JOURNAL OF COMPUTATIONAL PHYSICS 50, 209–214 (1983)

Technical Improvements for Direct Numerical Simulation of Homogeneous Three-Dimensional Turbulence

C. BASDEVANT

Laboratoire de Météorologie Dynamique,
École Normale Supérieure,
24, rue Lhomond, 75231 Paris Cedex 05, France

Received April 28, 1982

Several equivalent forms of incompressible Navier–Stokes equations in three dimensions are presented. The required number of Fourier transforms and input/output passes are discussed for a spectral simulation of periodic flow.

I. INTRODUCTION

The study of homogeneous turbulence via spectral numerical simulations of periodic flows is very popular among specialists. Huge resolutions, up to 128^3 or 256^3 with symmetries, are currently utilized. This paper provides some improvements to the algorithms used at present. In Section II we list four equivalent forms of the Navier–Stokes equations; the fourth one, which is new, leads to more efficient numerical algorithms as discussed in Sections III and IV. In these sections we compare the number of Fourier transforms and transfers between central and peripheral memories for various spectral algorithms. The comparison is made on these numbers rather than on the total operation count or on the effective computing time because first, for large resolutions it is clear that most of the computing time is spent in fast Fourier transforms and/or in input/output passes; and second, total operation count and effective computing time are largely dependent on the programmer's skill as well as on the particular computer used.

II. DIFFERENT FORMS OF NAVIER–STOKES EQUATIONS

The velocity being defined by its components (u, v, w), P being the pressure and ν the kinematic viscosity, the three classical formulations of incompressible Navier–Stokes equations in dimension three are

$$\dot{u} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial P}{\partial x} + \nu\Delta u, \tag{1}$$

0021-9991/83 $3.00

$$\dot{u} + \frac{\partial}{\partial x} u^2 + \frac{\partial}{\partial y} uv + \frac{\partial}{\partial z} uw = -\frac{\partial P}{\partial x} + \nu \Delta u, \tag{2}$$

$$\dot{u} - v\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + w\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right) = -\frac{\partial \Pi}{\partial x} + \nu \Delta u, \tag{3}$$

$$\Pi = P + \tfrac{1}{2}(u^2 + v^2 + w^2).$$

For each we have given only the evolution equation of the first component of the velocity, the incompressibility condition $(\partial u/\partial x) + (\partial v/\partial y) + (\partial w/\partial z) = 0$ being implicit. Formulation (2) is associated with conservation of momentum whereas energy conservation expresses itself simply in form (3).

The following new formulation, yet unpublished, will present advantages over the preceding ones in the derivation of algorithms for spectral simulation methods:

$$\dot{u} + \frac{1}{3}\frac{\partial}{\partial x}(u^2 - v^2) + \frac{1}{3}\frac{\partial}{\partial x}(u^2 - w^2) + \frac{\partial}{\partial y} uv + \frac{\partial}{\partial z} uw = -\frac{\partial \Pi}{\partial x} + \nu \Delta u, \tag{4}$$

$$\Pi = P + \tfrac{1}{3}(u^2 + v^2 + w^2).$$

A similar form is available for incompressible Navier–Stokes equations in dimension two.

III. Number of Fourier Transforms per Time Step

We analyse the evaluation of time derivatives of the velocity Fourier components by means of a collocation method. Examine, for instance, the collocation method associated with formulation (2).

In spectral form, dissipation terms are obtained by simple algebraical operations; likewise, because of incompressibility, the divergence of equation of motion gives, in spectral form, an algebraical relation between the pressure (P or Π) and nonlinear terms; pressure terms are then simply eliminated. The only difficulty is then to calculate nonlinear terms, going from Fourier space to physical space, by means of fast Fourier transforms (FFT), to evaluate pointwise products.

Knowing $(\hat{u}(\mathbf{k}), \hat{v}(\mathbf{k}), \hat{w}(\mathbf{k}))$ the Fourier components of the velocity, in order to evaluate the Fourier components of nonlinear terms of Eqs. (2), we execute successively

$$\begin{matrix} \hat{u} \\ \hat{v} \\ \hat{w} \end{matrix} \xrightarrow{3\text{FFT}^{-1}} \begin{matrix} u \\ v \\ w \end{matrix} \longrightarrow \begin{bmatrix} u^2 & uv \\ v^2 & vw \\ w^2 & wu \end{bmatrix} \xrightarrow{6\text{FFT}} \begin{bmatrix} \widehat{u^2} & \widehat{uv} \\ \widehat{v^2} & \widehat{vw} \\ \widehat{w^2} & \widehat{wu} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} ik_1\widehat{u^2} + ik_2\widehat{uv} + ik_3\widehat{uw} \\ \cdots \\ \cdots \end{bmatrix},$$

where $\hat{f}$ denotes a function described by its Fourier components, and f the same function described by its values on a regular grid, and $\mathbf{k} = (k_1, k_2, k_3)$.

Thus formulation (2) requires 9FFT *per time step*. Similarly one would show that formulation (1) requires 14 FFT per time step (11 inverse, 3 direct), and formulation (3) 9FFT *per time step* (6 inverse, 3 direct).

Formulation (4) has the advantage of requiring only 8FFT *per time step* (3 inverse, 5 direct):

$$\begin{matrix}\hat{u}\\ \hat{v}\\ \hat{w}\end{matrix} \xrightarrow{3\text{FFT}^{-1}} \begin{matrix}u\\ v\\ w\end{matrix} \longrightarrow \begin{bmatrix} u^2-v^2 & uv \\ v^2-w^2 & vw \\ & wu \end{bmatrix} \xrightarrow{5\text{FFT}} \begin{bmatrix} \widehat{u^2-v^2} & \widehat{uv} \\ \widehat{v^2-w^2} & \widehat{vw} \\ & \widehat{wu} \end{bmatrix}$$

the "sixth" term $\widehat{w^2-u^2}$ being obtained trivially from the others.[1]

When using a Fourier–Galerkin method, two possibilities are available for removing aliasing errors: one developed by Patterson and Orszag [1] requires twice the given numbers of FFTs, the other one the same numbers of FFTs as collocation methods but the transforms must be performed on arrays one and a half times larger in each space direction.

IV. Number of Input/Output Passes per Time Step

We consider large resolutions so that a single scalar array is larger than the available memory of the computer. Data fields are then resident on peripheral memory (disks) and processed by input and output of (small) parts of them. Two techniques are available for this processing: a full three-dimensional array can be divided in cubes or in planes.

For present array processors or super computers able to compute FFTs very rapidly, the input/output time becomes predominant and is only partially covered by computing time. It is then of great importance to reduce to the minimum the number of inputs and outputs. We shall examine in that respect these two techniques associated with the algorithms described previously.

The Cube Technique

Any three-dimensional array is kept on disks as a collection of elementary cubes; a fast Fourier transform on such an array is obtained by computing successively the transform in each of the three directions of space, inputting and outputting piles of elementary cubes ranked along the direction of computation (Pouquet, [2]). One FFT on a full scalar array then requires 6 input/output (I/O) passes of the equivalent of a full scalar array:

[1] In two dimensions the number of FFTs per time step is five for currently used methods and four for the similar new formulation we derived.

Start with u written on disk,
input packs of x lines,
FFT along x direction: FFT these lines,
output the results,
same along y from preceding results,
same along z from preceding results,
the result $\hat{u}$ is on disk.

For the centered second order time scheme (leapfrog), we need to compute the time derivative of the velocity field at time t, then input the velocity at time $t - dt$ to compute the velocity at time $t + dt$ and finally write the result on disk:

$$\hat{\mathbf{u}}(t) \rightarrow \begin{cases} \dot{\hat{\mathbf{u}}}(t) \rightarrow & \hat{\mathbf{u}}(t + dt) = \hat{\mathbf{u}}(t - dt) + 2dt\, \dot{\hat{\mathbf{u}}}(t), \\ \hat{\mathbf{u}}(t - dt). \end{cases}$$

Suppose the chosen spectral formulation requires to evaluate the time derivatives q inverse Fourier transforms followed by $(p - q)$ direct transforms, so that the total number of FFT's per time step is p. To avoid useless I/O passes we can chain the calculations of inverse and direct transforms by suitable organization of the calculation sequence: the last output pass and first input pass between batches can be eliminated. Likewise the final result of direct transforms need not be outputed if these transforms are chained with the time integration. For one time step we then need the following number of I/O passes:

$3 + q(6 - 2)$	for the q inverse transforms, starting with the three components of $\hat{\mathbf{u}}$ and ending without outputting the q final results,
$(p - q)(6 - 2)$	for the $(p - q)$ direct transforms, the initial data being already in the computer and the results being kept there,
3	for the input of $\hat{\mathbf{u}}(t - dt)$,
3	for the output of $\hat{\mathbf{u}}(t + dt)$,
total: $4p + 9$	I/O passes of a full scalar array per time step.

So usual collocation methods (formulation (2) or (3)) require, in this technique, 45 I/O *passes per time step.* Formulation (4) we propose needs only 41 I/O *passes per time step.*

We can still reduce this number of I/O passes by using incompressibility which allows us to store on disks only two components of the velocity (e.g., $\hat{u}$ and $\hat{v}$). Keeping in central memory (or on disk) only the "zonal" components of w ($k_3 = 0$), i.e., a two-dimensional array, one can reconstruct, from $\hat{u}$ and $\hat{v}$, the entire vertical velocity. This technique saves three more I/O passes, which leads to 38 I/O *passes per time step.* Finally, we can chain the termination of a time step with the beginning

of the next one, saving the input of the last computed velocity field, and we end up then with 36 I/O *passes per time step.*

The Plane-by-Plane Technique

The plane-by-plane technique, which we are now going to define, can be used only if the central memory is sufficiently large. In this technique a three-dimensional array is inputted by packs of one or several planes parallel to the x and y directions for processing in these directions, and by packs of lines parallel to the z direction for processing in the vertical direction.

One FFT on a full scalar array then requires four I/0 passes of the equivalent of a full scalar array:

Start with u written on disk.

FFT along x and y: input planes parallel to x and y,
FFT these planes,
output the results.

FFT along z: input z lines of the preceding results,
FFT these lines,
output the results.

the result $\hat{u}$ is on disk.

Now for the leapfrog scheme, assuming p FFT's for the computation of the time derivatives, we need, following the same calculations as for the cube technique:

$$3 + q(4-2) + (p-q)(4-2) + 3 + 3 = 2p + 9$$

I/O passes per time step.

So usual collocation methods (formulation (2) or (3)) require in the plane-by-plane technique 27 I/O *passes per time step.* Formulation (4) requires 25 I/O *passes per time step,* with the incompressibility trick previously described: 22 I/O *passes per time step* and with chaining of two time steps: 20 I/O *passes per time step.*

V. Conclusion

As soon as the number of degrees of freedom of a numerical simulation of turbulence is large, input/output passes between peripheral and central memory become very costly. We pointed that this cost is very different from one scheme to the other, varying between 45 and 20 I/O passes per time step.

Acknowledgments

We thank Dr. Philippe Roy for useful comments and an unknown reviewer for pertinent recommendations.

REFERENCES

1. G. S. PATTERSON AND S. A. ORSZAG, *Phys. Fluids* **14** (1971), 2538.
2. A. POUQUET, Note sur la manipulation des données en simulation numérique directe de la turbulence, Observatoire de Nice, France, 1976.