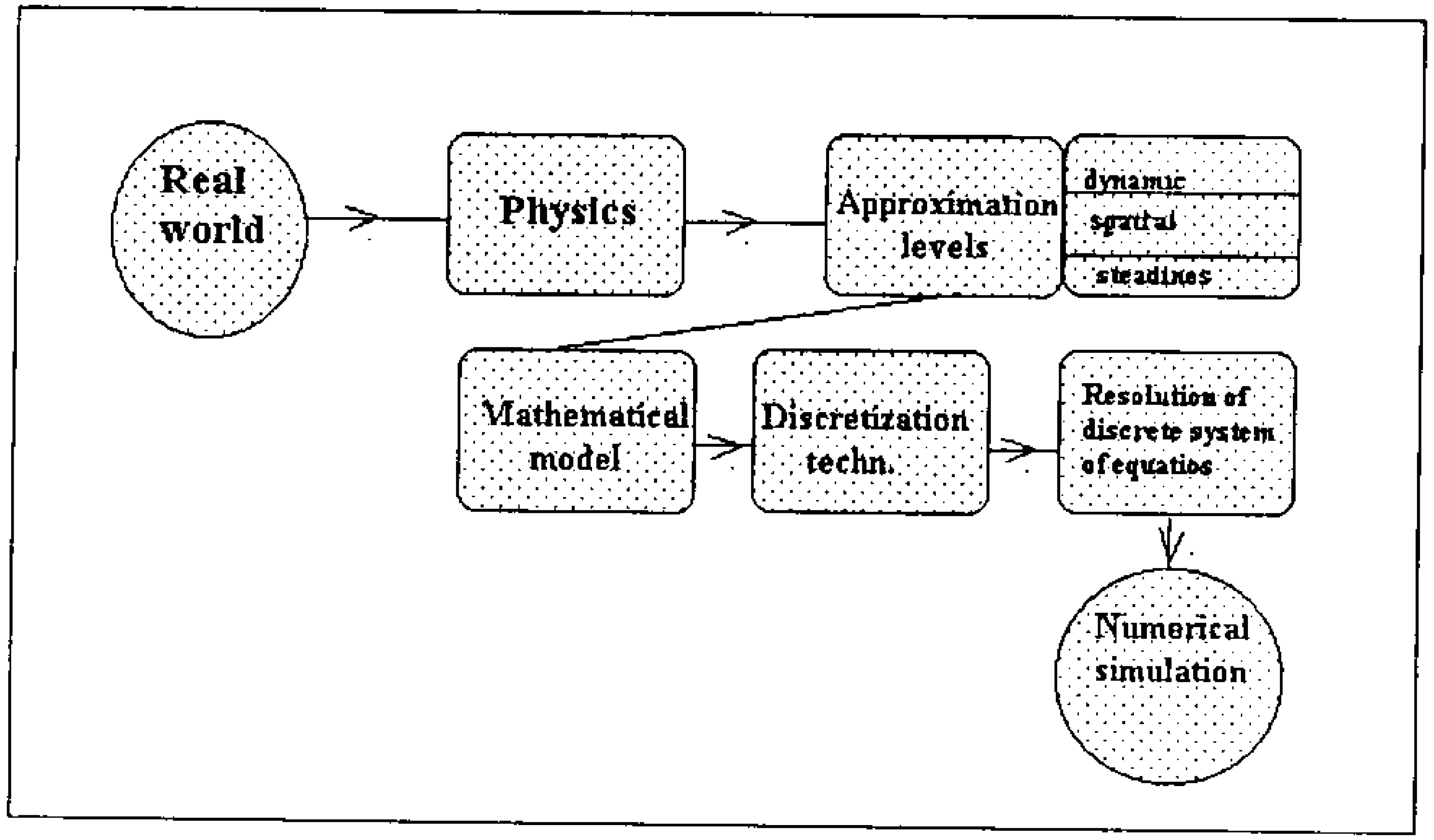

Figure 1. Numerical approximation.

All discretization techniques convert a given system of equations into a system of algebraic equations that can be solved by computer. Discretization introduces an error, which is connected with number of the grid points. Due to the fact, that to solve the Navier-Stokes equations we need approximations of derivatives in both space and time, each of the standard numerical approaches applied to it requires the very large number of grid points. This number depends on the dimensionality of the problem, its geometric complexity and the severity of the gradients of dependent variables. For example: the model of the pressure contour of the plane of symmetry of a car needs about 50,000 vertices in space [13] while when a flow over a periodic array of cylinders was calculated 40,000 vertices were needed [13]. Moreover, at each grid point, each dependent variable and a number of auxiliary variables must be stored in the computer mem ory. Therefore it is easy to see that the computational requirements for the hig resolution solution of Navier-Stokes equations are: very fast computers (capab of delivering Teraflop performance) with very large memories (of order of 2 Gigabytes or more). In addition, to be able to store the intermediate results (instance for the purpose of visualization), a very large I/O bandwidth is requir as well as extremely large data storage capability (of order of thousands Terabytes). Let us look into the history and the current state of the art of hi performance computers.

PARALLEL COMPUTER BASICS

Hardware

The technological development of computers during the last thirty ye produced machines faster by many orders of magnitude and substantially creased the size of available memory. The late 1970s and the 1980s saw t emergence of machines with vector-processing units and, later, multiple exe tion pipelines inside of a single processor. Currently the most powerful sin processor (Cray T-9x series vector processor) is capable of delivering appro mately 2 Gigaflops of sustained performance on dense matrix applicatio Since its clock is running at 1.5 nano-seconds it can be expected that we are f reaching the limits of what can be achieved on s single processor. It is thus cl that only multiprocessor computers will be able to deliver enough computatio power to solve the equations of fluid dynamics.

The development of parallel computers can be traced to the early 19 when Illiac IV was constructed at the University of Illinois. Since then the te nology evolved considerably and became mainstream. Nowadays, it is qu typical to buy a PC with multiple (usually up to four) processing units (es cially when such a PC is used as a network server). In addition each proces has multiple execution path for both integer and floating point arithmetic. T changes occurred also on the cutting edge of parallel computer architectu Below we discuss three important areas of change: model of computati memory arrangements and changes among the parallel computer manufactu (for more information see [9,10]).

The computational model. Initially two basic types of architectures h been tried: single instruction multiple data (SIMD) and multiple instructi multiple data (MIMD). In the SIMD model a very large number of process (usually relatively weak -- even one-bit processors) works synchronously. W the program is executed each processor performs the same operation on a dif ent data element (or no operation at all). Typically such machines consisted a mesh of about 512-1024 processors, but machines with 4000+ processors have been also tried (e.g. Connection Machine). In the MIMD model a parallel computer consist of a small to medium number of (relatively powerful) processors. Typically, when a program is executed, each processor contains a copy of the code and works independently on its "assignment." Processors are expected to communicate relatively infrequently. Over time the flexibility of the MIMD model was recognized to be more valuable than the advantages of the SIMD approach and most of the currently produced machines are based on the MIMD model. Observe that, in principle, it is possible to program an MIMD machine using an SIMD model, but the reverse is not true.

Memory arrangement. Two memory architectures have been initially experimented with: shared memory and distributed memory. In the case of shared memory all processors have a uniform access to a large global memory (sometimes divided into independent sections to improve the access rate). The connection between the processors and memory is usually based on a bus or a crossbar switch. Due to the bandwidth saturation such systems were limited to about 32 processors in case of bus-based systems and about 128 processors for the crossbar-based machines. In case of distributed memory computers, each processor had its own memory and the data exchange occurs via a network (usually a hypercube, a mesh or a torus). The local memory was rather small for most SIMD systems and quite large for the MIMD distributed memory machines. Since the data access was no longer a limiting factor (all the data is local to some processor) distributed memory machines reached even 512 and 1024 relatively powerful processors (e.g. i860 based Intel hypercubes). Over time the two models started to merge as an attempt was made to build computers that benefit from both, the simplicity of the shared memory model, and the capability for connecting a very large number of processors in the distributed memory machines. The dynamic shared memory model has been developed and became the leading approach used currently by most of the computer manufacturers. In this model the physical memory layout remains distributed, or consists of a mixture of shared and distributed memory components. At the same time the logical memory is to be viewed by the programmer as a uniform address space shared memory.

Evolution of parallel computer manufacturers. Initially a relatively large number of vendors attempted at designing and manufacturing high performance parallel computers. When the support provided by the US Government decreased (due to the end of the Cold War) only very few manufacturers were able to sustain competition. Most of the small parallel computer vendors either went out of business (e.g. Alliant, Thinking Machines, Kendall Square) or changed their production profile (e.g. BBN, Sequent). Even large and relatively successful companies like Cray Research Inc. were purchased by competitors (Silicon Graphics Computers). Currently the following companies can be considered as major vendors providing high performance computers. In US: HP-Convex (th Exemplar series -- dynamic shared memory), Silicon Graphics-Cray (Cray T3- -- distributed memory, Cray J-9x and T-9x -- shared memory vector machine SGI Power Challenge -- shared memory, SGI Origin -- dynamic shared mem ory), IBM (SP -- distributed memory), Sun (HPC Cluster -- distributed memor and Tera (multithreaded machine). In Japan: NEC (SX-5 -- dynamic shar memory) and Fujitsu (VPP 700 -- distributed memory, vector processors). At th same time, the fastest computer in the world (as of June'98 issue of the TOP5(list) has been custom build out of 9125 Pentium Pro processors running at 2(MHz. This machine (known as ASCI Red) is capable of 1.3 Tflop sustain performance.

Finally, let us return to the issue of mainstream parallel computing. One the interesting approaches to cheap parallel computing is the Beowulf Proje (http://cesdis.gsfc.nasa.gov/). The main idea is to connect PC's running Linu operating system using Ethernet and thus to create a parallel computer. Cu rently a Beowulf machine consisting of 16 dual 400 MHz Pentium II processo together with a networking hardware and software as well as software suppor ing parallel computing can be purchased for approximately $65K.

Programming environments

The software support for parallel computing also has evolved overtime, b seems remain behind the hardware technology to (as in other areas of comput science). Until the introduction of Java programming language most of the sc entific computing oriented parallel processing has been done using Fortran a C. With Introduction of Fortran 90 (and its slight modification -- Fortran 9 elements of matrix and vector oriented parallel processing capabilities ha been introduced to the language. Currently, High Performance Fortran (HP became the language with a complete support for parallelism. The jury is st out if this attempt to revitalize an old language will be successful. In C no dire support was added to the language and multiprocessing was supported by cal to libraries like PVM [11] or MPI [12]. PVM (Parallel Virtual Machine) a MPI (Message Passing Interface) are two libraries of routines that facilitate t multiprocessing execution of both Fortran and C based codes. Both products a freely available from Oak Ridge and Argonne National Laboratories respe tively. PVM was an earlier product and became extremely popular in late 198(Currently its support base is decreasing and the MPI becomes a de facto standa for writing distributed parallel applications for scientific calculations. Most man facturers of parallel hardware provide users with optimized versions of the M library. Finally, a word about the Java programming language. There is an i creasing interest in developing scientific computing based applications in Ja

However, since scientific computing requires full utilization of underlying computer hardware, until Java does not improve its performance its utility will be rather limited.

CURRENT OPEN QUESTIONS

Practical flow problems often require a calculation of the computational solution in complicated three-dimensional domains, involving thousands of nodal unknowns. For such "large scale" problems the assessment of the relative computational efficiency of competing numerical schemes is an essential prerequisite to the main body of the computational investigation. It is important to observe that the efficiency of the overall method results from both the efficiency of the numerical method and the efficiency of its implementation on parallel computers. Let us briefly discuss problems arising when adopting to parallel computations the main computational methods: finite difference method (FDM), finite element method (FEM), finite volume method, and spectral methods.

When the fluid flow is studied the FDM combined with domain decomposition methods are widely used and the fourth order algorithm for the Navier-Stokes equations was recently constructed [4]. For the FEM, adaptive strategy is usually involved and an adaptive code for the Euler equations (simplest form of N-S eq.) has been developed. FDM and FEM methods are based on relations connecting only nearest neighbors. They are thus attractive for parallel computations as the overall communication between processors is reduced. However, a large number of problems related to grid generation, especially for general unstructured flow domains.

Many codes used in the aerospace industry for solution of the both Euler and Navier-Stokes equations rely on multi-block grids. The computational grid is partitioned into a number of blocks each containing a number of computational cells. This data structure is suitable for parallel computation on a limited number of processors. Each processor can handle calculations in one or more blocks in the grid. Next, the discretized equations are solved iteratively.

The boundary methods lead to the solution of dense systems, because all elements interacts with all other elements. The mapping of such dense systems to parallel architectures is a non-trivial, but solvable, problem. This approach is widely used in problems involving hydrodynamics interactions of many bodies at flows with low Reynolds numbers [6].

Recently, a number of new (sequential) numerical techniques attempting to solve Navier-Stokes equations have been developed. For instance, the stochastic vortex-blob methods [9], large particle methods [1] and asymptotic solution methods [1] are constructed and intensively studied. It is still unclear how well will they parallelized.

In addition to these problems, one needs to pay careful attention to the in terface between the program and the underlying computer architecture and th programming environment. Some methods will run better on a vector processor while others will be more efficient on RISC-based processors. Some method naturally match the SIMD programming model (e.g. some models of micro flow), while others naturally match the MIMD model (particularly the larg granularity multi-block methods).

CONCLUSIONS

Our observations presented in this paper can be summarized as follows:

- Only multiprocessor systems can deliver enough computational power to b able to solve large practical problems in fluid dynamics.
- Parallelization of existing numerical methods may not be easy and a lot o work will be required to take full advantage of multiprocessor systems.
- Various aspect of the computational simulation should be considered jointl because of a symbiotic relation between them can be attained.
- It is imperative that one considers the choice of the mathematical model, th mesh generation technique, the discretization process and the solution to gether as a package.
- One cannot design algorithm totally independent of software and hardwar issues.

REFERENCES

[1] *Encyclopedia of Mathematics*, vol. 5., Kluver Academic Publishers Dodrecl 1990.

[2] C. A. J. Fletcher, *Computational Techniques...*, Springer-Verlag, Berlin, 1990.

[3] W. Gentzch, *Computational Fluid Dynamics*, Agardograph, NATO, 1988.

[4] J. Rokicki, J. M. Florian, *Journal of Computational Physics*, 116, 79-96, 1995.

[5] *Development of Parallel Implicit Navier-Stokes Solvers on MIMD Multiproces System*. AIAA Paper 93-0062, 1993.

[6] S. Kim, S. Karilla, *Microhydrodynamics*, Butterworth-Heinemann, 1991.

[7] C. Hirsch, *Numerical Computation of Internal and External Flows*, 1994.

[8] J. Szumbarski, P. Wald, *The stochastic vortex flow simulation of unsteady visco flow in a multiconnected domain*, Second Inernational Workshop, Montreal 199 p. 125-126.

[9] R. M. Hord, *Parallel Supercomputing in SIMD Architectures*, CRC Press, Bo Raton, 1990.

[10] R. M. Hord, *Parallel Supercomputing in MIMD Architectures*, CRC Press, Bo Raton, 1993.

[11] MPI Tutorial:
http://science.nas.nasa.gov/Groups/SciCon/Tutorials/MPIintro/toc.html.
[12] PVM Information: http://www.netlib.org/utk/papers/pvm4/
[13] R. Glowinski, *Supercomputing and the finite element approximation ...In Recent Advances in Computational Fluid Dynamics*. Vol 43. Springer-Verlag, 1990.

STRESZCZENIE

SUMMARY

Nowe algorytmy numeryczne oraz programowanie równoległe są narzędziami umożliwiającymi rozwiązywanie złożonych obliczeniowo zagadnień mechaniki płynów. Przegląd ostatnich osiągnięć w obu dziedzinach jak i nadal istniejących problemów został przedstawiony w artykule.

New numerical algorithms and parallel computing provide tools for studying previously unsolvable problems in fluid dynamics. A summary of recent development in both areas and the perspective on the future research is presented.