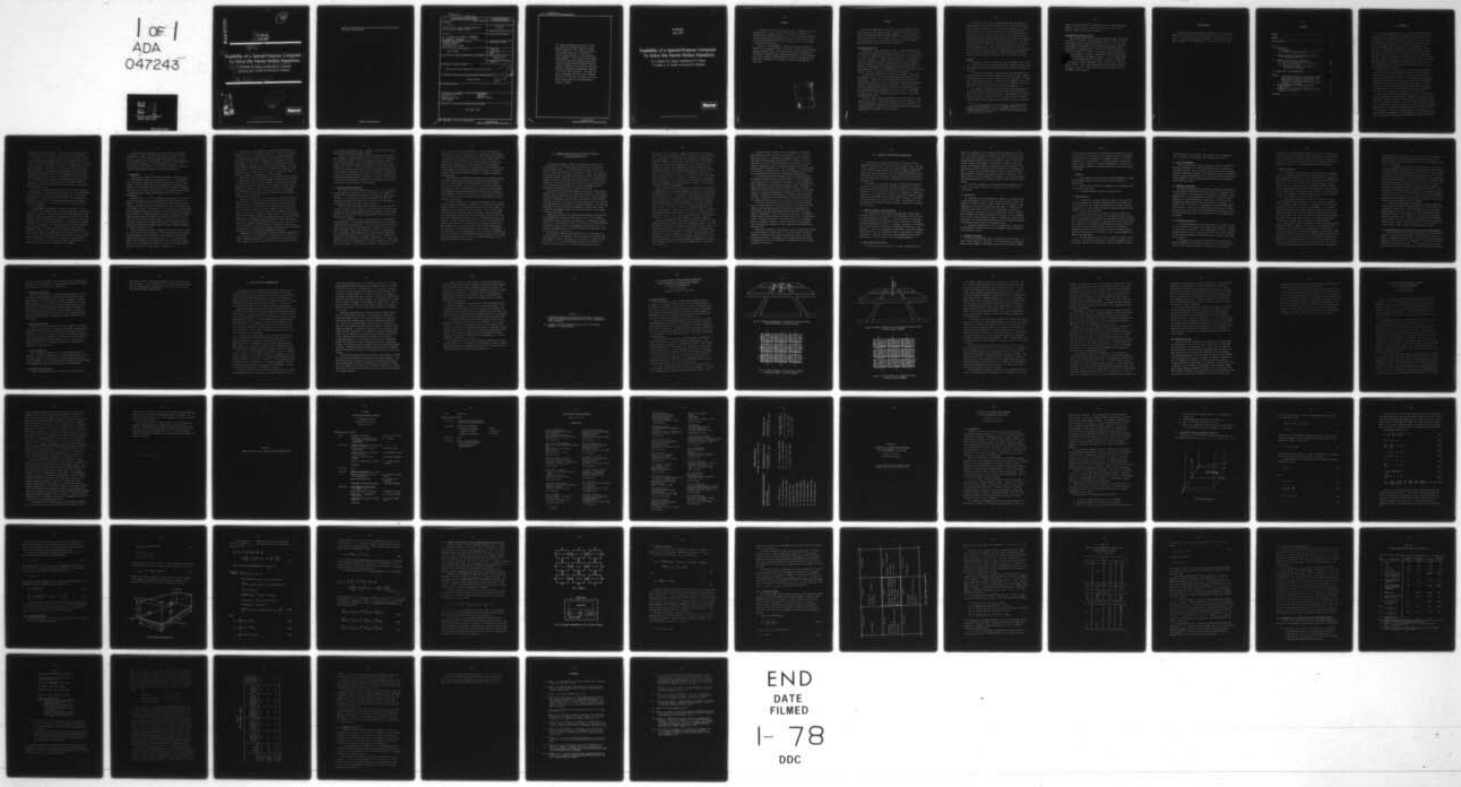
AD-A047 243 RAND CORP SANTA MONICA CALIF F/G 20/4
FEASIBILITY OF A SPECIAL-PURPOSE COMPUTER TO SOLVE THE NAVIER-S--ETC(U)
JUN 77 E C GRITTON, W S KING, I SUTHERLAND

UNCLASSIFIED R-2183-RC NL

1 OF 1
ADA
047243



END
DATE
FILMED
1-78
DDC

12 mc

14 R-2183-RC
11 June 1977

12 75p.

6

Feasibility of a Special-Purpose Computer To Solve the Navier-Stokes Equations.

10 E. C./Gritton, W. S./King, I./Sutherland, R. S./Gaines/
C./Gazley, Jr., C. Grosch, M. Juncosa, H. Petersen

9 Interim rept.,

Rand
SANTA MONICA, CA. 90406

1472

296 600

AB

Published by The Rand Corporation

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: R-2183-RC	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Feasibility of a Special-Purpose Computer to Solve the Navier-Stokes Equations	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): E. C. Gritton, W. S. King, I. Sutherland, R. S. Gaines, C. Gazley, Jr., C. Grosch, M. Juncosa, H. Petersen	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Rand Corporation, 1700 Main Street, Santa Monica, Ca. 90406	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: same as above	12. REPORT DATE: June 1977
	13. NUMBER OF PAGES: 66
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No restrictions

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Navier-Stokes Equations
Viscous Flow
Boundary Layer Flow
Fluid Dynamics
Hydrodynamics
Computers
Numerical Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This report summarizes the results of a two-day workshop at the Rand Corporation, Santa Monica, March 9-10, 1977, where participants discussed the feasibility of developing a special-purpose computer to solve the Navier-Stokes equations. The conclusions from a Rand study and the workshop consensus suggest that a special-purpose, parallel-processor machine capable of important fluid dynamics simulations might be technically and economically feasible in the early 1980 time period. The report presents a conceptual design for such a computer, an analysis of how it can be used to solve the Navier-Stokes equations, and performance estimates. (Author)

R-2183-RC
June 1977

Feasibility of a Special-Purpose Computer To Solve the Navier-Stokes Equations

E. C. Gritton, W. S. King, I. Sutherland, R. S. Gaines,
C. Gazley, Jr., C. Grosch, M. Juncosa, H. Petersen

Rand
SANTA MONICA, CA. 90406

PREFACE

This report describes the deliberations of a workshop held at The Rand Corporation on March 9-10, 1977 to study the feasibility of a special-purpose computer designed to solve the Navier-Stokes equations--the basic expressions describing fluid flow. It includes the results of a preliminary Rand study of such a computer, used as a "straw man" by the workshop participants.

Interest in fluid mechanics, coupled with early and continuing research in the computer sciences, led Rand to sponsor this study using corporate research funds. The report should interest researchers in the fields of computer science, numerical analysis, and fluid mechanics, particularly persons concerned with the application of high-speed computational techniques to the solution of the governing equations of fluid flow.

SUMMARY

This report summarizes the results of a two-day workshop at The Rand Corporation, Santa Monica, where participants discussed the feasibility of developing a special-purpose computer to solve the Navier-Stokes equations for a fairly general class of fluid mechanics problems. It includes the text of a preliminary Rand study of such a computer, which the workshop participants used as a straw man.

PRELIMINARY RAND STUDY

In the fall of 1976 and winter of 1977, a preliminary concept for a Navier-Stokes computer was developed at Rand. This concept consists of an array of 10,000 identical large-scale integrated circuits arranged in a 100 × 100 matrix. To reduce wiring congestion, each processor is limited to communication with its nearest neighbors. Each processor includes some space for storing algorithms and carrying out simple calculations. Three-dimensional problems are solved by having each processor represent a point in a two-dimensional plane. The storage on the processor is used to represent data in the third dimension.

The potential benefits of this design are enormous. Each processing element can carry out a 64-bit fixed-point multiplication in five microseconds. Though this is quite slow compared with the speed of today's best computers, an array of 10,000 such elements performs simultaneously 10,000 multiplications in that time, or approximately two billion multiplications per second. Given that integrated circuits of the type described may be produced in the near future for about $100 per chip or less, this architecture concept can lead to a very large performance/cost ratio.

An entire large problem (approximately 10^6 points) can fit on this computer at one time. Thus, there would be no need to break the solution up into sequential pieces or to reload the array frequently. The resulting saving in communications time and other overhead contributes substantially to the power of such a machine.

To exploit the full power of the array computer, the numerical algorithms must match the machine architecture. The preliminary Rand study included an analysis of a finite-difference approach for solving the Navier-Stokes equations that was found to be particularly well suited to the array processor's architecture. This approach makes use of finite-difference approximations in the xy plane and for the time variable but applies transform methods to the z direction. The results of a model problem* showed that the problem could be solved at a rate of about 0.35 sec per time step. The same problem run on a fast conventional computer (e.g., a CDC 7600) takes ~ 82 sec per time step. The conclusions of Rand's initial study suggested that a special-purpose, parallel-processor machine capable of important simulations might be technically and economically feasible in the early 1980 time period.

WORKSHOP

To explore the concept further, Rand held a workshop on March 9-10, 1977. Researchers from the computer-technology, numerical-analysis, and fluid-dynamics communities discussed the practicality and timeliness of this idea.

The workshop participants generally agreed that appropriate technology now exists to consider a machine of the nature proposed capable of attacking problems of theoretical and practical importance. It would have a computing capacity of at least an order of magnitude superior to the largest current computers and could be built within the next 3 to 5 years at a cost substantially less than current computers.

Both adequate algorithms and sufficient numerical experience are available for the solution of the relevant partial differential equations. The family of designs considered in the workshop offers the potential for efficient direct Navier-Stokes simulations of nonlinear laminar instabilities, boundary-layer transition and perhaps turbulent

*The problem consisted of the 3-D solution to the Navier-Stokes equations for incompressible flow of a constant-density fluid in a boundary layer adjacent to a rigid, impermeable, no-slip wall. The problem was modeled on an array of 10,000 cells with 128 grid points in the cross-stream direction.

spots, a limited simulation of turbulence at low to moderate Reynolds numbers, and a simulation of large-scale turbulent flows at high Reynolds numbers using sub-grid modeling.

RECOMMENDATIONS/PROPOSED ACTION

Both the Rand study and the workshop consensus suggest that it may be feasible to construct a Navier-Stokes computer by the early 1980s. A two-stage research program should follow. In the first stage, both the mathematical problems and the machine-design problems should be explored in an integrated manner in order to confirm the hypothesis that such a machine is feasible. This initial stage should include a coordinated program of research on algorithms, microprocessor design, machine architecture, and computer software. The second stage should consist of prototype hardware development, including the design and construction of a small experimental array of processors to test the design concepts. This research program would provide the basis for a decision to construct a full-scale array processor machine to solve the Navier-Stokes equations.

ACKNOWLEDGMENTS

The authors want to thank Stephen Lukasik, who originally recommended that this study be performed and who guided the initial organization of the research program and workshop. We gratefully acknowledge the guidance and recommendations of the workshop participants.

CONTENTS

I. INTRODUCTION

The fluid-dynamics community has for some time been fascinated by the possibility of obtaining direct, numerical solutions to the Navier-Stokes equations for turbulent flows and flows undergoing transition to turbulence. Fluid-flow calculations were among the earliest applications of digital computers. For viscous-flow calculations, either the Reynolds number was sufficiently low for the flow to be truly laminar, or turbulence was introduced in a phenomenological way (e.g., the use of a "turbulent viscosity") so that the mean flow was smooth. These phenomenological models are still employed and will certainly be used in the foreseeable future for many flow calculations of major interest. However, the models are empirical and success is not assured when variations in flow parameters are encountered. Thus, there is a continuing interest in developing the capability for direct solution of the Navier-Stokes equations.

During the fall of 1976 and winter of 1977, The Rand Corporation conducted a corporate-funded study in collaboration with members of the California Institute of Technology and Old Dominion University staffs to investigate the feasibility of developing a special-purpose computer that would simulate viscous flow around realistic geometries, including some aspects of the development of turbulence. The Rand study focused on special-purpose parallel-array computers because we anticipate that even in the early 1980s the fastest general-purpose computer will be insufficient to cope with such simulations.

Increasing the amount of parallelism in computer solutions has long been recognized as the best way to achieve substantial performance improvement. Rand has evolved an approach which accomplishes this for a class of problems. The Rand concept is to design the computer architecture around the mathematical structure of the Navier-Stokes equations. Applications to other specific problem areas and differential equations may be feasible. To ensure maximum performance improvement, the computer should be designed by a multidisciplinary team of experts in computer architecture, computational fluid mechanics, numerical analysis,

algorithms, and software systems. This will ensure that the machine architecture is tailored correctly to the simulation requirements.

The computer as conceived by Rand consists of an array of 10,000 identical large-scale integrated (LSI) circuit processors arranged in a 100 × 100 matrix (see Appendix A(I) by Dr. Ivan Sutherland). To reduce wiring congestion, each processor is limited to communication with its nearest neighbors. Each processor can carry out simple calculations (add and multiply). Three-dimensional problems are solved by having each processor represent a point in a two-dimensional plane and the storage on the processor is used to represent data in the third dimension. Computation, data transfer, and communication are carried out in parallel. The result is that both the total number of addition/multiplications and words transferred per second can be large at relatively low overall cost.

The reason for developing such a computer is to make it possible to perform complex fluid-flow calculations *routinely*. The calculations can then be used as standard tools by researchers in fluid dynamics and designers of military and civilian systems. This capability contrasts with the potential of contemporary computers for performing these calculations once every few months by calculating for hundreds of hours per problem.

A simple example may, perhaps, illustrate this point. In 1952, L. H. Thomas[(1)] used about 300 hours of time on the Selective Sequential Electronic Computer of IBM to compute 18 points around the neutral stability curve for plane Poiseuille flow, about 17 hours per point. This calculation was repeated some years later,[(2)] as a check of a different numerical method. It took about 20 minutes on an IBM 7094 (1.1 minutes per point). In addition to the least stable mode, 29 other modes were also calculated at each point. Quite recently this calculation was repeated,[(3)] again as a test problem, on a medium sized computer. It only required 2 minutes to compute the first 30 modes for all 18 points (6.7 sec per point). The great speedup in this type of calculation has made it possible to use linear-stability theory as a routine design tool.[(4)] The speedup is not due to more efficient algorithms but is largely due to the increase in computer size and speed.

Historically, the performance of serial computers has improved dramatically. However, this improvement has not been sufficient to allow the solution of the Navier-Stokes equations directly. Alternative concepts should be investigated, and the Rand study demonstrates that parallel processing is a viable candidate architecture. The central theme of this report is to study the application of this architecture to the numerical simulation of fluid-flow problems.

A. BACKGROUND

Some time ago, Corrsin[5] estimated the range of scales of motion in a true numerical simulation of a turbulent flow. A true simulation is one in which all significant scales of motion are accurately represented. Corrsin concluded that the requirements were far beyond the capability of any computer existing at that time. Emmons[6] reexamined this question in 1970 and concluded that, in the not-too-distant future, low-Reynolds-number turbulent flows might be within the range of investigation.

Recently, Case et al.[7,8] reconsidered the question: Is the direct numerical simulation of turbulence possible with the most modern computers? They found that a few simulations had already been carried out at moderate Reynolds numbers, but concluded that, even with the most modern computers likely to be available in the next few years, only a limited range of Reynolds numbers can be treated. They reached these conclusions: The construction of a high-performance conventional computer to do high-Reynolds-number simulations appears impractical for the next 10 to 15 years; this problem could be attacked by a very large (on the order of 10^6 cells) cellular-array computer, which might be feasible in the mid-1980s; and a smaller-array computer useful for many problems appeared to be feasible in the near future.

These conclusions led them to suggest that a possible solution would be to build a Navier-Stokes computer. They envisioned a special-purpose computer designed to solve the Navier-Stokes equations, and only the Navier-Stokes equations, with maximum efficiency. By using turbulence models, such a computer could further be used for the numerical simulation of those fluid-dynamic problems, which, even on the Navier-Stokes computer, are beyond the range of direct simulation.

A capability to perform direct simulations and complex "model" computations would be a powerful resource for researchers and designers of military and civil fluid-mechanics systems. Computer simulations of a fluid-mechanics problem offer several advantages over physical experiments. These include: flexibility for independently varying flow parameters, the capacity to simulate complex geometries not possible during an experiment, control of boundary conditions, and quantitative estimates of flow characteristics impossible to measure. These advantages are quite important to a designer of systems who needs to predict aerodynamic or hydrodynamic forces accurately.

Many practical problems require the accurate and rapid calculation of fluid-dynamic forces and flow phenomena, including aircraft and missile design, meteorology and global weather forecasting, environmental simulation modeling, and submarine and ship design. Progress in the quantitative analysis of these problems is limited to a great extent by the computing power available for the simulation of flow phenomena. Advances in these areas would benefit greatly from the development of a "Navier-Stokes" computer.

During the past few years, large advances in computer technology have taken place. Although we are now entering the era of the supercomputer that will process hundreds of millions of instructions per second, researchers are interested in performing more accurate simulations of physical phenomena, creating a demand for still larger computers. In spite of the present impressive performance of serial processing general-purpose computers, an upper limit to their capabilities seems close. Even using large-scale integrated micro-circuits, improved heat-extraction techniques, innovative hardware design, and creative numerical techniques, improvements only on the order of factors of 2 to 5 are anticipated in the early 1980s with conventional architecture. This performance is inadequate to simulate numerically many fluid-dynamic problems without questionable phenomenological modeling.

Communication limits performance improvement because instructions, data, and results are transmitted at a finite rate over a finite distance. Sequential handling of the massive amount of data generated in large computers is thus rate-limited and this is a definite bottleneck

in increasing computational speed. Attempts to overcome this problem have produced more complex hardware designs.

To penetrate this performance barrier, researchers have proposed the concept of a computer consisting of a parallel array of thousands of fast microprocessors. Data are divided and distributed among a number of microprocessors that perform identical calculations. The architecture of such a computer is strongly dictated by the characteristics of the problem to be solved; so one must incorporate the nature of the simulation problem in the design of the hardware, the numerical analysis, and the development of software. A numerical simulation that requires many different types of operation on a single set of data will not exploit the advantages of this parallel arrangement, and many processors will stand idle while a few are working.

B. PARALLEL-PROCESSOR COMPUTERS

The concept of an array computer is quite old. (See, for example, the discussion and description of the Von Neumann array computer in Thurber.[9]) Far fewer array computers than sequential computers have been built in the past because of system complexity and high cost. Individual processors were expensive, and reproducing large numbers of components increased costs plus control overhead.

ILLIAC IV is the best known general-purpose array processor.[9] It can be viewed as an 8×8 array of cells (processing elements, PE's) arranged in a grid communicating with nearest neighbors and end around. Each PE is a rather sophisticated general-purpose computer and has 2K 64-bit words of memory.

In a way, ILLIAC IV is both too big and too small for the Navier-Stokes equations. It is a general-purpose computer with more features than needed for the purposes here. It is also too big because it was designed in anticipation of major advances in a number of areas of computer technology. Some of these advances did not occur on schedule, so its performance and reliability did not reach original design goals. At the same time, ILLIAC IV is too small because an 8×8 array is inadequate for most problems in computation fluid dynamics. If the ILLIAC memory is used in the 32-bit mode, the 4K words of memory per PE can hold the data for about 4×10^2 grid points assuming 10 variables per grid

point. Therefore ILLIAC can hold about 2.5×10^4 grid points in PE memory--far too few for most problems in computational fluid dynamics. Increasing the number of grid points would require frequent loading and unloading of the PE memories. A realistic viscous fluid-dynamics problem will require at least 10^6 to 10^7 grid points to resolve the important and experimentally observed features of a high-Reynolds-number flow. The realization of a computer which can accommodate such a large number of grid points is suggested by the idea of a special-purpose, fluid-dynamics computer with tens of thousands of parallel microprocessors.

Recent advances in large-scale integrated circuit technology and reduction in manufacturing costs have removed the economic barrier to the use of large numbers of microprocessors in the array computer. Once the circuit design and set-up costs are incurred, chip reproduction costs are small. By using current technology in the chip design, as developed for widespread use throughout the electronics industry, the development costs and technical risks associated with the construction of the array machine are reduced without compromising the effectiveness of the design.

The initial conclusions of the Rand study suggested that an array computer capable of important simulations of the Navier-Stokes equations might be technically and economically feasible by the early 1980s. To further explore the practicality and timeliness of this concept, Rand invited researchers from the computer-technology, numerical-analysis, and fluid-dynamics communities to attend a workshop on March 9-10, 1977. This report summarizes the deliberations of the workshop along with the results of the initial Rand study, which was used as a straw man by the workshop participants. It also presents the conclusions derived from these discussions. Section II describes the workshop's organization and the results of the preliminary Rand study. Section III summarizes the deliberations of the working groups. Conclusions and recommendations are given in Section IV. Detailed presentations of Rand's initial design study and numerical analyses, a list of workshop participants, and the agenda are included in the appendixes.

II. WORKSHOP ORGANIZATION AND DISCUSSION OF RESULTS OF PRELIMINARY RAND STUDY

The workshop provided a forum for researchers involved in the various aspects of the problem; hardware architecture, software systems, and applications. It included informal presentations on each aspect to spark discussion and an interchange of ideas. The objectives of the workshop were to determine the status of computer technology and numerical techniques necessary for the development of a special-purpose computer designed to solve the Navier-Stokes equations; to discuss its feasibility in the early 1980s; and to generate from a group proficient in both numerical and experimental fluid mechanics suggestions for appropriate applications to key fluid-dynamics problems.

The first day of the two-day workshop was devoted to the informal presentations and the second to working group discussions. The subjects covered during the first day were computer technology, numerical simulation, and fluid-mechanics applications. The purpose of these presentations was to provide a focus for working group discussions and to provide an interdisciplinary interchange of ideas and information. An underlying theme was to isolate the pacing technologies and issues and to arrive at a preferred development program if the state-of-the-art could be shown to warrant such actions.

During the first day, Rand consultants and members of the research staff presented the results of their initial study. This included a preliminary concept of the basic architecture of the machine (see Appendix A(I) by I. Sutherland), a study of the numerical techniques that could efficiently exploit the parallel arrangement (see Appendix C by C. E. Grosch), and suggestions for possible fluid mechanics applications (see Section III(C) by C. Gazley, Jr.).

The straw-man computer concept developed by Dr. Sutherland is an array consisting of 10^4 identical large-scale integrated circuits that are arranged in a 100×100 matrix. Each processor is capable of storing about 1.6×10^4 bits of information arranged as 256 words of 64 bits or 512 words of 32 bits. To reduce wiring congestion, each

processor (cell) is limited to communication with its nearest neighbors. This feature minimizes the cost of interconnections, which could be a major expense. In addition to this simple local communication, there are communication lines from a central source to the 10,000 processors. To enhance the computing power of each processor, it is desirable for each to have a maximum of storage and some space for storing algorithms. Three-dimensional problems are solved by having each processor represent a point in a two-dimensional plane, and storage on each processor is used to represent data in the third dimension. This points out the necessity of having some memory associated with each processor. After reviewing Dr. Sutherland's concept during the two-day workshop, Dr. Gaines developed a variant on this concept, and this is included in Appendix A.

The potential benefits of the parallel architecture are enormous. We estimate that each processing element can carry out a 64-bit fixed-point multiplication in 5 microseconds (double-precision multiplication, each word 32 bits long). Though this is quite slow compared with the speed of today's best computers, an array of 10,000 such elements performs simultaneously 10,000 multiplications in that time, or approximately 2 billion multiplications per second. We also estimate that each processor can transfer one word to a neighboring processor in about 3 microseconds. Again, this is quite slow by today's standards; but, with 10^4 processors the data rate is about 3×10^9 words/sec, or about 10^{11} bits/sec. This is quite high when compared to the memory-to-CPU data rate of even the most powerful conventional computers.

These large computation and communication rates can lead to a very high performance/cost ratio, given that integrated circuits of the type described are produced in the near future for $100 per chip or less. An important characteristic of the array of processors just described is that, given some reasonable amount of memory per chip, an entire large problem can fit on this computer at one time. Thus there would be no need either to break the solution into sequential pieces or to reload the array frequently. The resulting saving in communications time and other overhead is expected to contribute substantially to the power of such a machine.

To exploit the full power of the array computer, the numerical algorithms must reflect the machine architecture. To be effective, the problem has to fit the machine. Research in parallel numerical algorithms has been pursued intensively.[9-14] However, it is not clear that this body of research has direct application to the array computer consisting of tens of thousands of microprocessors with nearest neighbor communications. At the workshop, Dr. Grosch presented his analysis of the results from a test case designed to utilize the array architecture proposed by Dr. Sutherland (see Appendix C).

In this analysis the three-dimensional Navier-Stokes equations in primitive variables and the Poisson equation for pressure had been Fourier-transformed in the span-wise direction. For the remaining two space dimensions and time variable, the equations were approximated by using second-order-accurate finite-difference schemes. The Adams-Bashforth method was used to solve the velocity equations, and several point-wise relaxation schemes were considered for solving the Poisson equation. The results of the analysis of a model problem consisting of a 100×100 array of cells and 128 Fourier components showed that the problem could be solved at a rate of about 0.35 sec per time step.

Professor Grosch concluded that velocity calculations were amenable to an array computer but that the pressure calculations were more difficult and required more than 60 percent of the computing time. This implies that computational performance improvements are being paced by improvements in the pressure calculations and that experimentation on ways to perform these calculations could lead to a substantial payoff. The major conclusion of this study is that it is possible to develop a numerical strategy for using a parallel-processor machine while accepting the constraint of local communication between nearest neighbors only.

During the second day of the workshop, the participants were divided into three groups to discuss each of the critical areas of interest: computer technology and architecture, numerical analysis, and fluid-mechanics application. The members of the working groups are listed in Appendix B, and their deliberations are summarized in the following sections of this report.

III. SUMMARY OF WORKING GROUP DELIBERATIONS

The workshop participants were selected to provide as broad a representation as possible from the relevant research areas. This provided an opportunity for an active and free exchange of ideas and experience. The first day's series of briefings on computer design and technology, numerical analysis, and fluid-mechanics applications gave each participant an appreciation for the problems outside his area of expertise and helped to provide a common basis for the discussions of the second day.

Each group was charged with the task of evaluating the status of its area of concern as that area related to the design of a special-purpose parallel-processor machine for the solution of the Navier-Stokes equations. If the status of the area was too immature for reasonable predictions, the groups were to suggest possible developmental or experimental programs needed to advance the state-of-the-art of their areas to a level required to develop such a machine. The summaries of the working-group deliberations along with their conclusions are presented in this section.

A. COMPUTER TECHNOLOGY AND ARCHITECTURE

The technology and architecture group addressed issues ranging from the feasibility of such a machine through several aspects of overall design philosophy and approach. A major conclusion of this group was that appropriate technology now exists to enable one to consider an interesting machine of the nature proposed. Important to the development of such a system would be work on algorithms, architecture, and software. It was also recommended that feasibility studies and modeling must be done before any commitment is made to large amounts of funding and manpower. These and other topics and recommendations considered important are covered in detail in the following discussion.

1. Algorithms and Architecture

a. First priority in the design of an array computer should be

given to an extensive effort aimed at algorithm development and simulation. This effort should be concurrent with, and should interact with, the architectural development. This is important since experience with developing algorithms for parallel processing is limited. While algorithms, software, and hardware can be modeled on general-purpose computers, such simulations must incorporate processing appropriate to the methods being studied, and must include communications and timing overhead. As part of the algorithm-simulation and cost trade-off analysis, small hardware prototypes and experiments should be considered. With these, hard data could be developed with which to validate the simulation and modeling--particularly in terms of data-and-control paths and their timing.

b. Above all, simplicity of interconnections must dominate architecture at all levels: within LSI, at the board level, and at the system level.

2. Reliability

a. Reliability in both hardware and software is of primary concern and must be designed in at all levels from the beginning. A major contributor to failure is memory. Memory within the array should include single-error correction and double-error detection. In addition, considerable effort should be devoted to the development of methods for error correction and detection in each computational element. While it was recommended that gradual degradation be included, it was recognized that it is not always feasible. Diagnostic hardware and software including restart and rollback procedures should be given extensive attention during system design and should be available early.

b. One major question that is yet to be answered is: What is the maximum number of processors that can be assembled without exceeding an acceptable failure rate?

3. Component Technology

At the time specifications for a processing system are frozen, only production LSI should be considered. For custom LSI, such as read-only memory systems (ROMS) or programmable read-only memory systems (PROMS),

only those design rules and processing methods then currently in use for high-volume construction of such components should be considered. Before any custom LSI components are designed, subject to the above constraints, a thorough search for available component alternatives should be made. Further, a realistic performance/cost analysis, including design, testing, and estimated production volume costs, should be performed.

4. Software

a. Major emphasis should be placed on the development of a high-level language appropriate to the specific machine design and applications.

b. Special instructions that are important to performance should be available to the user.

c. A language simulator should be available early.

5. Human Interface

a. In addition to the usual input techniques, it was felt that interactive graphical data-input techniques would be a desirable feature. However, it was strongly recommended that standard available equipment be used with a minimum of special software and special interfaces in order to keep costs down.

b. Post-processing should include not only the processing of output data but also the ability to take a snapshot of what is going on in the algorithms, the solutions, or the hardware during the computation. Limited post-processing for snapshots must run in parallel with the computations. Intermediate results must be available in as near real time as possible, again primarily in the form of snapshots. Appropriate software should be available early for obtaining intermediate results and post-processing as an aid to system debugging and application development.

c. The host system, or control unit, should be a readily available unit. The interface between the Navier-Stokes computer and the host system should use off-the-shelf software and hardware including

input/output if at all possible. The intent of this recommendation is the avoidance of unnecessary costs and development efforts.

6. Data Base Management

There was insufficient time to consider in detail the question of data-base management for a system that would be as data-intensive as the proposed system. This is viewed as an extremely important subject and must be addressed in terms of the architecture, product availability, data-and-control communications paths, cost, reliability, and modularity. It was suggested that this might be more important than what goes on within the array itself.

B. NUMERICAL SIMULATIONS

The principal conclusion of the numerical simulation working group is that adequate algorithms and sufficient numerical experience exist for the solution of the relevant partial differential equations. This suggests that it is now appropriate to initiate a feasibility study of the design of a parallel-processor machine for the solution of the problems envisioned employing appropriate numerical techniques. It is expected that, as time goes by, improvements in algorithm development will accelerate. This in no way should delay the progress of a feasibility study in which modeling and experimentation will play key roles. The working group also made specific observations and recommendations as follows.

1. Variables and Equations

a. For three-dimensional problems, the primitive variables, velocity and pressure, are believed to be computationally superior to vector vorticity and vector potential. They demand less storage (four dependent variables instead of six), boundary conditions are more easily applied, and low-speed compressibility and energy transfer are more easily added to the equations.

b. Employing the continuity and momentum equations (where the pressure gradients in the momentum equations would be adjusted to guarantee that the continuity equation is satisfied) as the governing

equations gives computational advantages over the use of the momentum and pressure (Poisson) equations. The solution of an elliptic equation for the pressure is replaced by pressure adjustments in the momentum equation. Even though the relative merits of each method are not fully understood, progress in a feasibility study will not be hindered.

2. Numerical Methods

a. The method of finite differences was favored for the numerical solution of the equations over spectral and other methods. The principal considerations that led to this conclusion were: the expectation that problems with bodies of more complicated shapes than flat plates, spheres, etc., would be more difficult to treat by spectral methods; the existence of a very large body of favorable experience and knowledge of advantages and pitfalls with finite-difference schemes compared to the other methods; the flexibility and robustness of the method; and the ease of programming. The major disadvantage is that some problems, by their nature, will require nonrectangular grids.

b. Second-order differences in space and time have advantages over higher-order differences because of lower storage and communication requirements, ease of application of boundary conditions, shorter computation time per step, and ease of programming. (It is possible that in using higher-order methods, lower-order approximations for the boundary conditions might, in some problems, propagate the same order of error throughout the solution thereby destroying whatever advantages the higher-order methods seem to have.) Thus, the group recommends second-order over higher-order differences for these problems.

c. The numerical stability advantages of implicit finite-difference schemes over explicit schemes for the solution of the Navier-Stokes equations is not as obvious for a three-dimensional simulation as it is for boundary-layer flow. Additional programming, extra arithmetic operations, logic complications, and extra storage demands may obscure anticipated benefits of a large time step, except perhaps for computations that extend to steady-state flow. Thus, a critical assessment of the relative merits of implicit vs. explicit schemes remains to be made and could be accomplished by performing appropriate numerical experiments.

Such studies can prove or disprove the conjecture that some implicit schemes should be considered as iterated explicit schemes, a point which can allow the computer hardware to be the same regardless of the nature of the scheme.

d. If the design of the computer were to adhere to a principle of nearest-neighbor communication, it is believed that iterative methods for solution of the Poisson equation and for the Navier-Stokes equations (if implicit finite-difference equations are used) would be more appropriate than direct methods to both the computation and the programming. There may be some advantage to incorporating a direct tridiagonal linear systems solver, because this might offer the possibility of using line successive overrelaxation methods or alterning direction methods. This can be investigated during an algorithm-simulation study. Recent developments in accelerating the convergence of relaxation methods for the solution of the elliptic-difference equations or implicit parabolic-difference equations by adaptively coarsening and refining the difference-equation grids suggest that provisions to incorporate such a capability into the design be seriously considered.

e. Some recent experiments comparing finite-element methods with finite-difference methods on advective/diffusive problems suggest that the issue of the selection of finite-difference methods over finite-element methods may not be closed. The finite-element methods have considerable leeway in the irregularity of the grids and incorporate boundary conditions as well. However, the finite-element experiments indicated above were not in three space dimensions and were applied to linear problems. What is clear is that there is limited experience in utilizing finite-element methods in fluid-mechanics problems.

C. FLUID-MECHANICS APPLICATION AREAS FOR A SPECIAL-PURPOSE COMPUTER

In considering the spectrum of possibilities for a special-purpose computer for application to problems in fluid mechanics, there was a general consensus in the Applications Group that the first such machine should be at the smaller end of the spectrum. It should be relatively inexpensive and relatively simple. In considering the class of problems which are "do-able" and which are of sufficient general "practical"

interest to permit development of such a machine, the group agreed that the problem of incompressible (but possibly variable-property) boundary-layer stability and transition is a good choice.

1. Stability and Transition

This is a problem which is of great practical importance in both aerodynamics and hydrodynamics, is difficult and expensive to study experimentally, and is prohibitively expensive to compute on general-purpose machines. A machine which would allow computation of laminar instabilities, including nonlinear and three-dimensional effects, and the effects of external disturbances on their origin and growth to turbulent spots would be of great practical interest. It would also be of considerable theoretical interest since such computations would improve the understanding of the physics of laminar instability and their growth.

2. Resolution and Memory

Of importance to the architecture of such a machine is the resolution required for computation of laminar instabilities. The group estimated that at least a million grid points would be required for such computations. The resolution problem could be minimized by developing a grid in which the mesh automatically becomes finer in those parts of flow where instabilities begin to develop. Also of concern to the machine architects (in the subsequent discussion) are the memory requirements for this grid size.

3. A Prototype Machine

There was general agreement that a prototype machine would be desirable. This would probably not be a large enough machine to yield any new information on fluid physics but would be very useful for improving knowledge of the interactions between algorithm and architecture--thus improving the architectural design of the ultimate machine.

4. Efficiency and Development

A desire was expressed that the machine be efficient enough to

keep down the cost of re-running problems for which more detailed output is desired. It was recommended also that an experimenter, familiar with the corresponding analog machine (i.e., reality), be associated with the development of the machine.

IV. CONCLUSIONS AND RECOMMENDATIONS

The workshop participants generally agreed that appropriate technology now exists to build a large parallel-processor array computer consisting of thousands of microprocessors capable of some direct simulations of the Navier-Stokes equations. Such a machine would have a computing capacity of at least an order of magnitude superior to current supercomputers and could be built within the next 3 to 5 years at a cost substantially less than current general-purpose supercomputers.

Adequate algorithms and sufficient numerical experience were found to exist for the solution of the relevant partial differential equations. As an example, a test case was examined consisting of an incompressible flow of a constant-density fluid in a boundary layer adjacent to a rigid, impermeable, no-slip wall. The three-dimensional Navier-Stokes equations were solved. The results of a model problem as applied to an array of 10,000 cells with 128 grid points in the cross-stream direction showed that the problem could be solved at a rate of about 0.35 sec per time step. This is about 230 times faster than a fast conventional computer (e.g., a CDC 7600).

The large array computer evaluated in the workshop was found to offer the potential for efficient direct Navier-Stokes simulations of nonlinear laminar instabilities and boundary-layer transition. A limited simulation of turbulence at low to moderate Reynolds numbers and a simulation of large-scale turbulent flows at high Reynolds numbers using sub-grid modeling also appear feasible with such a computer.

The results of this preliminary work suggest that it is now appropriate to continue to explore the feasibility of designing a parallel-processor machine for the solution of the Navier-Stokes equations using applicable numerical techniques. Important to the development of such a system is the early initiation of a coordinated program of research on algorithms, architecture, and software for this computer. We suggest that the research plan be undertaken in stages.

In the first stage, both the mathematical problems and the machine-design problems should be explored in an integrated manner in order to

confirm the hypothesis that it is reasonable to build such a machine. Machine-design problems which need to be explored include: the design of the processor itself, a determination of how actually to assemble an array of 10,000 computers, an investigation of the problem of communications between elements of the array, an analysis of questions concerning the appropriate type and balance of memory and computational capability associated with each processor, and a determination of appropriate peripheral equipment including means for input/output. None of these questions are simple, and it would be a mistake at this point to rush into the design of such a machine without preliminary investigation of them.

Algorithm development must be concurrent with and must interact with the architectural development. The challenge in applying the parallel-processor concept is to couch the problem to be solved in a form which does not require each processor to communicate with more than a few of its nearest neighbors, since both many communication paths and long communication paths make the computer architecture fundamentally expensive. The extent to which one is able to develop algorithms that map the geometry of the problem (including boundary conditions) onto the geometry of the computer will determine the magnitude of the computational gains achieved as promised by the parallel-processing architecture. This requires a close working relationship between the computer architects and numerical analysts. Algorithms, software, and hardware should be modeled using available general-purpose computers. These simulations must perform the processing in a manner that accurately models the type of architecture proposed for the actual machine.

This first stage should include an analysis of the range of problems that can be solved using the array architecture being considered. This analysis should use a realistic estimate of the computational power of the proposed machine based on the performance characteristics of hardware as it exists or will exist one to two years in the future. Extrapolation five to ten years into the future must be avoided if the analysis is to remain credible.

If, after completion of this stage, it still seems reasonable to proceed, the second stage should consist of prototype hardware development. The first phase of this development should be the design and construction or simulation of the actual processing elements to be used at each point in the array. Experimentation with these elements, to explore their properties in some detail, should be performed before determination of the final design of the elements that will compose the final array processor.

Next, we recommend the construction of a small experimental array of processors (perhaps 10×10 or 20×20) to test the concept. The individual processors should be the type proposed for the large-scale machine but need not be produced to the final physical dimensions. In particular, small packaging and other problems that have to be addressed only for the final machine could be avoided in the initial exploration.

This small experimental array will be invaluable because it will permit the acquisition of hard data on the suitability, cost, and reliability of current technology when used in the development of this concept. The results of simulations and modeling of algorithms can also be tested against reality. Finally, solutions developed during earlier phases of the study for problems associated with hardware/software diagnostics, input/output, interfacing, data management, etc., can be tested on real hardware at low cost.

The completion of these two design stages will provide the information necessary to decide whether it is appropriate to enter a detailed design, development, and construction phase for a full-scale array processor machine to solve the Navier-Stokes equations.

Appendix A

I. A STRAW-MAN COMPUTER FOR NAVIER-STOKES SIMULATION--A PAPER PRESENTED AT THE NAVIER-STOKES WORKSHOP HELD AT RAND 9-10 MARCH 1977. IVAN E. SUTHERLAND

II. A VARIANT ON IVAN SUTHERLAND'S PROPOSAL FOR A NAVIER-STOKES COMPUTER. R. STOCKTON GAINES

I. A STRAW-MAN COMPUTER FOR NAVIER-STOKES SIMULATION--
A PAPER PRESENTED AT THE NAVIER-STOKES WORKSHOP
HELD AT RAND 9-10 MARCH 1977

Ivan E. Sutherland
California Institute of Technology

A STRAW-MAN COMPUTER

This note describes a specific computational system believed to be relevant to the Navier-Stokes problem. This straw-man design was presented at the opening of the conference to serve as a specific point of discussion during working group deliberations. A straw man provides us a machine of sufficient complexity to help us determine whether the computations that can be handled by such a machine or a scaled-up version are of interest from an applications point of view.

While the straw-man design may not be correct in detail, it nevertheless presents the appropriate constraints on the design of specialized computing machinery that must be considered in problem formulation and algorithm development. The understanding of the design limitations in such a machine and the implications of such limitations on the forms of computations will form an important part of any further consideration of such machines.

The straw-man design consists of 10,000 identical integrated circuits--each mounted in a 16-pin integrated circuit package. These 10,000 circuits are arranged in a rectangular array 100 × 100 occupying a space approximately 8 feet square. One can imagine the entire computer mounted on a wall. Each integrated circuit in the "straw-man" design is capable of storing about 16,000 bits of information arranged as 256 words of 64 bits each. In addition, each of the integrated circuits is capable of performing a 64-bit fixed-point addition in about 100 nanoseconds. Each of these computational elements receives identical instructions from a common central source.

In order to reduce the wiring congestion of the machine, each of these elements communicates only with its nearest neighbors. Two alternative communication schemes are possible as shown in Figs. 1 and 2.

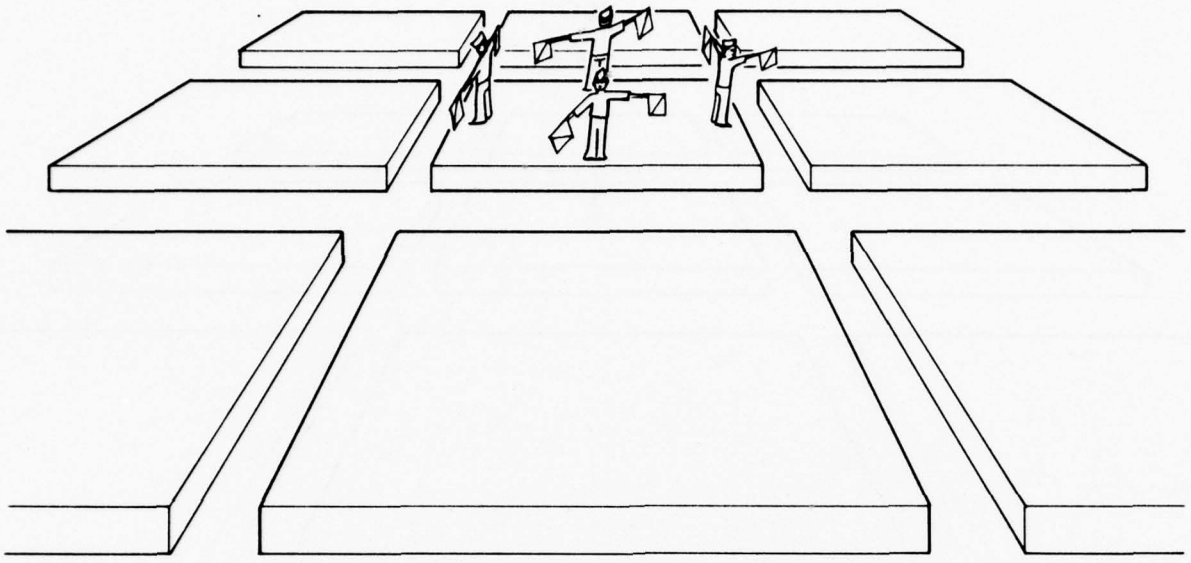


Fig. 1a — Schematic representation of communication between elements (individual outgoing or incoming messages)

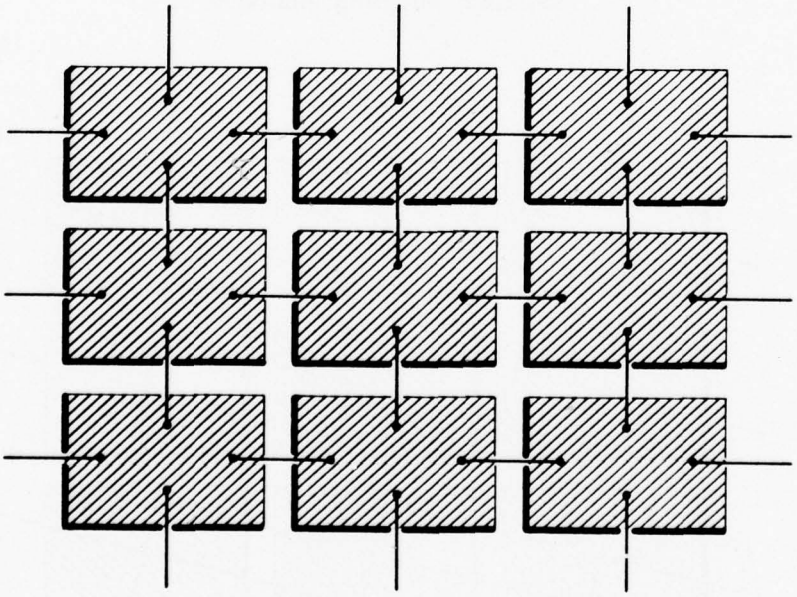


Fig. 1b — Wiring schematic of the large-array computer (individual outgoing or incoming messages)

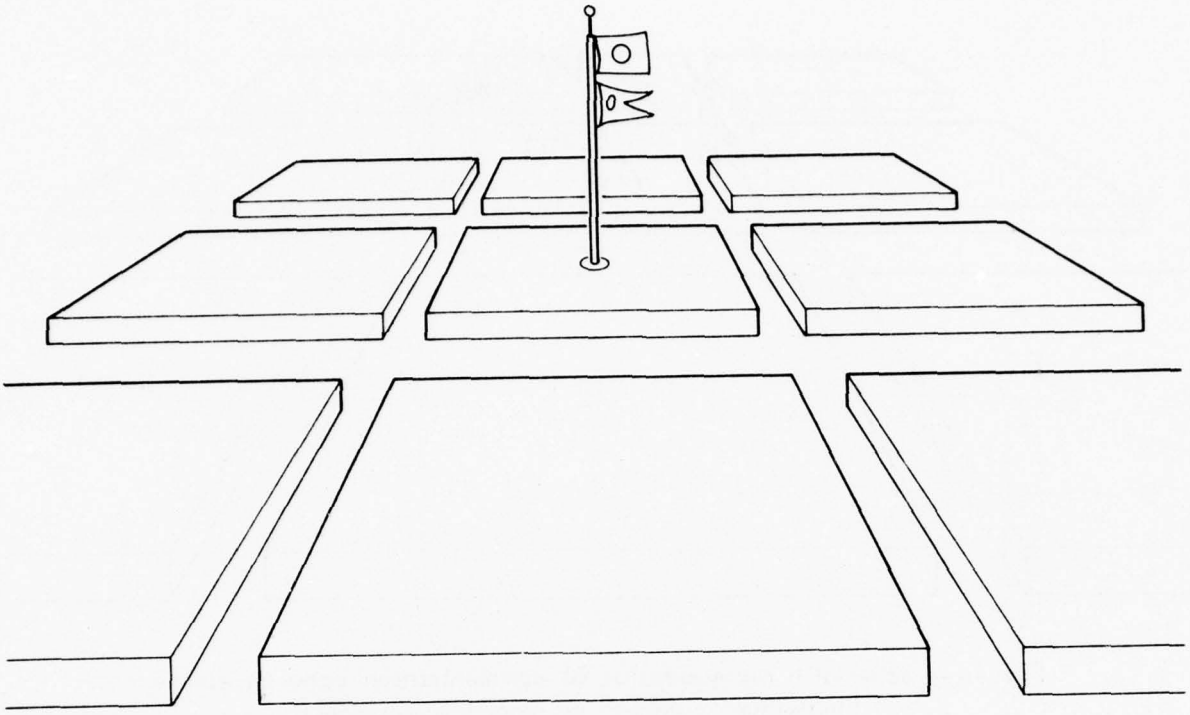


Fig. 2a — Schematic representation of communication between elements (identical outgoing messages)

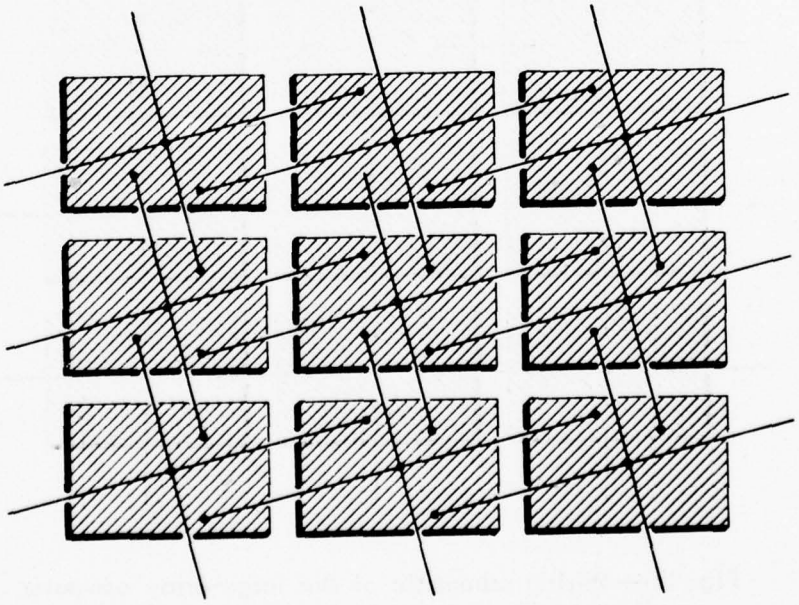


Fig. 2b — Wiring schematic for a large-array computer (identical outgoing messages)

In the simpler communication scheme, schematically illustrated in Fig. 1a, a one-way reversible communication path exists between each adjacent pair of computing elements. This communication path is, in fact, a single wire between the two elements as shown in Fig. 1b. In this scheme, each computing element may communicate one bit at a time to or from an adjacent neighbor but cannot send and receive simultaneously. This scheme permits a computing element to send up to four messages simultaneously, each of which may be different. An alternative scheme shown in Fig. 2 has each element transmitting an identical message to its four immediate neighbors and is able simultaneously to receive a message from each of them. This scheme requires 5 pins on the integrated circuit package but provides for twice as much communication so long as the problem can make use of identical messages transmitted in all four directions.

In addition to this simple local communication, communication lines from a central source to the 10,000 processors are provided. One hundred individual row wires and 100 individual column wires are driven by the central control system which enable it to designate any individual processor for special attention. The row and column wires account for 2 additional pins on the package. The remaining 9 wires on the integrated-circuit package are used to obtain power and ground and to communicate 7 command signals with the central machine. These 7 command signals go in parallel into all 10,000 processors and instruct them as to what step to do next.

Notice that the straw-man machine design does not have long communication paths within the array. It is not possible for an element to talk to another element 8 or 10 positions away in the array without transmitting the message through intermediate computing elements. This feature of the design is important because it minimizes the number of wires required to interconnect with the processing elements and, thus, minimizes the cost of interconnection. The communication costs between the processing elements could easily grow to be a major factor were long-distance communication made available.

Three-dimensional problems are handled in the straw-man machine by using the storage available in any processor to represent a set of data

elements from sample points that lie in a column in the volume of space in which computation takes place. The arithmetic units in the processing elements will process various planes of the solution in sequence using data from different places in the column. Thus, each processor, in effect, marches up and down its column, doing its computations at the same time its neighbors are marching up and down their columns performing similar computations there. In part, the size of the problem which can be attacked is limited by the number of storage elements available and, thus, the number of elements in the column which can be adequately represented.

Input/output to the array can be accomplished through appropriate use of row and column wires. Individual messages to individual processors may also be passed into the array by designating the recipient with the row and column wires and sending the appropriate message along the many available broadcast wires. All processors will receive the message, but only the designated one will act on it.

Output from the array processors might be greatly facilitated by including a light emitting diode on each of the integrated circuit packages. Thus, each of the 10,000 integrated circuits could light up under suitable conditions. For example, one could have any processor where the pressure exceeded a certain threshold light up and, thus, watch the play of high-pressure regions as the problem solution proceeds. Since the output of many computations suitable to array processing is appropriately represented in pictorial form, the notion of generating the pictures *in situ* and using optical communication photography as the direct output medium makes good sense.

The challenge in using this straw man to solve problems is to couch the problem in terms which do not require long-distance communication since long-distance communication is fundamentally expensive from the computer architecture point of view. One hopes that the solution technique for solving the Navier-Stokes problem, by its very nature, can easily be couched in such terms that long-distance communication is not necessary. The flexibility of communication available in a random-access memory has lowered the sensitivity of most computer people to the very real cost of communication. Our major exercise during this conference is

to examine the match between the chosen problem area, the Navier-Stokes equations, and the hypothetical machine architecture, considering principally the communication limitations which will exist. The extent to which we are successful in mapping the geometry of the problem area onto the geometry of the wiring provided in the computer is the extent to which we will be successful in obtaining the computational gains promised by the "army of ants" architecture. Should we fail to map the problem domain onto the specific configuration of the machine, we will fail to reap its benefits.

The potential benefits of the architecture described are enormous. Although each processing element is capable of performing a 64-bit fixed-point addition in 100 nanoseconds and, thus, will require 6.4 microseconds to perform a multiplication, the array of 10,000 such elements performs at a rate of approximately 1.5×10^9 multiplications per second, a very impressive computational power indeed. If we assume for the moment that these simple integrated circuits can be built for $10 per chip, we are talking about a spectacular cost/performance ratio and herein lies the appeal of the architecture. The challenge, of course, is to get the individual computing elements to work successfully together.

THE STRAW MAN REVISITED

During the two-day conference, our discussions of problem areas and technological feasibility led me to a modification of the straw-man architecture. I am convinced, first of all, that the amount of storage available at each node must be maximized since storage is the major limitation on the size of the problem which can be attacked. Each processing element, I believe, should have as many bits of memory as possible even if those bits of storage are somewhat inconvenient to use. For example, one might use a serial storage device such as a charge-coupled device (CCD) memory. In addition to a large serial store of perhaps one or two hundred thousand bits, each processing element should have random access memory of perhaps as few as 32 64-bit words. The arithmetic capability of a single adder per processing element seems appropriate to our present purpose since to incorporate additional

arithmetic elements will increase the design cost substantially and not change the performance characteristics very much.

Each processor should include some space for storing algorithms and data. This feature is important because it permits individual processors to be distinguished from each other not only in their position in the array and the data which they contain, but also in terms of the specific functions they perform. These distinctions between processors will give the system an enormous flexibility, flexibility well worth the cost in storage and logical complexity required to implement the functions. If each machine is, in effect, a stored program microcomputer, the variety of functions available to the array is greatly increased, since the processors need not function identically.

II. A VARIANT ON IVAN SUTHERLAND'S PROPOSAL FOR A NAVIER-STOKES COMPUTER

R. Stockton Gaines
The Rand Corporation

I would like to suggest a variation on the proposal made by Ivan Sutherland. I would then argue that on the basis of this suggestion and some of the ideas presented in the talks at the workshop, one can accept the hypotheses that a special-purpose computing machine with a computing capacity of at least an order of magnitude superior to current supercomputers can be built at a cost substantially less than current supercomputers and that such a machine can be considered in the next 3 to 5 years.

The Sutherland design proposed a 100 by 100 array of small processors manufactured on individual chips. The power of each of these 10,000 processors would essentially be that of a current 16 K memory chip with a little bit of additional logic to allow an adder and a one-stage shifter to be incorporated on the chip. Ivan proposed that such a chip could be produced within the next few years at a cost of about $10 a chip, which seems to be a reasonable prediction given the current state of integrated-circuit technology. This array would be controlled in a broadcast mode so that all processors in the array would either carry out no activity or the same activity at any individual time step in the process of calculating a result. Each chip would have the ability to communicate the contents of any portion of its memory to its adjacent neighbors in the array.

I wish to suggest that there should be substantially more computing power in each node of the array. The presentation by INTEL at the workshop offered the hypothesis that within five years INTEL will be able to manufacture a single chip with computing power approximately equivalent to a PDP-11/70 for a cost of about $100 per chip. On the basis of the details of the INTEL presentation and other discussions during the workshop, I believe it is reasonable to suggest that the processing power at each node of the array be substantially more powerful than the rather

simple computing capacity of the Sutherland proposal. In particular, I suggest that we consider that the computing power at each node of the array be equivalent to a small computer that can: (a) perform the small set of operations needed to compute finite-difference approximations, (b) store the amount of information necessary for computing tasks at one point in the array, and (c) communicate with its neighbors as well as in broadcast mode to the entire array or to the controller at the edge of the array. I have been informed that it would be reasonable to put together a small processor on a board consisting of a few chips made out of current technology which might have this power already for a cost of approximately $500 in today's market. Such a processor would have one chip for communications, one chip for processing and several chips for memory. The amount of memory that would be necessary would be enough to store several variables for each point in the third dimension of a three-dimensional space. The amount of memory to be stored on each chip would be about the same as in the Sutherland proposal; the main difference being proposed here is the amount of computing power that should be available at each point in the array.

There should be storage at each point in the array for a small number of program steps, say 100, which would be roughly equivalent to the inner DO-loop of a finite-difference approximation program. Since the computer at each point would not need to store many instructions or many data (compared with general-purpose computers), the space necessary to store a program would be quite small. The advantage of storing the program in each chip is that different chips in the array can be doing different activities at the same time. Since an actual problem involves computations on the fluid for which the finite-difference approximation holds, while at the same time carrying out special computations at all boundaries (external and internal), external control will mean that for the greater portion of the computing time most of the elements of the array will be shut off while special boundary conditions are being computed in part of the array. The effective computing capacity of the array as a whole could be reduced by a factor of 6 or more for many problems because of this.

It seems likely that a special-purpose computer unit, designed to hold a small program and carefully tailored to performing just those

computational activities appropriate to finite-difference approximations, would not be too difficult to design given today's technology, and could be produced in quantity at costs comparable to those we were talking about for other chips.

The presentation by INTEL lends credence to the likelihood that powerful computing chips will be available at reasonable costs. In addition, I have discovered that there is already a chip available for about $100 that contains a NOVA computer on it. Other discussions during the workshop also suggest that chips such as those discussed are feasible to produce.

Appendix B

WORKSHOP AGENDA, LIST OF PARTICIPANTS, AND ORGANIZATION

AGENDA

NAVIER-STOKES COMPUTER WORKSHOP

The Rand Corporation

Santa Monica, California

March 9-10, 1977

Wednesday, March 9, 1977

9:00	Introduction to Rand	D. Rice, E. C. Gritton
	Background, Problem Definition, and Summary of Computational Requirements	S. J. Lukasik
9:45-12:00	COMPUTER TECHNOLOGY (I. Sutherland)	
	State-of-Art and Technology Projections	W. Pohlman, INTEL
	Computer Architecture and Multi-Processor Design	I. Sutherland, Cal Tech
	Summary of Experience with ILLIAC	D. Stevenson, NASA/IAC
	Summary of Experience with STAR	P. J. Bobbitt, NASA/ Langley
	Discussion	
12:00-1:00	Lunch	
1:00-3:00	NUMERICAL SIMULATIONS (W. S. King)	
	Numerical Algorithms and Solution Techniques	W. S. King, M. Juncosa, Rand
	Navier-Stokes Simulator	C. E. Grosch, Old Dominion
	Large Eddy Turbulence Simulation	J. Ferziger, Stanford
3:00-5:00	FLUID MECHANICS APPLICATIONS (C. Gazley, Jr.)	
	Applications and Problem Areas	C. Gazley, Jr., Rand
	Application to Experimental Facilities	D. E. Coles, Cal Tech
	Computational Aerodynamic Design Facility	F. R. Bailey, NASA/ Ames
	Discussion	

5:30	Social Hour	

Thursday, March 10, 1977

8:30-9:00	Introduction to Working Group Session (Main Conference Room)	
9:00-12:00	Working Group Meetings:	
	--Computer Technology	R. Rice
	--Application Areas	C. Gazley, Jr.
	--Numerical Simulations	W. S. King
12:00-1:00	Lunch	
1:00-5:00	Report of Working Groups (E. C. Gritton, Chairman) Discussion and Final Recommendations	

NAVIER-STOKES COMPUTER WORKSHOP

March 9-10, 1977

PARTICIPANTS

Dr. F. Ronald Bailey*
Thermo & Gas Dynamics Division
Mail Stop 229-3
Ames Aeronautical Laboratory
Moffett Field, California 94035
(415) 965-6419

Mr. Percy J. (Bud) Bobbitt
Head, Theoretical Aerodynamics Branch
STAD
NASA Langley Research Center
Mail Stop 360
Hampton, Virginia 23665
(804) 827-1110

Professor Donald E. Coles*
Aeronautics Department, 205-50
California Institute of Technology
Pasadena, California 91125
(213) 795-6811

Dr. George L. Donohue
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209
(202) 694-1903

Professor Joel Ferziger*
Department of Mechanical Engineering
Stanford University
Stanford, California 94305
(415) 497-3615

Dr. S. Fernbach
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, California 94550
(415) 447-1100, X3767

Dr. R. Stockton Gaines
Information Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

Dr. Carl Gazley, Jr.*
Physical Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

Dr. Eugene C. Gritton*
Physical Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

Professor Chester E. Grosch*
Institute of Oceanography
Old Dominion University
Norfolk, Virginia 23508
(804) 489-6477

Dr. C. W. Hirt
Los Alamos Scientific Laboratory
Mail Stop 216
P. O. Box 1663
Los Alamos, New Mexico 87445
(505) 667-4156

Dr. Mario L. Juncosa
Physical Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

*Speaker.

Dr. Michael Kascic
STAR Operations Division
Control Data Corporation
4290 Fernwood Avenue
Arden Hills, Minnesota 55112

(612) 482-2100

Professor H. Keller
Applied Mathematics Department,
101-50
California Institute of Technology
Pasadena, California 91125

(213) 795-6811

Dr. William S. King*
Physical Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

(213) 393-0411

Dr. Richard Lau
Office of Naval Research
1030 East Green Street
Pasadena, California 91106

(213) 681-5264

Dr. Stephen J. Lukasik*
S. J. Lukasik, Ltd.
8400 Westpark Drive
McLean, Virginia 22101

(703) 356-4490

Dr. David D. Loendorf
Institute for Computer Applications
in Science & Engineering
NASA Langley Research Center
Mail Stop 246
Hampton, Virginia 23665

Mr. Marshall Pease
Stanford Research Institute
333 Ravenswood
Menlo Park, California 94025

(415) 326-6200, X4123

Dr. Harold Petersen
Physical Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

(213) 393-0411

Mr. William Pohlman*
INTEL
3065 Bowers
Santa Clara, California 95051

(408) 246-7501

Mr. Rex Rice
Fairchild Memory Systems
1725 Technology Drive
San Jose, California 95110

(408) 998-0123, X474

Professor Philip Saffman
Applied Mathematics Department, 101-50
California Institute of Technology
Pasadena, California 91125

(213) 795-6811

Dr. John Steinhoff
Research Department
Plant 35
Grumman Aerospace Corporation
Bethpage, New York 11714

(516) 575-0574

Dr. David Stevenson*
Institute of Advanced Computation
1095 East Duane Avenue
Sunnyvale, California 94086

(408) 735-0635

Mr. Jon Surprise
Department 470-G3
Goodyear Aerospace Corporation
Akron, Ohio 44315

(216) 794-2804

Dr. Ivan Sutherland*
Computer Science Department, 256-80
California Institute of Technology
Pasadena, California 91125

(213) 795-6811

Mr. Charles Thacker
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94301

(415) 494-4000

WORKSHOP ORGANIZATION

CHAIRMAN: DR. E. C. GRITTON (RAND)

COMPUTER TECHNOLOGY AND ARCHITECTURE

Session Chairman:
Mr. Rex Rice (Fairchild)

Dr. S. Fernbach (LLL)

Prof. Joel Ferziger (Stanford)

Dr. R. Stockton Gaines (RAND)

Dr. Michael Kascic (CDC)

Mr. Marshall Pease (SRI)

Dr. Harold Petersen (RAND)

Dr. John Steinhoff (Grumman)

Dr. David Stevenson (NASA/IAC)

Mr. Jon Surprise (Goodyear)

Dr. Ivan Sutherland (Cal Tech)

Mr. Charles Thacker (Xerox)

FLUID MECHANICS APPLICATION

Session Chairman:
Dr. Carl Gazley, Jr. (RAND)

Dr. F. Ronald Bailey (NASA/AMES)

Mr. P. J. Bobbitt (NASA/LANGLEY)

Prof. D. E. Coles (Cal Tech)

Dr. George L. Donohue (ARPA)

NUMERICAL SIMULATIONS

Session Chairman:
Dr. William S. King (RAND)

Prof. Chester E. Grosch
(Old Dominion University)

Dr. C. W. Hirt (Los Alamos)

Dr. Mario Juncosa (RAND)

Dr. Richard Lau (ONR)

Appendix C

SOLUTION OF THE NAVIER-STOKES EQUATIONS IN A CELL COMPUTER: A TEST CASE

Chester E. Grosch

Old Dominion University

A paper presented at the Navier-Stokes Workshop held at Rand 9-10 March 1977.

SOLUTION OF THE NAVIER-STOKES EQUATIONS IN A CELL COMPUTER: A TEST CASE

Chester E. Grosch
Old Dominion University

C-1. INTRODUCTION

Theoretical fluid dynamicists have looked to ever larger and faster general-purpose computers to solve the Navier-Stokes equations using numerical methods. Increasing size, i.e., memory, permits a finer spatial resolution for a fixed volume of fluid and/or the calculation of flows in larger volumes. The number of operations (additions, multiplications, memory fetches, etc.) per time step increases at a slightly faster rate than the number of mesh points (or modes, for spectral methods). In fact, it is easily shown that if N^3 mesh points are used in a calculation the number of operations per time step is proportional to $N^3 \ell n_2 N$ for the best methods.

While there have been some improvements in algorithms, progress has been tied to the development of ever faster general-purpose computers. The most recent "supercomputers" incorporate substantial amounts of pipelining and parallelism in their computer processing units (CPUs) in addition to using faster processing elements. In short, high-speed computation has been sought via faster and more complex CPUs. This is true even for ILLIAC IV. Each of the 64 processing elements (PEs) is a quite powerful general-purpose computer.

It is, of course, true that any computer which is equivalent to a Turing machine is a general-purpose computer. However, it is equally true that a given computer architecture will handle certain classes of algorithms better than others. It appears that all of today's "supercomputers," including ILLIAC IV, were designed to handle many different classes of scientific computing problems. The cost of this versatility appears to be that these architectures are not nearly optimum for any one class of problems.

Dr. Sutherland has suggested an architecture which appears to be much closer to optimum, for computational fluid dynamics, than that of

existing large computers. This architecture is a two-dimensional array of cells. Each cell contains a simple, and rather slow, arithmetic unit and a modest amount of memory. The potential advantages of this architecture are threefold: first, it appears possible to build such a computer using existing technology (each cell is only a few chips, perhaps a single chip) at fairly modest cost, particularly if the intercellular connections are minimized; second, technological developments in the semiconductor industry are leading toward increasing complexity and density per chip *and* lower cost per chip; and third, if the array of cells can be mapped onto the fluid domain so that N^2 operations can be performed in parallel on N^2 chips, the number of sequential operations per time step can be reduced from $O(N^3 \ell n_2 N)$ to $O(N \ell n_2 N)$.

This architecture appears to be quite promising, at least for the finite-difference methods commonly used in computational fluid dynamics. The flow field is represented by a three-dimensional array of fluid cells so that each cell in the planar array of computer cells could represent one or more of the fluid cells. The common algorithms to compute the flow field are largely parallel in that the operations performed at any one point in a space to compute the velocity field are the same operations performed at nearly all other points in the space.

There are, of course, certain problems. Fluid cells on the boundaries of the computational region are inherently different from fluid cells in the interior (hence the "nearly all" used above). The pressure field, for incompressible flow, is global, i.e., the pressure at any point depends on the velocity field everywhere.

Nevertheless, this architecture appears so promising that it is worthwhile to examine a test case. The test case considers the incompressible flow of a fluid in a boundary layer adjacent to a rigid, impermeable, no-slip wall.

The objectives of this exercise are to determine:

1. How well a standard algorithm fits a cell computer.
2. What modifications to the algorithm are required.
3. Which part, if any, of the algorithm dominates the calculation.

4. Which operation (transfer, addition, etc.) dominates the calculation.
5. How many operations are required per time step.
6. What is the memory requirement per cell.
7. What is the computation time per time step, using conservative estimates of the transfer, multiply, and add times.

C-2. EQUATIONS OF MOTION AND BOUNDARY CONDITIONS

The geometry of the flow to be calculated is shown in Fig. 3. The computational region is $0 \leq x \leq x_o$, $0 \leq y \leq \infty$, $0 \leq z \leq z_o$. The

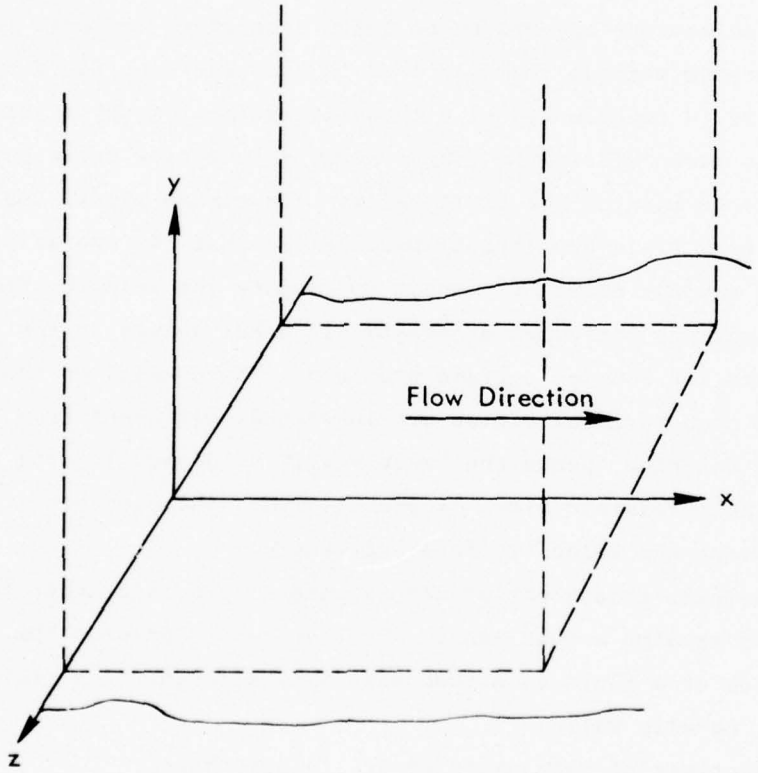


Fig. 3 — Flow geometry

equations of motion are, of course, the Navier-Stokes equations for an incompressible fluid

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\nabla p + \frac{1}{R} \nabla^2 \vec{u} \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

Here $R = U_o \delta/\nu$ is the Reynolds number, where U_o is a characteristic free-stream speed, δ is the boundary-layer thickness, and ν is the kinematic viscosity. The velocity

$$\vec{u} = \hat{i}u + \hat{j}v + \hat{k}w \tag{3}$$

has components (u,v,w) in the x, y, and z directions ($\hat{i}$, $\hat{j}$, and $\hat{k}$ are unit vectors), and p is the pressure per unit density.

A Poisson equation for the pressure can be obtained by taking the divergence of Eq. (1)

$$\nabla^2 p = Q \tag{4}$$

with

$$Q = E - S \tag{5}$$

$$E = \frac{1}{R} \nabla^2 D - \frac{\partial D}{\partial t} \tag{6}$$

$$S = \nabla \cdot [(\vec{u} \cdot \nabla)\vec{u} \tag{7}$$

$$D = \nabla \cdot \vec{u} \tag{8}$$

The divergence, D, must be zero from Eq. (2), but E is retained on the right-hand side of the Poisson equation in order to correct, at each time step, for roundoff and truncation errors which cause $D \neq 0$. This is not an artificial viscosity term because the object is to force D, and hence E, to be zero at each time step. In a sense the Poisson equation with E plays the role of the corrector in a predictor/corrector scheme.

The boundary conditions are:

$$\vec{u} \quad \text{and} \quad \frac{\partial p}{\partial x} \quad \text{given at} \quad x = 0 \tag{9}$$

$$\vec{u} = 0 \quad \text{at} \quad y = 0 \tag{10a}$$

$$\frac{\partial p}{\partial y} = \nu \frac{\partial^2 v}{\partial y^2} \quad \text{at} \quad y = 0 \tag{10b}$$

$$u \to U(x,z) \quad \text{as} \quad y \to \infty \tag{11a}$$

$$v \to V(x,z) \quad \text{as} \quad y \to \infty \tag{11b}$$

$$w \to 0 \quad \text{as} \quad y \to \infty \tag{11c}$$

$$\frac{\partial p}{\partial y} \to 0 \quad \text{as} \quad y \to \infty \tag{11d}$$

$$\vec{u} \text{ and } p \text{ periodic in } z \tag{12}$$

$$\frac{\partial^2 \vec{u}}{\partial x^2} = 0 \quad \text{at} \quad x = x_o \tag{13a}$$

$$\frac{\partial p}{\partial x} = \nu \left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) - \frac{\partial u}{\partial t} - \left(\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} \right) \quad \text{at} \quad x = x_o \tag{13b}$$

Here U and V are the free-stream velocity components and are O(1) because of the scaling with U_o. The periodic boundary conditions in z are a convenience, not a necessity. Either flow-through, free-slip conditions or solid-wall conditions could have been used for the z boundary conditions. In either case the final results (number of operations per time step) would have been changed by less than ten per cent.

The "correct" outflow boundary conditions are not known, but it is known that "reasonable" outflow conditions, such as that used here at $x = x_o$, cause a boundary layer to form at the rear of the computational region. Its thickness, in the upstream direction, is $O(\delta)$.

In order to represent the infinite physical domain $0 \leq y < \infty$ in the finite computational domain a mapping is used

$$\xi = y/(y + \ell) \tag{14}$$

The value chosen for the scale factor, ℓ, determines the resolution in the boundary layer, because (in the scaled variables) $y = 1$ is the top of the boundary layer which, in the mapped variable, is at

$$\xi = 1/(1 + \ell)$$

This maps the infinite domain ($0 \leq y < \infty$) onto the finite domain ($0 \leq \xi < 1$). With this mapping the derivatives have a simple form

$$\frac{\partial}{\partial y} = \frac{(1 - \xi)^2}{\ell} \frac{\partial}{\partial \xi} \tag{15a}$$

$$\frac{\partial^2}{\partial y^2} = \frac{(1 - \xi)^2}{\ell^2} \left[-2(1 - \xi) \frac{\partial}{\partial \xi} + (1 - \xi)^2 \frac{\partial^2}{\partial \xi^2} \right] \tag{15b}$$

It has been shown[15] that this mapping yields highly accurate results with relatively few grid points in those cases, as in this problem, where the flow field at infinity is a simple laminar flow. The only cost is that the metric coefficients must be stored in each cell.

C-3. DIFFERENCING METHODS

The physical space, $0 \leq x \leq x_o$, $0 \leq \xi \leq 1$, $0 \leq z \leq z_o$, is divided into $L \cdot M \cdot N$ cells centered on the points

$$(x_i, \xi_j, z_k) = (i\Delta x,\ j\Delta\xi,\ k\Delta z) \tag{16}$$

$i = 1, 2, 3, \ldots, L$

$j = 1, 2, 3, \ldots, M$

$k = 1, 2, 3, \ldots, N$

A typical fluid cell is shown in Fig. 4. The x component of the velocity is defined at the center of the front and back faces of the cell, i.e.,

$$u_{i-\frac{1}{2},j,k} \equiv u(i - \tfrac{1}{2})\Delta x,\ j\Delta\xi,\ k\Delta z,\ t) \tag{17}$$

Similarly the ξ and z components of the velocity, v and w are defined at the centers of top and bottom and side faces of the cell. The pressure is defined at point (i,j,k) in the center of the cell.

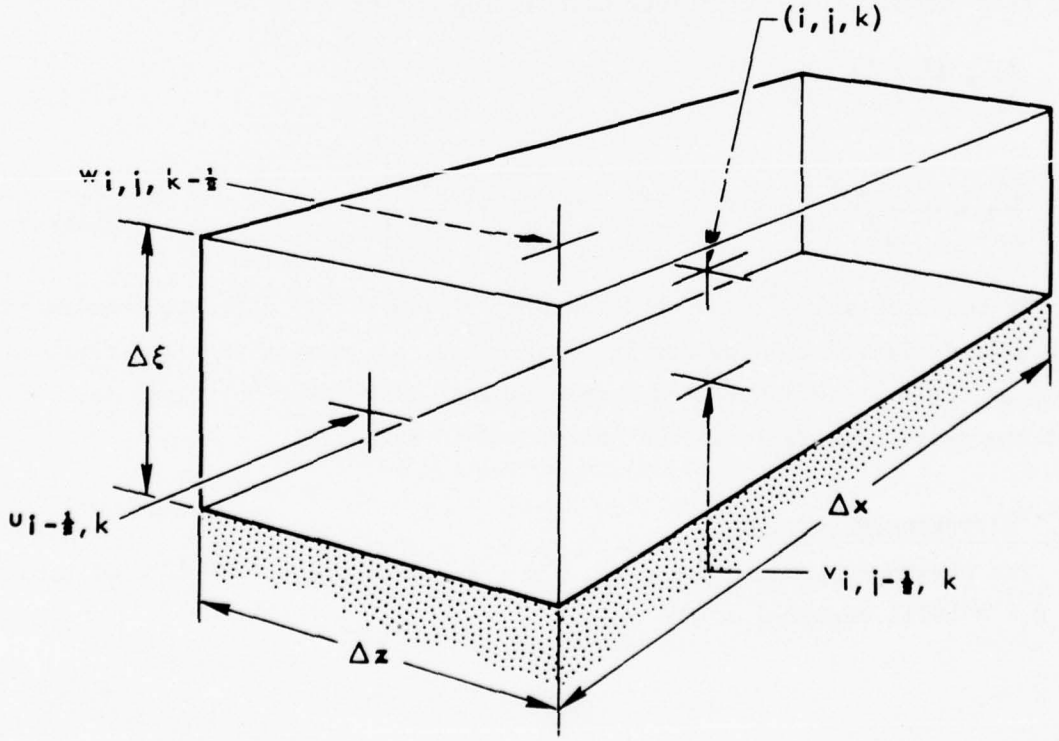


Fig. 4— Typical computation cell

The spatial differencing scheme is of the centered second-order type. As an example, consider the u-component of Eq. (1) written in conservation form

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{(1 - \xi^2)}{\ell} \frac{\partial uv}{\partial \xi} + \frac{\partial uw}{\partial z} + \frac{\partial p}{\partial x}$$

$$= \frac{1}{R} \left\{ \frac{\partial^2 u}{\partial x^2} + \frac{(1 - \xi)^2}{\ell^2} \frac{\partial}{\partial \xi} \left[(1 - \xi)^2 \frac{\partial u}{\partial \xi} \right] + \frac{\partial^2 u}{\partial z^2} \right\} \tag{18}$$

The semi-discrete approximation for $u_{i-\frac{1}{2},j,k}$ is

$$\frac{\partial u_{i-\frac{1}{2},j,k}}{\partial t} = \left(\frac{1}{\Delta x}\right) \Big\{ -(u^2_{i,j,k} - u^2_{i-1,j,k})$$

$$- a_j (u_{i-\frac{1}{2},j+\frac{1}{2},k} v_{i-\frac{1}{2},j+\frac{1}{2},k} - u_{i-\frac{1}{2},j-\frac{1}{2},k} v_{i-\frac{1}{2},j-\frac{1}{2},k})$$

$$- \left(\frac{\Delta x}{\Delta z}\right) (u_{i-\frac{1}{2},j,k+\frac{1}{2}} w_{i-\frac{1}{2},j,k+\frac{1}{2}} - u_{i-\frac{1}{2},j,k-\frac{1}{2}} w_{i-\frac{1}{2},j,k-\frac{1}{2}})$$

$$- (p_{i,j,k} - p_{i-1,j,k})$$

$$+ \frac{1}{(\Delta x)R} \Big[u_{i+\frac{1}{2},j,k} - 2u_{i-\frac{1}{2},j,k} + u_{i-3/2,j,k}$$

$$+ b_j (u_{i-\frac{1}{2},j+1,k} - 2u_{i-\frac{1}{2},j,k} + u_{i-\frac{1}{2},j-1,k})$$

$$- c_j (u_{i-\frac{1}{2},j+1,k} - u_{i-\frac{1}{2},j-1,k})$$

$$+ \left(\frac{\Delta x}{\Delta z}\right)^2 (u_{i-\frac{1}{2},j,k+1} - 2u_{i-\frac{1}{2},j,k} + u_{i-\frac{1}{2},j-1,k}) \Big] \Big\} \tag{19a}$$

where

$$a_j = \left(\frac{\Delta x}{\Delta \xi}\right) \left[(1 - \xi)^2 / \ell \right]_j \tag{19b}$$

$$b_j = \left(\frac{\Delta x}{\Delta \xi}\right)^2 \left[(1 - \xi)^2 \ell^2 \right]_j \tag{19c}$$

$$c_j = \left(\frac{\Delta x}{\Delta \xi}\right)^2 \left[(1 - \xi)^3 \, \Delta \xi / \ell^2 \right]_j \tag{19d}$$

and the subscript on the a_j, b_j, and c_j means that these functions are evaluated at $\xi = j \cdot \Delta\xi$. Note that certain of these variables, such as $u_{i,j,k}$, are not directly available. They are defined as the average of the variable at adjacent grid points,

$$u_{i,j,k} \equiv \frac{1}{2}(u_{i+\frac{1}{2},j,k} + u_{i-\frac{1}{2},j,k}) \tag{20}$$

It can be shown that this approximation, when applied to Eqs. (1) through (8) with boundary conditions (9) through (13), is conservative in the sense that mass is conserved for any R; and, in the limit $R \to \infty$, momentum, energy, and enstrophy, are also conserved.

The time differencing is of the Adams-Bashforth type. Let us define

$$F^n_{i-\frac{1}{2},j,k} \equiv \left\{ -\frac{\partial u^2}{\partial x} - \frac{(1-\xi)^2}{\ell}\frac{\partial uv}{\partial \xi} - \frac{\partial uw}{\partial z} - \frac{\partial p}{\partial x} + \frac{1}{R}\left[\frac{\partial^2 u}{\partial x^2} + \frac{(1-\xi)^2}{\ell^2}\frac{\partial}{\partial \xi}\left[(1-\xi)^2\frac{\partial u}{\partial \xi}\right] + \frac{\partial^2 u}{\partial z^2}\right]\right\}^n_{i-\frac{1}{2},j,k} \tag{21}$$

the righthand side of (19) defined at (subscripts), the point $(i\Delta x, j\Delta\xi, k\Delta z)$ in space and at (superscript) time $n\Delta t$. $F_{i,j-\frac{1}{2},k}$ and $F_{i,j,k-\frac{1}{2}}$ are defined in a completely analogous way. Then the Adams-Bashforth approximation to the time derivative is

$$u^{n+1}_{i-\frac{1}{2},j,k} = u^n_{i-\frac{1}{2},j,k} + \frac{\Delta t}{2}(3F^n_{i-\frac{1}{2},j,k} - F^{n-1}_{i-\frac{1}{2},j,k}) \tag{22a}$$

$$v^{n+1}_{i,j-\frac{1}{2},k} = v^n_{i,j-\frac{1}{2},k} + \frac{\Delta t}{2}(3F^n_{i,j-\frac{1}{2},k} - F^{n-1}_{i,j-\frac{1}{2},k}) \tag{22b}$$

$$w^{n+1}_{i,j,k-\frac{1}{2}} = w^n_{i,j,k-\frac{1}{2}} + \frac{\Delta t}{2}(3F^n_{i,j,k-\frac{1}{2}} - F^{n-1}_{i,j,k-\frac{1}{2}}) \tag{22c}$$

C-4. SCHEMATIC DESCRIPTION OF THE CELL COMPUTER AND THE DATA STRUCTURE

A schematic diagram of the cell computer is shown in Fig. 5. There is a two-dimensional array of L•M cells. Each cell connects (can communicate directly in the sense of interchanging data) with its four nearest neighbors "above" and "below" and to the "left" and "right." In order for cell (i,j) to communicate with cell (i-1,j+1), for example, the data must be transferred in a two-step operation; from (i,j) to (i,j+1) or (i-1,j) in step one and then to (i-1,j+1) in step two. It is clear, assuming that communication is slow, that local communication is relatively cheap and long-range communication may be prohibitively expensive. It is assumed that the left and right ends and the top and bottom of this cellular array are connected so that the array is equivalent to a torus.

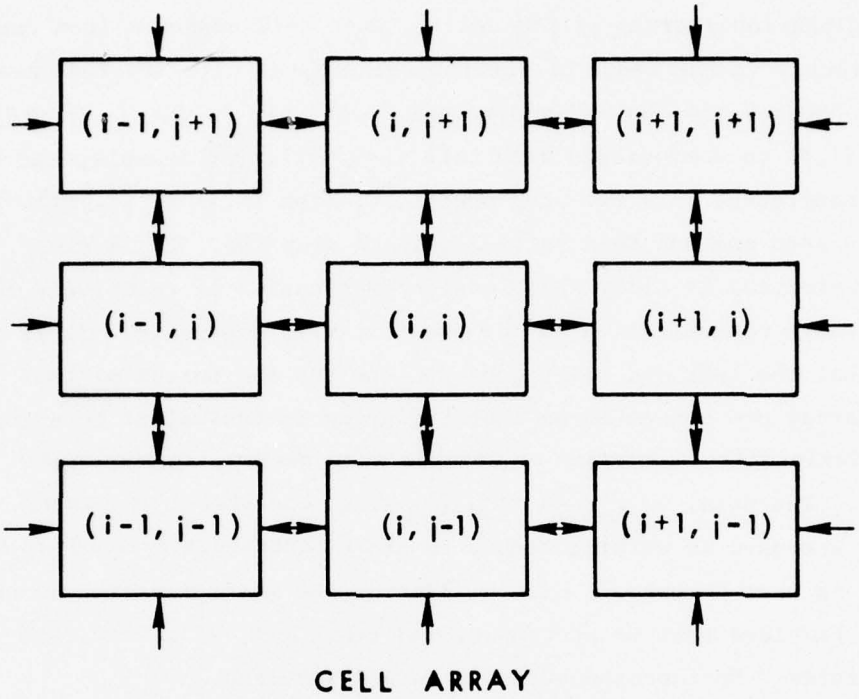
A single cell is assumed to contain some memory, an adder, and some registers. The data, $u_{i-\frac{1}{2},j,k}$, $v_{i,j-\frac{1}{2},k}$, etc. are stored in memory. The registers are used as working memory to store intermediate results, and the adder is used to perform binary addition and multiplication by shift and add. Provided that we precompute and store $1/\ell$, $1/\Delta x$, etc., we never need to divide. No function evaluations are required.

A column is defined as all those fluid cells at constant x; a row as all those fluid cells at constant ξ; and a rod as all those fluid cells at constant z. It will be assumed that one rod of data, the (i,j) rod, is stored in cell (i,j), that is

$$u_{i-\frac{1}{2},j,k},\quad v_{i,j-\frac{1}{2},k},\quad w_{i,j,k-\frac{1}{2}},\quad p_{i,j,k},\quad D_{i,j,k},\quad \text{etc.}$$

for k = 1, 2, 3, ..., N is stored in cell (i,j). This of course implies that cell (i,j) has some multiple of N words of memory plus sufficient memory for constants such as the metric coefficients, etc.

Not shown in Fig. 5 are the control word and status word paths. Instructions are sent, in parallel, from a central controller to all cells. Each cell, depending on control bits stored at initialization of the cell, either performs that instruction or does nothing. Each cell can send back to the controller a status word. All initialization, data, constants, and control bits are sent from the controller to the cells. All output from the cells is sent via the central controller.



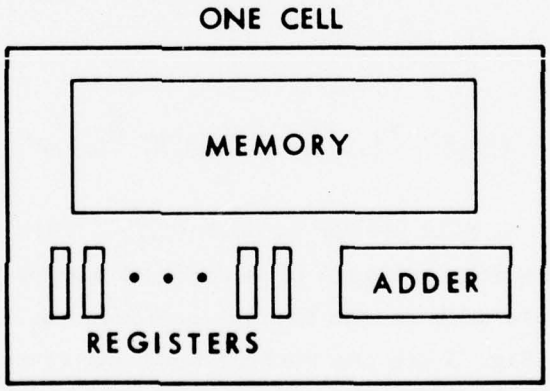


Fig. 5— Schematic representation of the cell-array computer

C-5. SAMPLE CALCULATIONS

In order to advance the calculation one time step, a number of quantities must be calculated. Among these are the set of $D_{i,j,k}$ divergences in each fluid cell. $D_{i,j,k}$ is defined by

$$D_{i,j,k} = \left(\frac{1}{\Delta x}\right)\left\{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k} + a_j\left(v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}\right) + \left(\frac{\Delta x}{\Delta z}\right)\left(w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}\right)\right\} \tag{23}$$

with

$$a_j = \left(\frac{\Delta x}{\Delta \xi}\right)\left[(1 - \xi)^2/\ell\right]_j \tag{24}$$

It will be assumed that the data paths within the cell are from memory to the register bank and back, and from any two registers to the adder and back to any register. If it is assumed that $(1/\Delta x)$, a_j, $(\Delta x/\Delta z)$, $u_{i-\frac{1}{2},j,k}$, $v_{i,j-\frac{1}{2},k}$, and $w_{i,j,k-\frac{1}{2}}$ are already in register, then, for fixed k, one in-cell transfer (1t) is needed to get $w_{i,j,k+\frac{1}{2}}$, and two cell to cell transfers (2T) are needed to get $u_{i+\frac{1}{2},j,k}$, $v_{i,j+\frac{1}{2},k}$. Once these are in the registers, three additions (3a) are performed, followed by two multiplications (2m), two more additions (2a), and finally one more multiplication (1m). This completes the calculation of $D_{i,j,k}$ for one k.

This must be repeated for each cell in the rod, k = 1, 2, ..., N; although all cells in the k'th plane are calculating in parallel. Therefore the total number of sequential operations which are required for calculating the $D_{i,j,k}$ for all i,j, and k is

Nt, 2N, 5Na, and 3Nm.

In addition to the velocity components each cell must contain the constants $(1/\Delta x)$, a_j, and $(\Delta x/\Delta z)$.

In order to advance $\vec{u}^n$ by one time step it is necessary to calculate D, then E, then S, then p, then $\vec{F}^n$, and finally $\vec{u}^{n+1}$. The operation count for one time step can be obtained by writing the finite difference equations for each of these quantities, finding where the required data are stored, counting the number of transfers needed to get these data into the registers in cell (i,j) and the number and type of arithmetic operations required.

It turns out that, as was expected, the entire calculation is largely local in that most cell-to-cell transfers are between nearest neighbors. Figure 6 shows cell (i,j) and its neighbors. All the data, and only the data, required to calculate $F^n_{i-\frac{1}{2},j,k}$, $F^n_{i,j-\frac{1}{2},k}$, and $F_{i,j,k-\frac{1}{2}}$ are shown in this figure in the cells where they are stored. It can be seen that twenty words of data must be transferred to cell (i,j); of these twenty only two, $u_{i+\frac{1}{2},\ j-1,k}$ and $v_{i-1,\ j+\frac{1}{2},k}$, require a two-step transfer.

C-6. THE POISSON PROBLEM

As one might expect, the most difficult part of the calculation is the solution of the Poisson equation with Neumann boundary conditions. This problem is elliptic, i.e., nonlocal. A direct solution, say by using a fast Fourier transform (FFT) in one direction and a tridiagonal equation solver in the other, is prohibitive because of the transfer time cost. More important, these methods are restricted to flows with very regular geometry; general geometries require the use of relaxation methods.

Consider a model problem.

$$\nabla^2 P = \frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial z^2} = Q \tag{25}$$

in $0 \le x \le 1$, $0 \le z \le 1$; Q given and

$$\hat{n} \cdot \nabla P = 0 \tag{26}$$

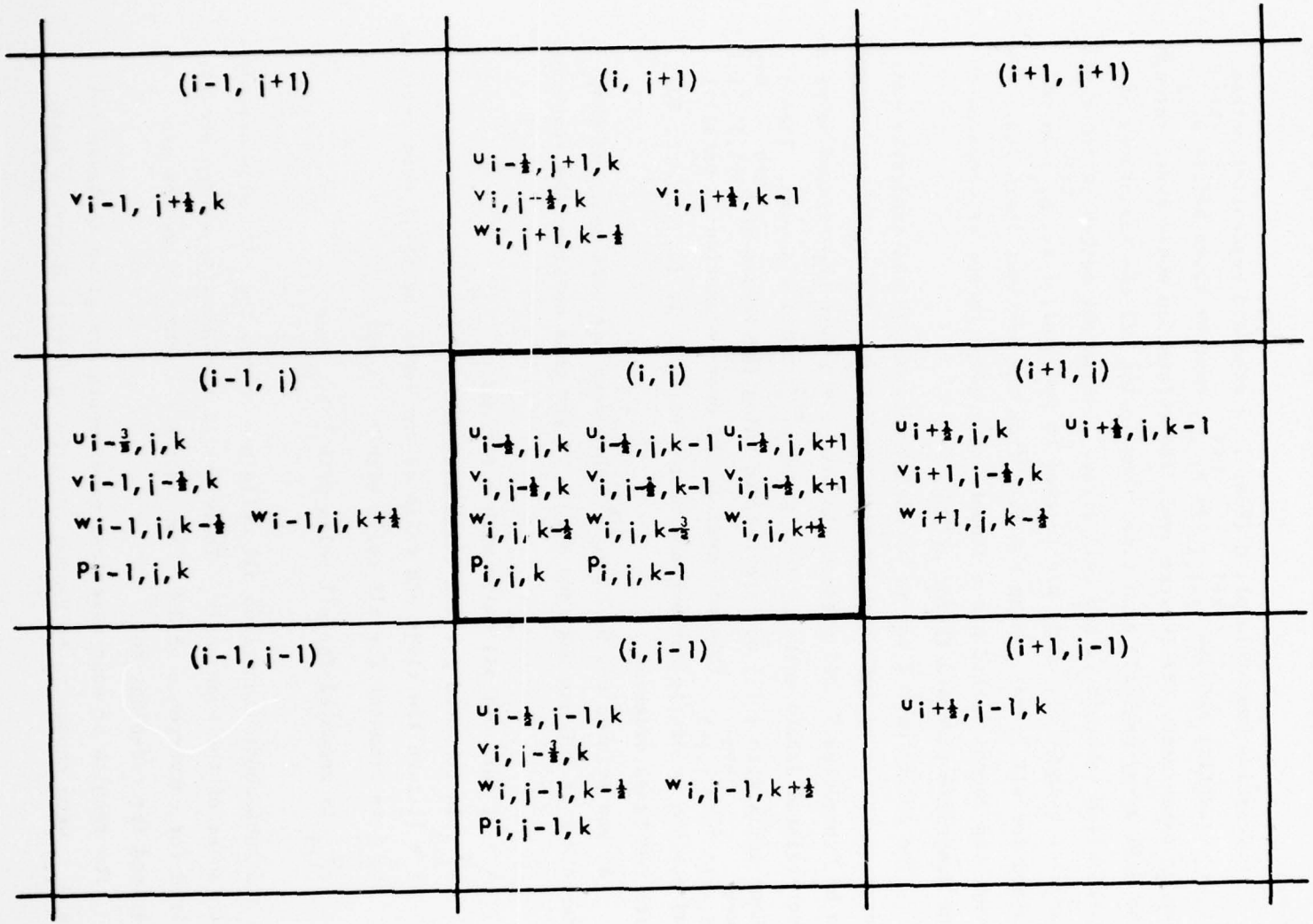


Fig. 6 — Data required to compute $\overrightarrow{F^{n}_{ijk}}$

with $\hat{n}$ a unit vector normal to the boundary on x = 0, x = 1, z = 0, and z = 1.

Successive over-relaxation (SOR) is a standard sequential method. It is sequential because $p_{i-1,j}^{k+1}$ and $p_{i,j-1}^{k+1}$ must be known before $p_{i,j}^{k+1}$ can be calculated. If it were used, the relaxation would sweep through the cell array one cell at a time, thus losing all the advantages of parallelism inherent in the cell array. The Jacobi method is inherently *parallel* because all cells are relaxed in parallel, i.e., $p_{i,j}^{k+1}$ is obtained for all i and j using the data from the previous iteration. However, the Jacobi method has a linear convergence factor as compared to the quadratic coverage factor of SOR.

The Jacobi method can be accelerated to obtain the quadratic convergence factor of SOR while remaining parallel. The cells are assumed to be "reordered." *All* those cells with i + j even are relaxed using an over-relaxed Jacobi method. This gives $p_{i,j}^{k+\frac{1}{2}}$, for i + j even. Then *all* those cells with i + j odd are relaxed using the values of $p_{i+1,j}^{k+\frac{1}{2}}$, $p_{i-1,j}^{k+\frac{1}{2}}$, $p_{i,j+1}^{k+\frac{1}{2}}$, and $p_{i,j-1}^{k+\frac{1}{2}}$. This odd/even Jacobi over-relaxation is parallel and has the quadratic convergence factor of SOR. It is, in fact, SOR with red-black ordering.

As was pointed out by King,[16] this does not cost twice as many iterations as SOR but only *one* more. To see this consider the sequence:

k = 0; turn off all cells with i + j odd.
$p_{i,j}^{\frac{1}{2}}$ is computed for all cells with i + j even.
k = 1; turn the (i+j) odd cells on and leave the (i+j) even cells on.
$p_{i,j}^{1}$ is computed for all cells with i + j odd.
$p_{i,j}^{\frac{1}{2}}$ is computed for all cells with i + j even.

In all subsequent iterations all cells are on and the odd cells are one step ahead of the even cells. Therefore if K *sequential* sweeps are required for convergence of SOR, then only K + 1 *parallel* sweeps are required for red-black SOR.

The results of some numerical experiments are given in Table C-1. The $Q_{i,j}$ were chosen to be random variates in (-1,1) with zero mean.

Table C-1

RESULTS OF THE RELAXATION SOLUTION TO $\nabla^2 P = Q$

NEUMANN BOUNDARY CONDITIONS

(N+2) × (N+2) MESH

N	ε	ω	K
		S O R	
10	10^{-4}	1.0	173
		1.1	110
		1.2	111
		1.3	105
		1.4	76
		1.5	79
		1.6	61
		1.7	55
		1.8	39
		1.9	57
20	10^{-4}	1.8	121
50	10^{-4}	1.8	489
		J A C O B I	
10	10^{-4}	1.0	263
20	10^{-4}	1.0	740
50	10^{-4}	1.0	3865
		R E D -- B L A C K S O R	
10	10^{-4}	1.0	166
		1.1	103
		1.2	105
		1.3	102
		1.4	77
		1.5	73
		1.6	56
		1.7	45
		1.8	33
		1.9	51
20	10^{-4}	1.8	104
50	10^{-4}	1.8	463

In this table ε is the error in the solution, i.e., if r_{ij} is the residual

$$r_{i,j} = Q_{i,j} - \nabla^2 p_{i,j} \tag{27}$$

convergence occurs when

$$\text{Max } |r_{i,j}| \leq \varepsilon$$

The relaxation factor is ω, and K is the number of iterations required to converge. Note that red-black SOR converges in slightly fewer iterations than SOR.

It has been assumed that the flow is periodic in z for the test simulation on this array computer. This suggests that p and Q be Fourier-transformed in the z direction. It is obvious that this converts one three-dimensional Poisson problem over L•M•N points to N two-dimensional Poisson problems each over L•M points. It is assumed, for this test case, that only a fraction, $0 < f < 1$, of the Fourier modes require relaxation. If all Fourier modes contained appreciable amounts of energy, there would be an aliasing problem; in effect the resolution in z is too coarse.

Two points must be made. The first is that the Fourier decomposition is *not* crucial. This decomposition only affects the order in which the calculation is done. If we Fourier-transform, the relaxation is over all (i,j) for each of N modes; mode zero is relaxed, then mode one, then mode two, etc. If the z boundary conditions were different so that we could not Fourier-transform in z, we would be relaxing for all (i,j) for $K = 1, 2, \ldots, N$ and then back to $K = 1$, etc. until all $p_{i,j,k}$ are determined. The only difference is the order of the operations--not the number of operations.

The second point is that a "standard" relaxation method (modified slightly for a cell computer) has been assumed. We have not yet examined other methods such as the cell-by-cell divergence-pressure method of Hirt and Cook[17] or even the use of block or multigrid relaxation methods,[18] possibly to increase the convergence rate.

C-7. OPERATION COUNT FOR ONE TIME STEP

The number of in-cell transfers, cell-to-cell transfers, additions, and multiplications necessary for each phase of the calculation to advance $\vec{u}$ by one time step are given in Table C-2, and the totals for one time step are summarized in Table C-3. In these tables, N is, of course, the number of mesh points in the transverse flow direction (z direction), and the product fK is the number of iterations for relaxation of the Poisson equation.

From an examination of these tables, assuming that fK = 50 and N = 64, say, several facts become apparent. First (Table C-2) operations (1) through (9), which constitute 75 percent to 80 percent of the total transfers, additions, and multiplications, are required to calculate the pressure field. The calculation of the velocity field, operations (10), (11), and (12), constitute only about 20 to 25 percent of the operations. Second, about 60 percent of the transfers but only 20 percent of the additions and 10 percent of the multiplications are required to apply the boundary conditions. Third, if we take the in-cell transfer count as unity, the cell-to-cell transfer count is about 7, the addition count 14, and the multiplication count is about 5. Because it is generally true that the in-cell transfer time and the addition time are considerably smaller than the cell-to-cell transfer time and the multiplication time, it is clear that the total calculation time is controlled by the cell-to-cell transfer and multiplication counts and times. Fourth, the ratio of the multiplication count to cell-to-cell transfer count is about one, so reducing either cell-to-cell transfer time or multiplication time to zero would result in a speedup of only a factor of two.

C-8. PERFORMANCE OF AND MEMORY REQUIRED FOR SOME SAMPLE PROBLEMS

In order to gain some insight into the performance capabilities of this type of array computer, three sample problems will be considered:

(1) A (logical) array of 50 × 200 cells with N = 32; 50 points in the direction normal to the boundary, 200 in the downstream direction and 32 in the cross-stream direction. It is believed that this "one-third of a million point" problem is the absolute minimum problem of interest.

Table C-2

OPERATION COUNT FOR EACH PHASE OF ONE TIME STEP

Phase	Calculation	In-Cell Transfers	Cell-to-Cell Transfers	Additions	Multiplications
1	D_{ijk}	N	2N	5N	3N
2	E_{ijk}	2N	2N	14N	5N
3	S_{ijk} & Q_{ijk}	15N	23N	50N	30N
4	Fourier-Transform Q_{ijk} & p_{ijk} over k, i.e., z direction	2N	0	$\frac{1}{2}N\log_2 N$	$N\log_2 N$
5	Pressure-Boundary Conditions	5N	7N	27N	14N
6	Fourier-Transform Pressure-Boundary Conditions	4N	0	$N\log_2 N$	$2N\log_2 N$
7	Relax $p_{ij\alpha}$	0	2fKN	6fKN	3fKN
8	Apply Boundary Conditions on $p_{ij\alpha}$	0	4fKN	2fKN	0
9	Inverse Fourier-Transform $p_{ij\alpha}$	2N	0	$\frac{1}{2}N\log_2 N$	$N\log_2 N$
10	$\vec{F}^n$ (3 Components)	11N	16N	94N	38N
11	$\vec{u}^n$ (3 Components)	3N	0	12N	3N
12	Apply Boundary Conditions on $\vec{u}^n$	3N	13N	6N	0

f = Fraction of Fourier modes containing energy (f = $\frac{1}{2}$?).
K = Number of iterations required for the relaxation solution for $p_{ij\alpha}$ to converge (K = 50 ?).
N = Number of mesh points in the transverse flow direction (z direction); also number of Fourier modes in z.

Table C-3

TOTAL OPERATION COUNT FOR ONE TIME STEP

IN-CELL TRANSFERS = 48N
CELL-TO-CELL TRANSFERS = (63 + 6fK)N
ADDITIONS = (208 + 8fK + $2\nu_2$N)N
MULTIPLICATIONS = (93 + 3fK + $4\nu_2$N)N
Number of words of memory = 18 + 12N
Number of registers = 30

f = Fraction of Fourier modes containing energy (f = ½ ?).
K = Number of iterations required for the relaxation solution for one Fourier mode of the pressure field to converge (K = 50 ?).
N = Number of mesh points in the transverse flow direction (z direction); also number of Fourier modes in z.

(2) A (logical) array of 50 × 200 cells with N = 128; 50 points normal to the boundary, 200 in the downstream direction and 128 across the stream. The "million point" problem is quite interesting.

(3) A (logical) array of 100 × 1000 cells with N = 128; 100 points normal to the boundary, 1000 points in the downstream direction and 128 points across the stream. This "ten million point" problem is very interesting.

In order to calculate computation time per time step it is necessary to make assumptions about the in-cell transfer, the cell-to-cell transfer, the addition, and the multiplication times as well as to assume values for f and K. It will be assumed that

computer. This table is reproduced from Case et al.[19] In this technique, the spatial differencing is the same as that used in the analysis of the array computer's performance, second-order central differences; the time differencing is the leap-frog type; and a fast Poisson solver is used. The downstream direction has L points, the cross-stream direction has N points, and the direction normal to the boundary has M points. It will be assumed that, for the conventional computer (e.g., a CDC 7600),

$$\text{Addition time} = 10^{-7} \text{ sec} = 100 \text{ nsec}$$

$$\text{Multiplication time} = 2 \times 10^{-7} \text{ sec} = 200 \text{ nsec}$$

$$\text{Memory transfer time} = 10^{-6} \text{ sec} = 1 \ \mu\text{sec}$$

The estimates for each of the three problems are given in Table C-5. These estimates are broken down in several ways (all times are in seconds). Column one lists the problems. The total number of bits of memory required for each problem is listed in column two. In the third column the time to advance the interior points one time step is given, while in the fourth column is the time required to advance the boundary points one time step. The sum of these is the total time required to advance all the grid points one time step, and this is given in column seven. This total time is also the sum of the time required to compute the pressure field (given in column five; and the time required to compute the velocity field (column six). Finally, column eight is the total time required to calculate one time step on a conventional computer. This time was calculated from the total operation count given in Table C-4 and the addition, multiplication, and memory transfer times for the conventional computer, as tabulated above.

From an examination of Table C-5, it is clear that most of the computation time of the array computer is spent computing the pressure field for the interior points of the grid; the ratio of the calculation time for the interior to the calculation time for the boundary is about 2, while the ratio of the calculation time for the pressure field to that for the velocity field is about 7. This holds true for all three

Table C-5

TIME AND MEMORY ESTIMATES

Problem	Memory per Cell (words/bits)	Calculation Time for Interior Points per Time Step (sec)	Calculation Time for Boundary Points per Time Step (sec)	Calculation Time for the Pressure Field per Time Step (sec)	Calculation Time for the Velocity Field per Time Step (sec)	Total Calculation Time per Time Step (sec)	Total Calculation Time for One Time Step on a Fast Conventional Computer (sec)
(1) 1/3 million point problem	432/~14K[a]	0.060	0.027	0.076	0.011	0.087	20.0
(2) 1 million point problem	1584/~50K	0.242	0.112	0.309	0.045	0.354	81.9
(3) 10 million point problem	1587/~50K	0.242	0.112	0.309	0.045	0.354	839

[a] 1K = 1024.

problems. In short, the dominant part of the calculation is the relaxation solution of the pressure field on the interior of the grid.

Although it is not shown in this table, precisely the opposite holds true for the calculation on the fast conventional computer using a fast Poisson solver, wherein the ratio of the velocity-field calculation time to the pressure-field calculation time is about 3.

For problems (1) and (2), the ratio of total calculation time on the conventional computer to that of the array computer is ~230. Increasing the number of grid points in the cross-stream direction from 32 to 128 changes this ratio by less than 1 percent. However, increasing the number of cells from 10^4 to 10^5 increases the ratio to about 2370. As was expected, the ten-fold increase in the number of cells is fully reflected in the speed ratio.

The problem with 32 grid points in the cross-stream, z, direction requires 432 words of memory, about 14K bits per cell. Increasing the number of grid points in the z direction to 128 increases the size of the required cell memory to 1584 words or about 50K bits. Problem (1) fits nicely into a 16K bit memory and problems (2) and (3) are good fits to a 64K bit memory.

C-9. SUMMARY OF CONCLUSIONS

A standard algorithm, using a relaxation method to solve for the pressure field, fits quite well into an array computer. No major modifications of the algorithm were required. The entire calculation is dominated by the relaxation solution on the interior grid points of the Poisson equation for the pressure field. The number of cell-to-cell transfers is about equal to the number of multiplications.

A calculation with 10^4 cell processors and 32 grid points in the cross-stream direction requires about 0.09 seconds per time step, which is about 230 times faster than a fast conventional computer. It requires about 14K bits of memory per cell.

If the number of grid points in the cross-stream direction is increased to 128, the calculation time per time step increases to about 0.35 seconds, and the required memory per cell increases to about 50K bits. The ratio of the speed of the array to that of a conventional computer remains about 230.

If the cell array is enlarged by a factor of 10, i.e., 10^5 cell processors, the calculation time and the per-cell memory requirements for the cell computer are unchanged, but the speed ratio increases to about 2370, fully reflecting the increase in the cellular-array size.

REFERENCES

1. Thomas, L. H., "The Stability of Plane Poiseuille Flow," *Physical Review*, Vol. 91, 1953, p. 780.

2. Grosch, C. E. and H. Salwen, "The Stability of Steady and Time-Dependent Plane Poiseuille Flow," *Journal of Fluid Mechanics*, Vol. 34, 1968, p. 177.

3. Grosch, C. E., Private communication, 1977.

4. Wazzan, A. R. and C. Gazley, Jr., "The Combined Effects of Pressure Gradient and Heating on the Stability and Transition of Water Boundary Layers," to be published in *Proceedings, 2d Drag Reduction Conference, St. John's College, Cambridge, England, August 31-September 2, 1977;* also The Rand Corporation, R-2175-ARPA (in process).

5. Corrsin, S., "Turbulent Flow," *American Scientist*, Vol. 49, 1961, pp. 300-325.

6. Emmons, H. W., "Critique of Numerical Modeling of Fluid Mechanics Phenomena," M. Van Dyke et al. (eds.), *Annual Review of Fluid Dynamics*, Vol. 2, Palo Alto, California, 1970, pp. 1-14.

7. Case, K. M., F. J. Dyson, E. A. Frieman, C. E. Grosch, and F. W. Perkins, "Numerical Simulation of Turbulence," *Stanford Research Institute Technical Report JSR-73-3*, Palo Alto, California, 1973.

8. Case, K. M., A. M. Despain, E. A. Frieman, F. W. Perkins, and J. F. Vesecky, "Problems in Laminar Flow-Turbulent Flow," *Stanford Research Institute Technical Report JSR-74-2*, Palo Alto, California, 1975.

9. Thurber, K. J., *Large Scale Computer Architecture: Parallel and Associative Processors*, Hayden Publications, Rochelle Park, N.J., 1976.

10. Enslow, P. H. (ed.), *Proceedings of the 1976 International Conference on Parallel Processing*, Institute of Electrical and Electronics Engineers, Association for Computing Machinery, Wayne State University, Detroit, Michigan.

11. Miranker, W. L., "A Survey of Parallelism in Numerical Analysis, *Society for Industrial and Applied Mathematics Review*, Vol. 13, No. 4, October 1971, pp. 524-547.

12. *Proceedings of the Conference on Programming Languages and Compilers for Parallel and Vector Machines*, March 18-19, 1975, Goddard Institute for Space Studies, New York, N.Y., sponsored by Association for Computing Machinery, Special Interest Group on Programming Languages, National Aeronautics and Space Administration, in *SIGPLAN Notices*, Vol. 10, No. 3.

13. *Proceedings of the 1975 Sagamore Computer Conference on Parallel Processing*, Institute of Electrical and Electronics Engineers, Syracuse University, Syracuse, N.Y.

14. Traub, J. F., (ed.), *Proceedings of Conference on Complexity of Sequential and Parallel Numerical Algorithms*, Carnegie-Mellon University, Pittsburgh, Academic Press, New York, 1973.

15. Grosch, C. E., and S. A. Orszag, "Numerical Solutions of Problems in Unbounded Domains: Coordinate Transformations," *Journal of Computational Physics* (in press), 1977.

16. King, W., Private communication, 1977.

17. Hirt, C. W., and J. L. Cook, "Calculating Three-Dimensional Flows Around Structures and Over Rough Terrain," *Journal of Computational Physics*, Vol. 10, 1972, p. 324.

18. Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Institute for Computer Applications in Science and Engineering Report 76-27*, Institute for Computer Applications in Science and Engineering, National Aeronautics and Space Administration, Langley, Hampton, Va., 1976.

19. Case, K. M., A. M. Despain, E. A. Frieman, F. W. Perkins, and J. F. Vesecky, "Problems in Laminar Flow-Turbulent Flow," *Stanford Research Institute Technical Report JSR-74-2*, Palo Alto, California, 1975.