NASA Contractor Report 181770

ICASE REPORT NO. 89-1

ICASE

SPECTRAL SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS ON THE CONNECTION MACHINE 2

Sherryl Tomboulian

Craig Streett

Michele Macaraeg

Contract No. NAS1-18107, NAS1-18605
January 1989

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Spectral Solution of the Incompressible Navier-Stokes Equations on the Connection Machine 2

Sherryl Tomboulian [*]

MassPar Inc., 2840 San Tomas Exprswy, Suite 140 Santa Clara, CA 95051

Craig Streett

Mail Stop 159, NASA Langley Research Center, Hampton, VA 23665

Michèle Macaraeg

Mail Stop 156, NASA Langley Research Center, Hampton, VA 23665

ABSTRACT

This paper addresses the issue of solving the time-dependent incompressible Navier-Stokes equations on the Connection Machine 2, for the problem of transition to turbulence of the steady flow in a channel. The spectral algorithm used serially requires $O(N^4)$ operations when solving the equations on an N x N x N grid; using the massive parallelism of the CM, it becomes an $O(N^2)$ problem. Preliminary timings of the code, written in LISP, are included and compared with a corresponding code optimized for the Cray-2 for a 128 x 128 x 101 grid.

KEYWORDS: Parallel Algorithms, SIMD Architectures, Navier-Stokes equations, Spectral Methods

[*] Research was supported by the National Aeronautics and Space Administration under NASA contract No. NAS1-18107 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton VA, 23665.

INTRODUCTION

Fast solutions to the Navier-Stokes equations are of enduring interest in computational fluid dynamics. The level of resolution required to accurately simulate the relevant physics of flows involving transition to turbulence may span scales ranging eight orders or magnitude or more. Existing supercomputers still lack the capacity to resolve such problems. In this paper, we examine the possibility of employing massive parallelism towards meeting the ever-increasing computational demands of large-scale fluid dynamic simulations.

The general framework is the time-dependent incompressible Navier-Stokes equations. One of the methods of choice for solving these equations when a high degree of accuracy is required is spectral methods, and we are in the process of examining the feasibility of implementing spectral methods on the Connection Machine 2. The particular tensor-product matrix formulation used here involves kernels which are amenable to implementation on this architecture. These basic kernels are presented, along with a spectral algorithm that uses these kernels to solve the Navier-Stokes equations.

MACHINE MODEL

The Connection Machine 2 [6,2] is a parallel architecture with 65,000 processors. The machine can be segmented into quarters, and all the work on this paper was done on a 16K processor machine. Each processor has 64K bits of memory, or 2K of single precision floating point numbers. Each subset of 32 processors has a floating point co-processor available.

The processors are SIMD (single instruction multiple data), meaning that all the processors must do exactly the same thing at the same time. A serial processor, called the controller, gives the same instructions to the parallel array or processors. As part of the instruction, scalar values can be broadcast. For instance, one could instruct all processors to add '5' to location A in their local memories.

The processors are interconnected in a topology that is hyper-cube like. For the purposes of this paper it is sufficient to consider the machine as arranged as a 128 by 128 toroidal grid of processors, where each processor can communicate with its north, south, east, and west neighbors. A communication facility called the router can also be used for more generalized communication.

NUMERICAL METHOD

In this paper we solve the time-dependent incompressible Navier-Stokes equations:

$$\vec{u}_t + \vec{u}\cdot\nabla\vec{u} = -\nabla P + \nu\nabla^2\vec{u} \tag{1a}$$
$$\nabla\cdot\vec{u} = 0 \tag{1b}$$

The method employed for incremental time solution is a standard splitting method [1]. With $\vec{u}$ representing the three-dimensional velocity vector, we advance the solution from time level 'n' to level 'n+1' via:

$$\vec{u}_t^{\,*} + \vec{u}^n\cdot\nabla\vec{u}^n = \nu\nabla^2\vec{u}^{\,*} \tag{2}$$

which solves the momentum equations without the pressure terms, followed by:

$$\vec{u}_t^{\,n+1} = -\nabla P^{n+1} \tag{3a}$$
$$\nabla\cdot\vec{u}^{\,n+1} = 0 \tag{3b}$$

which solves for the pressure and corrects the velocity to satisfy continuity. This second step is actually carried out in two stages:

$$\nabla^2 P^{n+1} = \nabla\cdot\vec{u}^{\,*} \tag{4a}$$
$$\vec{u}_t^{\,n+1} = -\nabla P^{n+1} \tag{4b}$$

by utilizing a backward-Euler time discretization of eq. 4b. The time discretization employed in advancing eq. 2 is a mixed implicit-explicit scheme, with third-order low-storage Runge-Kutta for the nonlinear convection terms, and Crank-Nicholson for the viscous terms.

Spatial discretization is carried out using a spectral collocation method; Fourier series are used in the streamwise and spanwise directions of the channel, with Chebyshev series used in the normal direction between the channel walls. The now-standard matrix method is used for evaluating derivative quantities at the collocation points [1]. That is, the approximate derivatives of a function evaluated at the collocation points may be computed directly by a matrix-vector product; the coefficients of the basis function series need not be explicitly calculated. Formulae for the required Fourier and Chebyshev differentiation matrices are known in closed form [1]. Although at first glance this appears to require N^6 operations to compute the derivative in one direction at each point of an N x N x N three-dimensional grid, the computation can actually be carried out in N^4 operations, using the fact that the discretizations in each coordinate direction are separable. This is known as a tensor-product formulation. Similarly, if the overall discrete differential operator is linear and separable, its characteristic matrices may also be written in tensor product form, and thus the overall operator may be diagonalized in $O(N^4)$ operations [1].

It may be noted that since Fourier discretizations are used in this application, the scheme may be formulated using Fast Fourier Transforms (FFT's) rather than tensor products, with asymptotic work of $O(N^3 \log N)$ rather than $O(N^4)$. We decided, however, to study a general implementation of spectral collocation solution of the Navier-Stokes equations, rather than limit the implementation to the restrictive periodic form. The additions to the time-split scheme to enforce general boundary conditions in all directions are minor [5]. Additionally, for the initial proof-of-concept phase of this project, approximate pressure boundary conditions are used.

KERNEL OPERATIONS

The application we report here is on a 16K CM, with its processors configured as a 128 x 128 toroidally-cyclic grid; we denote processor number by the subscripts 'i,j'. The three-dimensional computational grid is mapped onto the processors in a straightforward manner, with the third direction, the 'z' or 'normal' direction, stored locally in each processor, and denoted by the subscript 'k'. Given the storage requirements of the scheme and the available local memory of each processor, the z-direction is discretized with 101 points, giving an overall mesh of about 1.65 million points.

The bulk of computation in this scheme lies in the tensor product operations. In indical notation, these operations in each of the three directions are defined:

$$T_x(L^x, u) = \sum_{l=1}^{l=N} L^x_{il}\, u_{ljk} \tag{5a}$$

$$T_y(L^y, u) = \sum_{l=1}^{l=N} L^y_{jl}\, u_{ilk} \tag{5b}$$

$$T_z(L^z, u) = \sum_{l=1}^{l=N} L^y_{kl}\, u_{ijl} \tag{5c}$$

The x-direction tensor product is simply a matrix multiply between L^x and each of the k-planes of U. Likewise, the y-direction tensor product is $U(L^y)^T$ for each k-plane. These matrix multiplies are carried out in O(N) operations instead of O(N^3), using the systolic algorithm [3]. This is because all N^2 processors can be used simultaneously. The algorithm works as follows: Before starting the multiply, the first matrix is skewed so that each row in the matrix is shifted over one position for each row down (i.e. the elements in row 3 are shifted 3 places to the right). Similarly, the columns of the second matrix are shifted down for each column over. The multiplication is then performed by shifting each row of the first matrix up one and each column of the second matrix over one, multiplying the elements and accumulating the result, then shifting again. This pattern is repeated N times to complete the product. The overall tensor product requires N separate matrix multiplies (one for each k-plane) so the product requires O(N^2) operations.

The summation in the third tensor product is independent of (i,j), so each processor decouples and performs a matrix-vector product. The operator L^z is stored in the CM controller and broadcast element-by-element in the process of computing $T_z(L^z, u)$.

ALGORITHM IMPLEMENTATION

The algorithm consists of applications of the three tensor products defined above (eqs. 5), and direct element-by-element sums (+) and multiplies ($\times$). The velocity variables are denoted u_n, n= x to z, and P for the pressure. D^n and D^{nn} (n= x to z) are used to denote the spectral differentiation operators, first and second derivative respectively, in each of the three

```
begin (time step)
$e_n = 0$, n = x to z                          -- initialize

for m= 1 to 3 do:                              -- loop on the three steps of the Runge-Kutta

    for n= x to z do:                          -- loop over components of
                                               -- the momentum equation

    -- Compute the RHS of the Crank-Nicholson step (eq. 2); the first three terms
    -- are viscous terms, the next three terms are convection terms, and the last
    -- is due to the Runge-Kutta scheme.

        $f_1 = T_x\left[D^{xx}, u_n\right] + T_y\left[D^{yy}, u_n\right] + T_z\left[D^{zz}, u_n\right]$
        $\quad + k_m \times \left[u_x \times T_x\left[D^x, u_n\right] + u_y \times T_y\left[D^y, u_n\right] + u_z \times T_z\left[D^z, u_n\right]\right] + e_n$

        $e_n = d_m^1 e_n + d_m^2 f_1$              -- update temporary for Runge-Kutta scheme

    -- Do tensor-product diagonalization for inversion of viscous operator
    -- in Crank-Nicholson

        $f_2 = T_x\left[R^x, T_y\left[R^y, T_z\left[R^z, f_1\right]\right]\right]$
        $f_3 = C_m \times f_2$
        $g_n = T_x\left[Q^x, T_y\left[Q^y, T_z\left[Q^z, f_1\right]\right]\right]$

    end (for n)                                -- end components loop

    $u_n = g_n$,  n = x to z                     -- update velocities for next step of Runge-Kutta

end (for m)                                    -- end R-K steps loop

    -- Compute pressure; generate RHS of 4a

        $g_1 = k_4\left[T_x\left[D^x, u_x\right] + T_y\left[D^y, u_y\right] + T_z\left[D^z, u_z\right]\right]$

    -- Invert Laplacian by tensor-product diagonalization

        $g_2 = T_x\left[R^x, T_y\left[R^y, T_z\left[R^z, g_1\right]\right]\right]$
        $g_3 = C_4 \times g_2$
        $P = T_x\left[Q^x, T_y\left[Q^y, T_z\left[Q^z, g_3\right]\right]\right]$

    -- Correct Velocities using eq. 4b

        $u_n = u_n + k_5\, T_n\left[D^n, P\right]$,    n=x to z

end (time step)
```

Figure 1. The Algorithm for one time step

directions. These operators are used in the appropriate tensor-product form to compute derivatives of the dependent variables as needed. The characteristic matrices used to diagonalize the implicit operator are denoted as Q^n and R^n (n= x to z), and are also used in the appropriate tensor-product form. In addition, the following quantities are used:

- k_m, m= 1 to 5, are algorithm and case-specific constants.
- d_m^1 and d_m^2, m= 1 to 3 are constants from the Runge-Kutta algorithm.
- C_m, m= 1 to 4 is an n x n x n field of constants used in the tensor-product diagonalization process.
- e_n, f_n, and g_n are temporary field arrays, used for clarity of presentation.

The algorithm for one time step is given in Figure 1.

COMPLEXITY AND TIMINGS

By using N^2 processors, the overall complexity reduces to $O(N^2)$ instead of $O(N^4)$. Hence, the parallel solution time should clearly be superior to serial solutions. However, one tradeoff with having a large number of processors is that they are not individually as fast as one might like. There is also communications overhead. Timing results are made comparing a CM2 and similar code on a Cray-2. For smaller problems, such as a 32^3 problem, the Cray is significantly faster than the CM. However, as the problem size is increased, the factor of $O(N^2)$ dominates, and the CM becomes more competitive.

The table below shows timing results for the two codes. The Cray achieves better timings, but this result is a somewhat unfair comparison. The Cray results were achieved on 1 processor of a Cray2 using the local memory micro-coded matrix operations provided by the SARL library which achieve peek performance. The CM code is written in *Lisp, a higher level language, and is not particularly well optimized. Nevertheless, the CM does achieve a rate of about 60 Megaflops (counting the multiply and add as 2 instructions).

Cray2 (1 proc) vs CM2 (16K proc) solving a (128 x 128 x 101) problem		
operations	Cray2	CM2
direct multiply	0.068	0.015
matrix op T_x	1.41	7.89
matrix op T_y	1.55	7.90
matrix op T_z	1.55	5.70
overall	216.	857.

Table 1 -- Timing Results in Seconds.

It seems obvious that for even larger problems, the CM algorithm will be much faster than the Cray. However, using this method, the 128x128x101 problem is the largest practical problem on a 16K CM2 because of the size of local memories. Using a 65K processor CM2, we estimate that the discretization of a (256×256×101) problem could be solved several times faster than a Cray-2.

CONCLUSIONS

The results obtained from implementing the Navier-Stokes equations were quite encouraging. The Connection machine was slower but still competitive with respect to time with a Cray, while being significantly more cost effective. More spectacular results can be expected with larger versions of the Connection Machine.

There is one final interesting point concerning this work: We have shown the efficient implementation of an implicit time-stepping algorithm utilizing a global discretization scheme on a SIMD parallel-processing machine. This is contrary to the "popular wisdom" which holds that SIMD machines are useful only when nearest-neighbor operations, such as those of an explicit finite-difference scheme, are carried out. The potential exists for this implementation to provide numerical simulations of transition to turbulence in channel flow at resolutions impractical on present supercomputers.

BIBLIOGRAPHY

[1] C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Zang, "Spectral Methods in Fluid Dynamics," Springer-Verlag, 1988.

[2] W. Hillis, "The Connection Machine," MIT Press, Cambridge, Massachusetts, 1985.

[3] L. Johnsson, "Dense Matrix Operations on a Torus and a Boolean Cube," Yale Report U/DCS/RR-377, March 1985.

[4] R. E. Lynch, J. R. Rice, and D. H. Thomas, "Tensor Product Analysis of Alternating Direction Implicit Methods," J. S. Indust. Appl. Math., Vol. 13, No. 4, December 1965.

[5] C. L. Streett, "Spectral Methods and their Implementation to Solution of Thermodynamic and Fluid Mechanic Problems," presented at the 6th International Symposium on Finite Element Problems, Antibes, France, June 1986.

[6] L. W. Tucker and G. G. Robertson, "Architecture and Applications of the Connection Machine," to appear in IEEE Trans. Comput.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-181770 ICASE Report No. 89-1	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle SPECTRAL SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS ON THE CONNECTION MACHINE2		5. Report Date January 1989
		6. Performing Organization Code
7. Author(s) Sherryl Tomboulian, Craig Streett, Michele Macaraeg		8. Performing Organization Report No. 89-1
		10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23665-5225		11. Contract or Grant No. NAS1-18107, NAS1-18605
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration, Langley Research Center, Hampton, VA 23665-5225		14. Sponsoring Agency Code

15. Supplementary Notes
Langley Technical Monitor: Richard W. Barnwell

Proc. of Supercomputing 1988 Orlando, Florida

Final Report

16. Abstract

This paper addresses the issue of solving the time-dependent incompressible Navier-Stokes equations on the Connection Machine 2, for the problem of transition to turbulence of the steady flow in a channel. The spectral algorithm used serially requires $O(N^4)$ operations when solving the equations on an $N \times N \times N$ grid; using the massive parallelism of the CM, it becomes an $O(N^2)$ problem. Preliminary timings of the code, written in LISP, are included and compared with a corresponding code optimized for the CRAY-2 for a 128 x 128 x 101 grid.

17. Key Words (Suggested by Author(s)) Parallel Algorithms, SIMD Architectures, Navier-Stokes equations, Spectral Methods	18. Distribution Statement 61 - Computer Programming & Software; 64 - Numerical Analysis; Unclassified - unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	10	A02