Numerical Turbulence Simulation on Different Parallel Computers Using the Sparse Grid Combination Method

W. Huber

Institut für Informatik, Technische Universität München
Arcisstraße 21, D-80290 München, Germany
e-mail: huberw@informatik.tu-muenchen.de

Abstract. The parallel numerical solution of the Navier-Stokes equations with the sparse grid combination method was studied. This algorithmic concept is based on the independent solution of many problems with reduced size and their linear combination. The algorithm for 3-dimensional problems is described. Its parallel implementation on an IBM SP2 and a cluster of 16 HP workstations is discussed.

1 Introduction

In the area of computational fluid dynamics (CFD), the study of turbulent, 3-dimensional flow of incompressible fluids is one of the most challenging tasks. To understand better the arising phenomena and their interaction with walls and boundaries, driven turbulence[1] in a (periodically closed) pipe or canal is studied frequently as a test problem. Its direct numerical simulation involves the solution of the 3-dimensional Navier-Stokes (NS) equations on extremely fine grids for a long period of observation time to gain a sufficient amount of flow field samples for a statistical evaluation. Thus, any direct numerical simulation (DNS) is very time-consuming and needs, at least for higher Reynolds numbers, an extreme amount of main memory and disk storage which is beyond existing computers. Therefore, the simulation was limited to relatively small Reynolds numbers. In the following, we consider the parallel simulation of a turbulent flow in a pipe for a Reynolds number of 6950.

2 The numerical method and its parallelization

On the discretization of the NS equations a second order FV method on a staggered grid is used. The time discretization is explicit (Euler/Leapfrog) and Chorin's projection method is used. To cope with the pipe geometry, we use cylindrical coordinates.

Instead of the treatment of the NS equations on a conventional full grid, we apply a new technique, the *combination method*, which works on a so-called

[1] in contrary to decaying turbulence, where no walls are present.

sparse grid only. It involves different regular grids $\Omega_{i,j,k}$ with different mesh width $h_i = 2^{-i}, h_j = 2^{-j}, h_k = 2^{-k}$ in r-, φ- and z-direction and the linear combination of the corresponding solution according to the formula

$$u^c_{n,n,n} = \sum_{i+j+k=n+2} u_{i,j,k} - 2 \sum_{i+j+k=n+1} u_{i,j,k} + \sum_{i+j+k=n} u_{i,j,k} \qquad (1)$$

gives a numerical solution on the sparse grid $\Omega^s_{n,n,n}$, see Fig. 1.

On each of the grids involved, the Navier-Stokes equations are treated explicitly for T successive time steps *independently* of each other. Then, a sparse grid solution is assembled by the linear combination process (1) and, from that, the starting iterates for the next cycle of T time steps on the different grids are gained by projection, and so on. For the treatment of the iterates on these different grids, any existing NS-Solver can be applied directly (see Fig. 2).

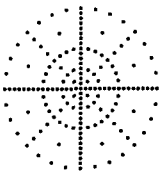

Fig. 1. The sparse grid $\Omega^s_{4,4,4}$, cross-section.

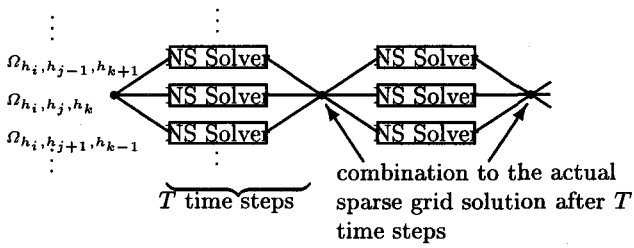

Fig. 2. Integration of the combination method into a NS solver.

Thus, our method involves substantially less grid points (i.e. $O(N \cdot (\log N)^2)$ with $N = 2^n+1$) instead of $O(N^3)$ for the full grid discretization but the accuracy of the obtained solutions is nearly the same (see [1, 2]) as for the conventional approach and is, for turbulent flow applications, within the range of the error of physical measurements anyway. To cope with the possibly extreme distorsion on some of the grids involved, we modified (1) in a way that certain grids are omitted and the combination of the solution iterates of only m grids takes place, see also [1, 3]. On results of the statistical evaluation for the combination method and the DNS, see [1, 3].

Furthermore, this method is perfectly suited for parallelization on a relatively coarse grain level (see [1, 3]) and allowing for a simple load balancing strategy. Thus, in each time step, the overall complexity of the parallelized combination method is reduced to $O(N)$ in comparison to $O(N^3)$ for the conventional approach in the sequential setting. To achieve this, $O((\log N)^2)$ processors are necessary.[2] [3]

[2] Note that, in the parallelized version of the combination method, the sparse grid is never assembled explicitly, but its relevant data are stored in a *distributed* way on the different grids which are associated to different processors.

[3] Of course, it is possible to further parallelize the work for each subproblem arising in the combination method e.g. by conventional domain decomposition techniques.

3 Run time results

For different numbers of grids m, Table 1 shows the fraction of the amount of memory necessary to store the data belonging to *one*, i.e. the largest grid versus the amount of memory necessary to store the data belonging to the full grid. In brackets we note the reduction rate in percent.

Table 1. Requirements for distributed storing.

Number m of grids involved in the combination method	Storage requirements in comparison with the full grid
1	1 (100.0%)
4	1/4 (25.0%)
10	1/16 (6.25%)
19	1/64 (1.56%)
31	1/256 (0.39%)

We implemented the parallel version of our combination method on an IBM SP2 and a cluster of 16 HP 9000/720 workstations. On the IBM machine we use PVM 3.3.7 and PVMe 3.2. On the workstations we use PVM 3.2.7.

The number of grids involved in the combination process were $m = 4, 10, 19, 31$ on the workstations and $m = 19, 31$ on the IBM machine.

The *total speedup* of an algorithm on parallel machines can be decomposed into the *parallel speedup* and the *numerical speedup*. The parallel speedup is defined by

$$S_{par}(P) := \frac{\text{execution time of the combination method on 1 node}}{\text{execution time of the combination method on P nodes}}. \quad (2)$$

Here, P defines the number of nodes (that means processors or workstations). The numerical speedup is defined as

$$S_{num} := \frac{\text{execution time of the full grid version on 1 node}}{\text{execution time of the combination method on 1 node}}. \quad (3)$$

Parallel speedup: First, we like to concentrate ourself on the parallel speedup of our implementation. Using PVMe it is not possible to run more than one process on one node. Hence, for $m = 19$ grids we have to use 19 nodes and for $m = 31$ grids we have to use 31 nodes. Dealing with PVM it is possible to run more than one process on one node. On the different architectures, we obtained the parallel speedup with PVM shown in Fig. 3 and 4. The overall speedup increases very slowly on increasing the number of nodes.[4]

[4] The non optimized version of our code is one reason for the poor speedup. Another reason is the storage requirement (see Table 1) and so the computing time arising in the combination process.

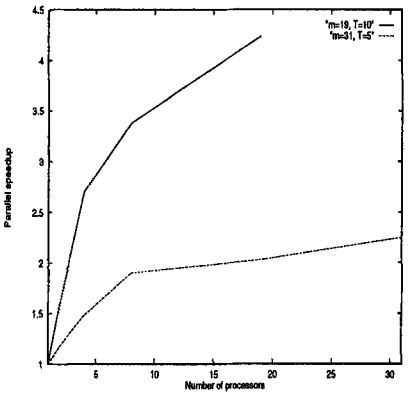

Fig. 3. Parallel speedup on P processors using PVM 3.3.7 on the SP2.

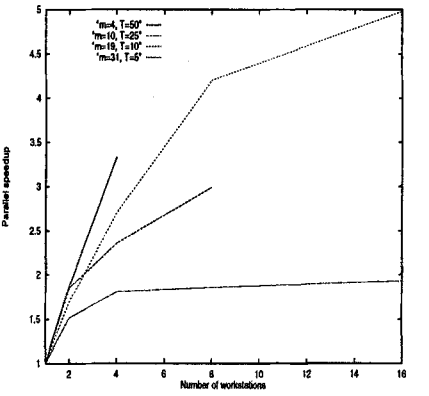

Fig. 4. Parallel speedup on P workstations using PVM 3.2.7.

Numerical speedup: Now, we turn to the numerical speedup of our combination method. Here, we compare the run times of the combination method on a sparse grid with the run times of the same NS solver on the associated full grid. It can be seen from Table 2 that our new method has a better numerical efficiency than the standard approach on the full grid.

Table 2. Numerical speedup.

architecture	$m = 4$ $T = 50$	$m = 10$ $T = 25$	$m = 19$ $T = 10$	$m = 31$ $T = 5$
max. value	1.1	2.1	4.7	11.9
HP workstations	0.8	2.1	3.3	7.3
IBM SP2			3.8	6.5

References

1. M. Griebel, W. Huber, and C. Zenger, *Turbulence simulation on sparse grids using the combination method*, in Notes on Numerical Fluid Mechanics, E. H. Hirschel, ed., vol. 52, Vieweg Verlag, 1996, pp. 34–47.
2. M. Griebel, M. Schneider, and C. Zenger, *A combination technique for the solution of sparse grid problems*, in Iterative Methods in Linear Algebra, P. de Groen and R. Beauwens, ed., IMACS, Elsevier, North Holland, 1992, pp. 263–281.
3. W. Huber, *Turbulenzsimulation mit der Kombinationsmethode auf Workstation-Netzen und Parallelrechnern*, Dissertation, Institut für Informatik, TU München, 1996.