ECCOMAS Computational Fluid Dynamics Conference 2001
Swansea, Wales, UK, 4-7 September 2001

PARALLEL COMPUTATIONAL FLUID DYNAMICS AT DELFT UNIVERSITY; SOME EXAMPLES

P. Wilders[1], B.J. Boersma[2], J.J. Derksen[3], A.W. Heemink[1], B. Niceno[4], M. Pourquie[2], C. Vuik[5]

Delft University of Technology, J.M. Burgers Centre
Leeghwaterstraat 21, 2628 CJ Delft, The Netherlands,
email: p.wilders@its.tudelft.nl

[1]Dept. Applied Mathematical Analysis, Section Large Scale Systems
[2]Dept. Mechanical Engineering, Section Fluid Mechanics
[3]Dept. Applied Physics, Kramers Laboratorium
[4]Dept. Applied Physics, Section Thermofluids
[5]Dept. Applied Mathematical Analysis, Section Numerical Mathematics

Key words: parallel computing, parallel computational fluid dynamics, large eddy, LES, direct numerical simulation, DNS, mixing and stirring, tracer flow, environmental flow, data assimilation, Kalman filtering,

Abstract. *At Delft University of Technology much research is done in the area of computational fluid dynamics with underlying models ranging from simple desktop-engineering models to advanced research-oriented models. The advanced models have the tendency to grow beyond the limit of single-processor computing. In the last few years research groups, studying such models, have extended their activities towards parallel computational fluid dynamics on distributed memory machines. We present several examples of this, including fundamental studies in the field of turbulence, LES modelling with industrial background and environmental studies for civil engineering purposes.*

1 INTRODUCTION

We present several examples of applications involving parallel computational fluid dynamics on distributed memory machines, such as found at Delft University of Technology. For choosing the examples we have considered all contributions from this university to the yearly ParCFD conference meetings, a major conference in the field. In 2001, the 13th ParCFD conference was hosted and organized by Delft University of Technology in collaboration with the J.M. Burgers Centre and the National Aerospace Laboratory NLR (http://www.parcfd.org/). This strategy for choosing the examples results in a good overview of ongoing work at the Delft University of Technology. The groups under consideration are all members of the J.M. Burgers Centre and, as such, this overview is illustrative for the ongoing work within this centre as well.

At Delft University of Technology parallel computational fluid dynamics is an ongoing research activity within several research groups. Typically, this research is set up and hosted within departments. For this purpose they use centrally supported facilities, most often only operational facilities. In rare cases, central support is given as well for developing purposes. Central support is provided by HPαC, http://www.hpac.tudelft.nl/, an institution for high performance computing splitted off from the general computing center in 1996. Their main platform is a Cray T3E with 128 DEC-Alpha processors, installed in 1997 and upgraded in 1999.

From the paralllel point of view most of the work is based upon explicit parallel programming using message passing interfaces. The usage of high-level parallel supporting tools is not very common at our university. Only time accurate codes are studied with time stepping procedures ranging from fully explicit to fully implicit. Typically, the explicit codes show a good parallel performance, are favorite in engineering applications and have been correlated with measurements using fine-grid 3D computations with millions of grid points. The more implicit oriented codes are still in the stage of development, can be classified as research-oriented codes using specialized computational linear algebra for medium size grids and show a reasonable parallel performance.

The physical background of the parallel CFD codes is related to the individual research themes. Traditionally, Delft University of Technology is most active in the incompressible or low-speed compressible flow regions. Typically, Delft University is also active in the field of civil engineering, including environmental questions. The present overview reflects both specialisms.

Of course, studying turbulence is an important issue. Direct numerical simulation (DNS) and large eddy simulation (LES) are used, based upon higher order difference methods or Lattice-Boltzmann methods, both to study fundamental questions as well as applied questions, such as mixing properties or sound generation. Parallel distributed computing enables to resolve the smallest turbulent scales with moderate turn-around times. In particular, the DNS codes are real number crunchers with excessive requirements.

A major task in many CFD codes is to solve large linear systems efficiently on parallel platforms. As an example, we mention the pressure correction equation in a non-Cartesian incompressible code. In Delft, Krylov subspace methods combined with domain decomposition are among the most popular methods for solving large linear systems. Besides applying these methods in our implicit codes, separate mathematical model studies are undertaken as well with the objective to improve robustness, convergence speed and parallel performance.

At the level of civil engineering, contaminant transport forms a source of inspiration. Both atmospheric transport as well as transport in surface and subsurface regions is studied. In the latter case the number of contaminants is in general low and there is a need to increase the geometrical flexibility and spatial resolution of the models. For this purpose parallel transport solvers based upon domain decomposition are studied. In the atmospheric transport models the number of contaminants is high and the grids are regular and of medium size. However, in this case a striking feature is the large uncertainty. One way to deal with this latter aspect is to explore the numerous measurements for improvement of the predictions. For this purpose parallel Kalman filtering techniques are used in combination with parallel transport solvers.

We will present various details encountered in the separate studies and discuss the role of parallel computing, quoting some typical parallel aspects and results. The emphasis will be more on showing where parallel CFD is used for and how this is done than on discussing parallel CFD as a research object on its own.

2 Turbulence

Turbulence research forms a major source of inspiration for parallel computing. Of all activities taking place at Delft University we want to mention two, both in the field of incompressible flow.

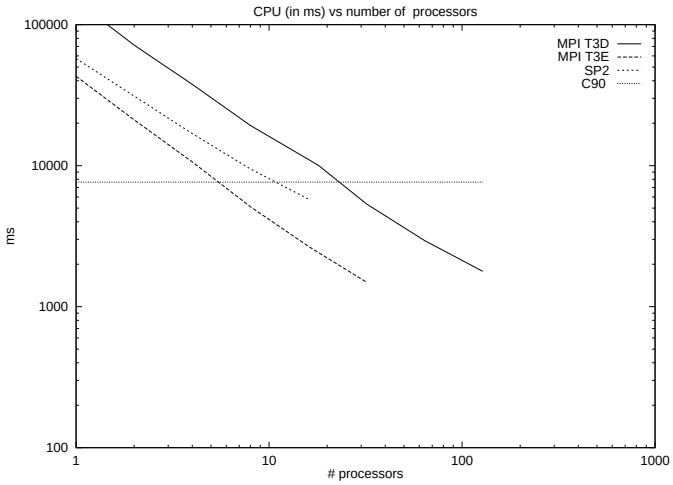

Figure 1: Wall clock time for 64^3 model problem.

A research oriented code has been developed in [9], [10]. Both DNS and LES methods are investigated and compared. The code explores staggered second-order finite differencing on Cartesian grids and the pressure correction method with an explicit Adams-Bathford or Runge-Kutta method for time stepping. The pressure Poisson equation is solved directly using the Fast Fourier transform in two spatial directions, leaving a tridiagonal system in the third spatial direction. The parallel MPI-based implementation relies upon the usual ghost-cell type communication, enabling the computation of fluxes, etc., as well as upon a more global communication operation, supporting the Poisson solver. For a parallel implementation of the Fast Fourier transform it suffices to distribute the frequencies over the processors. However, when doing a transform along a grid line all data associated with this line must be present on the processor. This means that switching to the second spatial direction introduces the necessity of a global exchange of data. Of course, the final tridiagonal system is parallelized by distributing the lines in the associated spatial direction over the processors. Despite the need of global communication, the communication overhead remains in general below 10%. Figure 1 gives an example of the measured wall clock time. The speed-up in nearly linear.

In figure 2 a grid type configuration at inflow generates a number of turbulent jet flows in a channel (modelling wind tunnel turbulence). Due to the intensive interaction and mixing, the distribution of turbulence becomes very quickly homogeneous in the lateral direction. A way to access the numerical results, see figure 3, is to compute the Kolmogorov length scales (involving sensitive derivatives of flow quantities). The grid size is 600 x 48 x 48 (1.5 million points), which is reported to be sufficient to resolve all scales in the mixing region with DNS for $R = 1000$. For $R = 4000$ subgrid LES modelling is needed: measured subgrid contributions are of the order of 10 %.

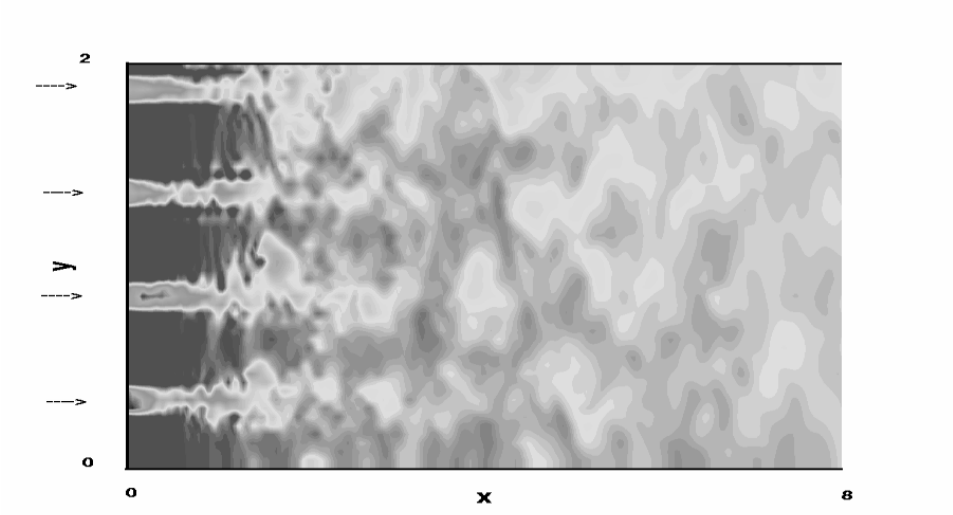

Figure 2: Contour plot of the instantaneous velocity for a flow behind a grid.

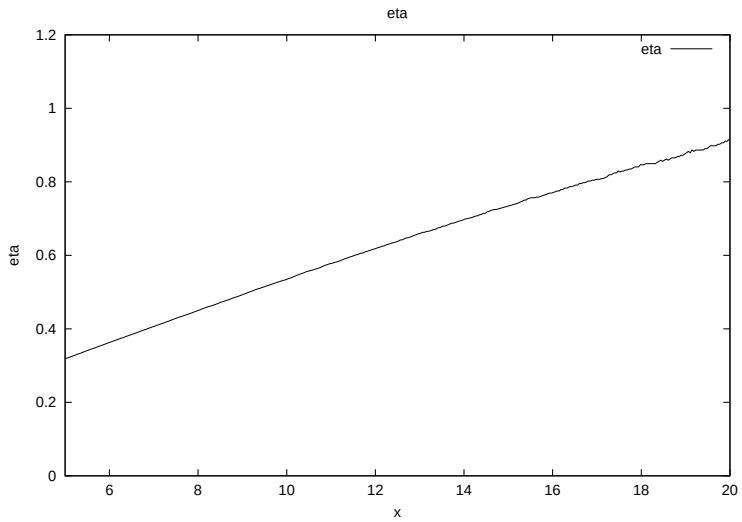

Figure 3: Kolmogorov scale.

A second example of turbulence modelling can be found in [8]. In this study the goals are directed towards industrial applications with complex geometries using LES. Unstructured co-located second-order finite volumes, slightly stabilized, are used in combination with the pressure correction method and implicit time stepping. Solving the linear systems is done with diagonally preconditioned Krylov methods, i.e. CGS for the pressure equation and BiCG for the momentum equations. The computational domain is split into subdomains, which are spread over the processors. Because diagonal preconditioning is used, it suffices to implement a parallel version of the Krylov method, which is done in a straightforward standard manner. As before, ghost-cell type communication enables the computation of fluxes, matrices, etc. As is well-known some global communication of inner products is necessary in a straightforward parallel implementation of Krylov methods.

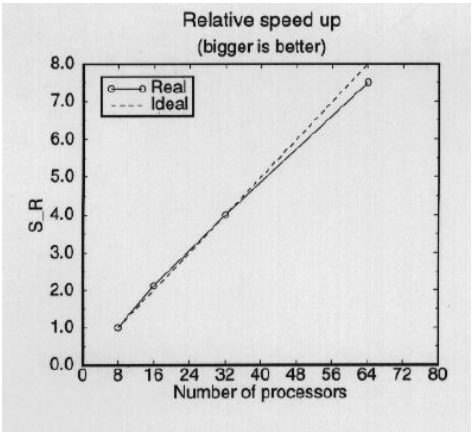

Figure 4: Speed-up on Cray T3E.

Figures 4 present some typical parallel performance results of this code. A nearly perfect speed-up is obtained. Here, the total number of grid points is around 400,000 and the number of subdomains is equal to the number of processors. Thus for 64 processors there are approximately 7000 grid points in each subdomain, being sufficient to keep the communication to computation ratio low. The code is memory intensive. In fact, on a 128 MB Cray T3E node the user has effectively access to 80 MB (50 MB is consumed by the system) and the maximal number of grid points in a subdomain is bounded by approximately 50,000. This explains why the graph in figure 4 starts off at 8 processors. LES results for the flow around a cube at $R = 13000$ are presented in figure 5. The results were obtained with 32 processors of the Cray T3E, running approximately 3 days doing 50,000 time steps.

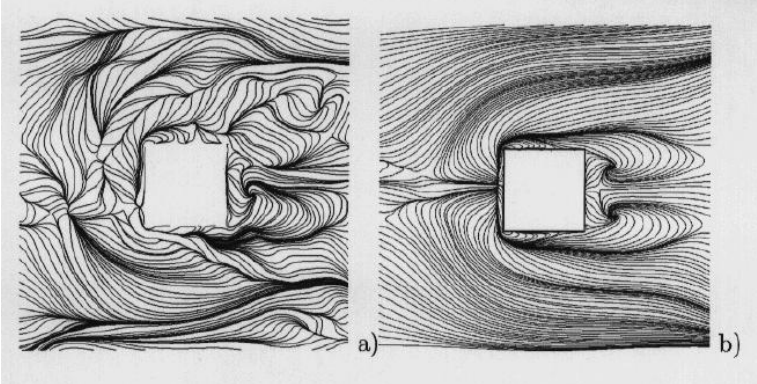

Figure 5: Streamlines: a) instantaneous, b) averaged.

3 Sound generation by turbulent jets

It is well-known that turbulent jets may produce noise over long distances. Studying the flow properties of a round turbulent jet has been done in [1],[7]. As a follow-up the sound generation by a low Mach number round turbulent jet at $R = 5000$ has been investigated in [2]. For low Mach numbers the acoustic amplitudes are small and a reasonable approximation results from using Lighthill's perturbation equation for the acoustic density fluctuation $\varrho' = \varrho - \varrho_0$, which amounts to a second-order wave equation driven by the turbulent stresses via the source term $T_{ij,i,j}$, involving the Lighthill stress tensor T_{ij}. The equation is written as a system of two first-order equations and treated numerically by the same techniques used for predicting the jet. Typically, the acoustic disturbances propagate over longer distances than flow disturbances and the domain, on which Lighthill's equation is solved, is taken a factor of two larger in each spatial direction. Outside the flow domain Lighthill's equation reduces to an ordinary wave equation because the source term is set to zero. From figure 6 it can be seen that this is a suitable approach.

DNS computations are done using millions of grid point on non-uniform Cartesian grids.

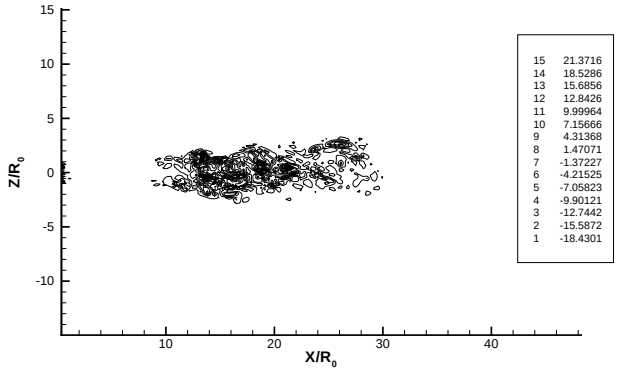

Figure 6: Magnitude of source term $T_{ij,i,j}$ in Lighthill's wave equation.

A sixth-order compact co-located differencing scheme is used in combination with fourth-order Runge-Kutta time stepping. In a compact differencing scheme not only the variables itself but also their derivatives are propagated. This introduces some specific parallel features with global communication patterns using the MPI routine MPI_ALLTOALL. With respect to communication protocols, this code behaves quite similar to the first code described in the previous section. Also here, communication overhead remains below 10%. Figures 7 and 8 present some results of the computation. Shortly after mixing the jet starts to decay. The quantity $Q = \partial\varrho'/\partial t$ is a measure of the frequency of the sound. We can see distinct spherical waves originating from the point where the core of the jet collapses. The computations are quite intensive and the present results have been obtained on a SGI-Origin 3000 machine, located at the national computing center SARA (http://www.sara.nl/), using 16 processors.

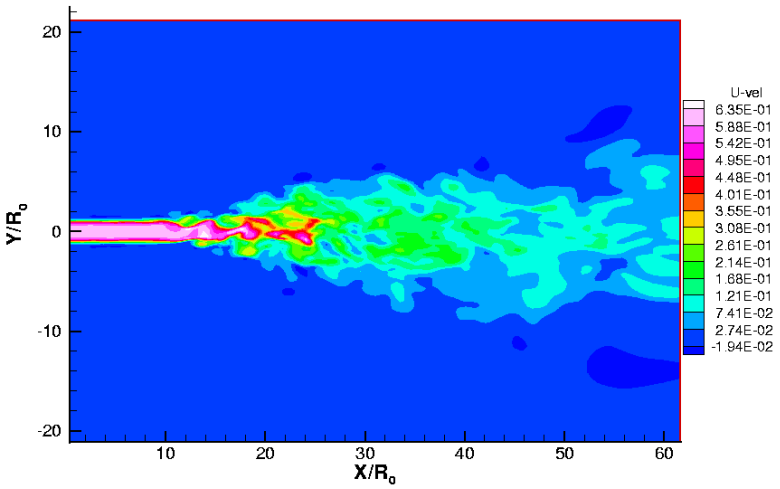

Figure 7: Contour plot of velocity.

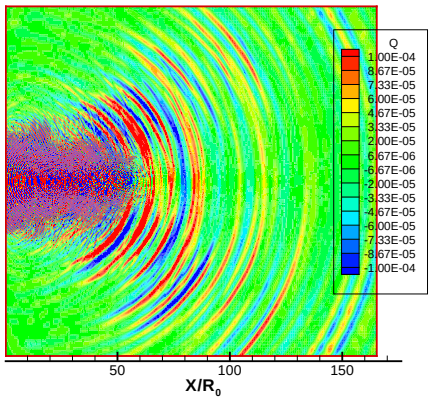

Figure 8: Contour plot of Q, measuring the frequency.

4 Stirring and mixing

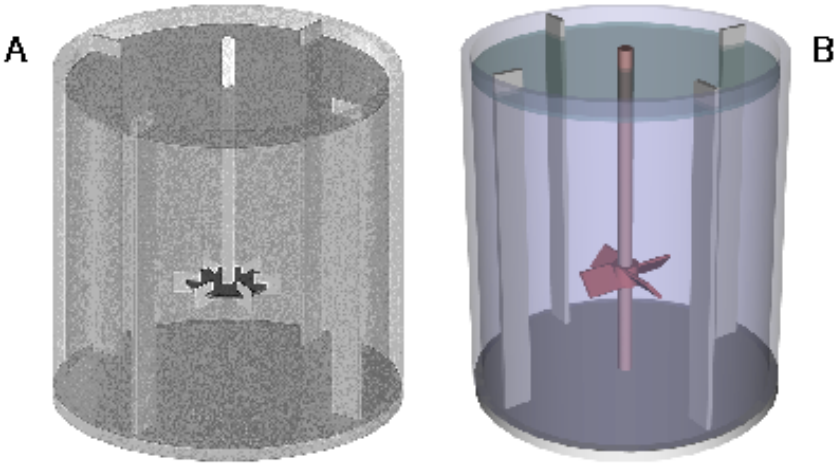

Figure 9: A: disk turbine, B: pitched blade turbine.

Stirring in tanks is a basic operation in chemical process industries. Mixing properties depend strongly upon turbulence generated by the impeller. LES modelling of two basic configurations at $R = 29000$, respectively $R = 7300$, see figure 9, has been done in [4], [3] using the Lattice-Boltzmann approach. This approach resembles an explicit time stepping approach in the sense that the total amount of work depends linearly upon the number

of nodes of the lattice. In the specific scheme employed [5] the solution vector contains, apart from 18 velocity directions, also the stresses. This facilitates the incorporation of the subgrid-scale model.

Here, parallelization is rather straightforward. The nodes of the lattice are distributed over the processors and only nearest neighbour communication is necessary. In order to enhance possible usage by industry (affordable computing), a local Beowulf cluster of 12 processors and a 100Base TX fast Ethernet switch has been build with MPICH for message passing. On this cluster the code runs with almost linear speed-up, solving problems up to 50 million nodes, taking 2 days per single impeller revolution. Here, it is worthwhile to notice that the impeller is viewed as a force-field acting on the fluid. Via a control algorithm the distribution of forces is iteratively led towards a flow field taking the prescribed velocity on the impeller, typically taking a few iterations per time step (< 5).

Most important for stirring operations are the average flow (average over impeller revolutions) and the turbulence generated on its way. It has been found that the average flow is well predicted, see figure 10. However, the turbulence is overpredicted, see figure 11. This is contributed to a lack of precision in the LES methodology.

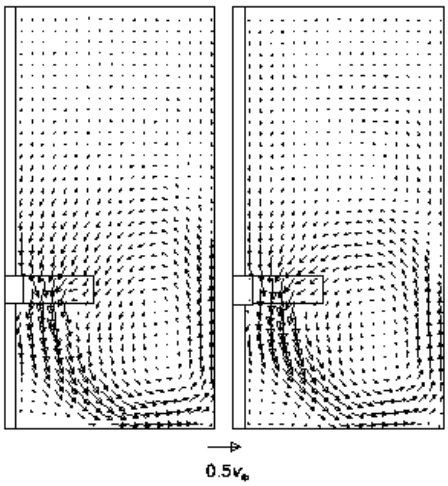

Figure 10: Pitched blade turbine. Average velocity. Left: LDA experiment. Right: LES on 360^3 grid.

5 Tracer Transport

Tracer studies in surface and subsurface environmental problems form a basic ingredient in environmental modelling. From the application viewpoint, there is a need to resolve large scale computational models with fine grids, for example to study local behaviour in coastal areas with a complex bathymetry/geometry or to study fingering due to strong inhomogeneity of the porous medium. A research oriented code has been developed in

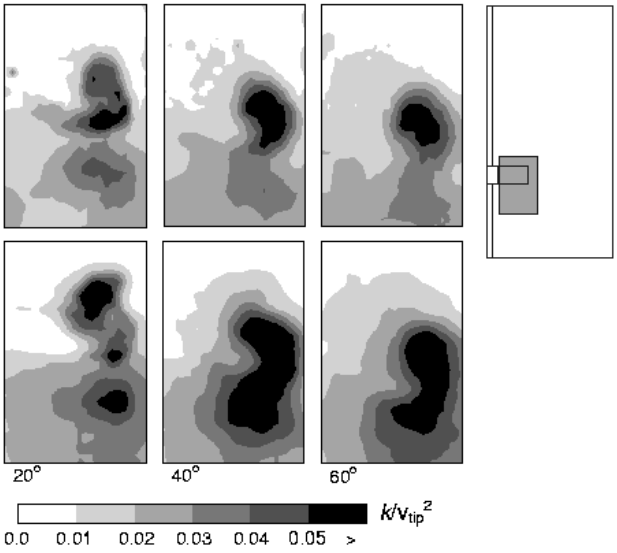

Figure 11: Pitched blade turbine. Contour plot of turbulent kinetic energy near the blade. Top row: experiment. Bottom row: LES on 240^3 grid.

[13], [14], [15]. Unstructured cell-centered finite volumes are implemented in combination with implicit time stepping and GMRES-accelerated domain decomposition. The parallel implementation is MPI-based and explores a master-slave communication protocol, see figure 12. The GMRES master process gathers the ghost cell variables, updates them, and scatters them back. Figure figure 13 presents the (relative) costs for a linearly growing problem size, such as measured on an IBM-SP2. It has been found that less than 15% of the overhead is due to communication and sequential operations in the master. The remaining overhead is caused by load imbalance as a consequence of variations in the number of inner iterations (for subdomain inversion) over the subdomains. It is in particular this latter point that hinders full scalability, i.e. to speak in terms of [6], the synchronization costs in code with iterative procedures are difficult to control for large number of processors.

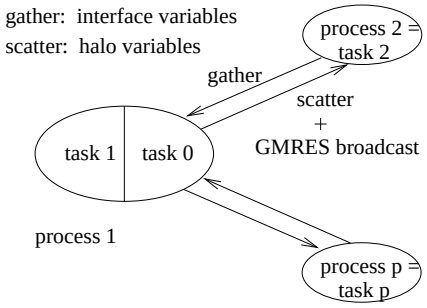

Figure 12: Communication patterns

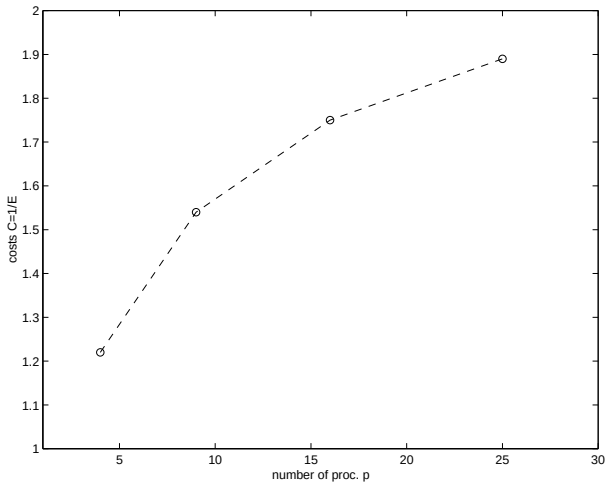

Figure 13: The costs C_p for a linearly growing problem size.

Typically, the code is applied off-line using precomputed and stored flow data. Figure 14 presents an injection/production-type tracer flow in a strongly heterogenous porous medium. A large gradient profile is moving from the lower left corner to the upper right corner. Figure 15 results from a release (near Rotterdam) study in a Dutch coastal region. Breakthrough - and arrival times are important civil parameters. It has been observed that arrival times in coastal applications are sometimes sensitive for numerical procedures.

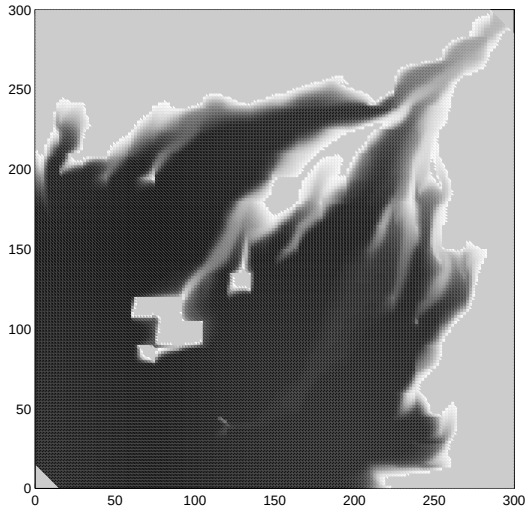

Figure 14: Concentration at 0.6 PVI.

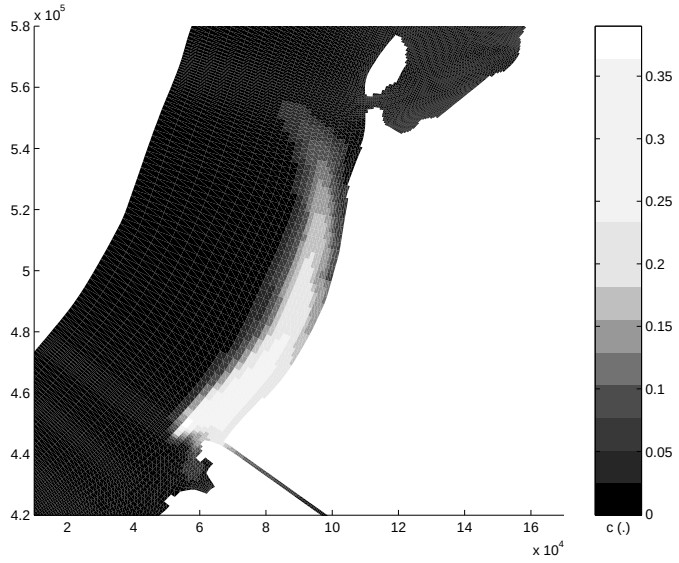

Figure 15: Concentration after 72 tidal periods.

6 Data assimilation

The idea behind data assimilation is to use observations to improve numerical predictions. Observations are fed on-line into a running simulation. First, a preliminary state is computed using the plain physical model. Next, this state is adapted for better matching the observations and for this purpose Kalman filtering techniques are often

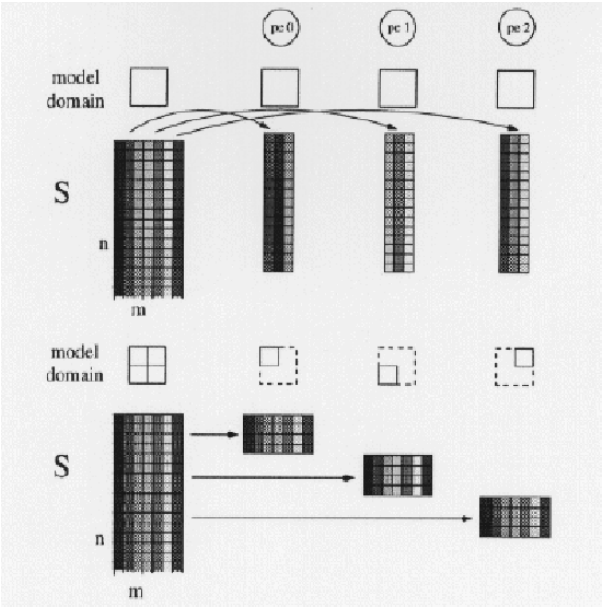

Figure 16: Decomposition: over the modes (columnwise) or over the domain (rowwise).

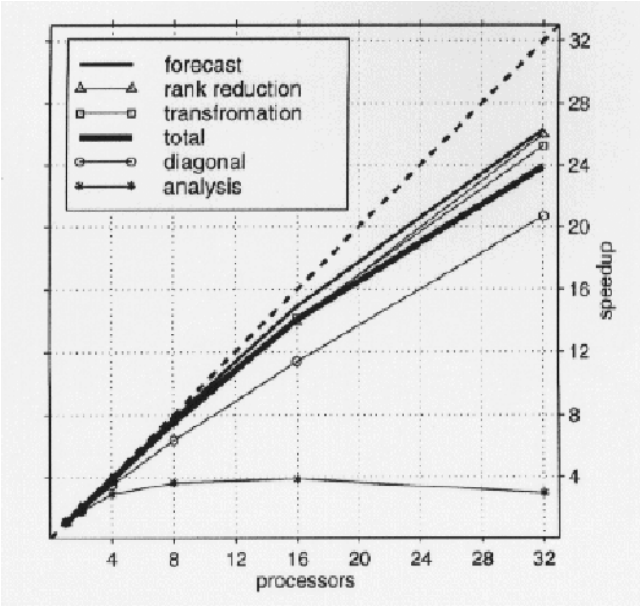

Figure 17: Speed-up for the mode decomposed filter.

employed. This approach has been followed in [11] for the atmospheric transport model LOTOS (Long Term Ozone Simulation) for a region covering the main part of Europe, with $n = 90,000$ the total number of unknowns (26 species).

The n x n covariance matrix P contains the covariance of uncertainties in the grid points and is a basic ingredient. Since P is too large to handle, approximations are introduced via

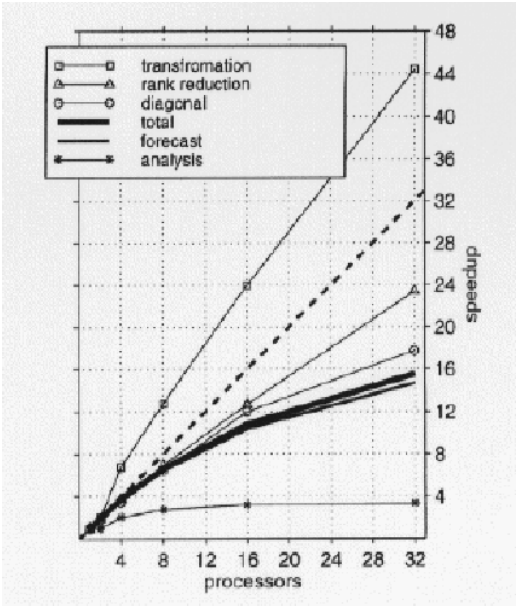

Figure 18: Speed-up for the domain decomposed filter.

a reduced rank formulation, in the present study the RRSQRT approximation (reduced rank square root) [12]. P is factorized ($P = SS'$), using the n x m low-rank approximation S of its square root. For obtaining the entries of S the underlying LOTOS model has to be executed m times, computing the response for m different modes (called the forecast below). In the present study values of m up to 100 have been used.

An obvious way for parallelization is to spread the modes over the processors, running the full LOTOS model on each processor. In a second approach spatial domain decomposition is used to spread the LOTOS model over the processors, see figure 16. Figure 17 and 18 present some performance results, such as obtained on the Cray T3E. Besides the forecast, several other small tasks (involving numerical linear algebra) have to be performed. However, their influence on the final parallel performance remains small, because only a fraction of the total computing time is spent here ($< 20\%$ in a serial run). For the problem under consideration the mode decomposition performs better. However, the problem size has been chosen in such a way that it fits into the memory of a single Cray T3E node (80 MB, see earlier), leading to a small problem size. For larger problem sizes it is expected that the situation turns in favour of the domain decomposed filter. Firstly, because the communication patterns show less global communication. Secondly, because of memory bounds. It shall be clear that scaling up with mode decomposition is more difficult in this respect, because the full physical model has to reside on each processor.

7 Conclusions and final remarks

Parallel computational fluid dynamics is on its way to become a basic tool in engineering sciences at least at Delft University of Technology. The broadness of the examples given by us illustrates this. We have also tried to outline the directions in which developments take place. Computations with millions of unknowns over moderate time intervals are nearly a day-to-day practice with some of the more explicit-oriented codes. Tools have been developed for post-processing the enormous amounts of data. For approaches, relying upon advanced numerical linear algebra and/or flexible finite volume methods, much remains to be done in order to scale up.

We have seen that the parallel computational fluid dynamics research in Delft takes place in diverse groups and that this research is often initiated by the need to extend the traditionally studied models towards large scale models. Only in the mathematics department there is some interest in parallel computational fluid dynamics as a discipline on its own. A consequence is that the parallel CFD community is scattered around the university with little interaction between the groups. Although this situation seems to be inherent in university communities, it seems to be worthwhile to investigate whether some mutual interaction pays off.

REFERENCES

[1] B.J. Boersma, G. Brethouwer and F.T.M. Nieuwstadt, A numerical investigation on the effect of the inflow conditions on the self-similar region of a round jet, *Physics of Fluids*, **10**, pages 899-909, 1998.

[2] B.J. Boersma, Direct numerical simulation of jet noise, In *Parallel Computational Fluid Dynamics 2001*, Egmond aan Zee, The Netherlands, May 21-23 2001.

[3] J. Derksen and H.E.A. van den Akker, Large eddy simulations on the flow driven by a Rushton turbine, *AIChE Journal*, **45**, pages 209-221, 1999.

[4] J. Derksen, Large eddy simulation of agitated flow systems based on lattice-Boltzmann discretization, In C.B. Jenssen et al., editors, *Parallel Computational Fluid Dynamics 2000*, pages 425-432, Trondheim, Norway, May 22-25 2000, Elsevier 2001.

[5] J.G.M. Eggels and J.A. Somers, Numerical simulation of free convective flow using the Lattice-Boltzmann scheme, *Int. J. Heat and Fluid Flow*, **16**, page 357, 1995.

[6] D. Keyes, private communication at *Parallel Computational Fluid Dynamics 2001*, Egmond aan Zee, The Netherlands, May 21-23 2001.

[7] C.L. Lubbers, G. Brethouwer and B.J. Boersma, Simulation of the mixing of a passive scalar in a free round turbulent jet, *Fluid Dynamic Research*, **28**, pages 189-208, 2001.

[8] B. Niceno and K. Hanjalic, Large eddy simulation on distributed memory parallel computers using an unstrucured finite volume solver, In C.B. Jenssen et al., editors, *Parallel Computational Fluid Dynamics 2000*, pages 457-464, Trondheim, Norway, May 22-25 2000, Elsevier 2001.

[9] M. Pourquie, B.J. Boersma and F.T.M. Nieuwstadt, About some performance issues that occur when porting LES/DNS codes from vector machines to parallel platforms, In D.R. Emerson et al., editors, *Parallel Computational Fluid Dynamics 1997*, pages 431-438, Manchester, UK, May 19-21 1997, Elsevier 1998.

[10] M. Pourquie, C. Moulinec and A. van Dijk, A numerical wind tunnel experiment, In *LES of complex transitional and turbulent flows, EUROMECH Colloquium Nr. 412* , München, Germany, October 4-6 2000.

[11] A.J. Segers and A.W. Heemink, Parallization of a large scale Kalman filter: comparison between mode and domain decomposition, In *Parallel Computational Fluid Dynamics 2001*, Egmond aan Zee, The Netherlands, May 21-23 2001.

[12] M. Verlaan and A.W. Heemink, Tidal forecasting using reduced rank square root filters, *Stochastic Hydrology and Hydraulics*, **11**, pages 349-368, 1997.

[13] P. Wilders, Parallel performance of domain decomposition based transport, In D.R. Emerson et al., editors, *Parallel Computational Fluid Dynamics 1997*, pages 447-456, Manchester, UK, May 19-21 1997, Elsevier 1998.

[14] C. Vittoli, P. Wilders, M. Manzini and G. Fotia, Distributed parallel computation of 2D miscible transport with multi-domain implicit time integration, *J. Simulation Practice and Theory*, **6**, pages 71-88, 1998.

[15] P. Wilders, Parallel performance of an implicit advection-diffusion solver, In D. Keyes et al., editors, *Parallel Computational Fluid Dynamics 1999*, pages 439-446, Williamsburg, Virginia, USA, May 23-26 1999, Elsevier 2000.