Mixing Simulations: Tracking Strongly Deforming Fluid Volumes in 3D Flows

A.S. Galaktionov1,2, P.D. Anderson1, G.W.M. Peters1

1 Centre for Polymers and Composites, Faculty of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
2 Institute of Hydromechanics of NAS of Ukraine, Zhelyabov Str. 8/4, 252057, Kiev, Ukraine

Abstract. An efficient and accurate technique to trace strongly deforming fluid volumes in 3-dimensional flows is presented. This algorithm significantly benefits from parallelization (data decomposition). Results of numerical simulations are presented, that are illustrative for the use of parallel computations for studying fluid mixing phenomena.

1 Introduction

The availability of the relatively low-cost tools for extensive numerical simulations significantly facilitates the research in many branches of science, and, among them, in fluid mechanics. One of the fast developing, important subjects of fluid mechanics is fluid mixing phenomena. Numerical simulation of these phenomena is computationally expensive and demands for special computational techniques such as parallel processing.

Such numerical simulations provide data for understanding of processes occurring in real mixing devices. One of the objectives of current work was to develop and test the set of tools for the tracking of strongly deforming fluid volumes in 3D flows.

2 3D flow simulations

The problem that has to be solved is the advection of passive tracers (markers) used for describing a strongly deforming fluid volume in a flow (distributive mixing). This involves the tracking of a large, varying amount of markers. It is this tracking that makes parallel processing attractive because for every point these computations are independent. These material points are placed in the flow domain and tracked by integrating the differential equations of motion. Computing the velocity field (especially for 3D flows) requires significant resources, especially in RAM size (although this also can be parallelized to certain extent), and preferably should be performed on high-performance workstations. Unlike this, the integration of the equations of motion, using a pre-computed velocity field, is a computationally extensive task, but typically requires less resources.
Moreover, in the case of the advection of a passive tracers (when their presence does not change the rheological properties of the fluid), the motion of each particular marker does not depend on the motion of all other markers. This gives a natural and straightforward way for parallelization of the computational task (PDE integration) by distributing it over a variety of hosts in a local network. At the same time some control decisions, involving information about all markers should be performed by a special master programme. Since the amount of data passed to and from slave processes is moderate, the usage of PVM [3] is very beneficial for such tasks. In the simplest case of a quasi-stationary flow, the velocity field can be computed in advance. For time-dependent transient flow, the velocity field needs to be updated during the simulations. In this last case the computation of the velocity field can be assigned to the most powerful workstation, from which results are broadcasted to other processes, while the raw work of the integration of the equations of motion is assigned to other hosts of the virtual machine.

PVM gives an excellent opportunity to use frequently underestimated computational power of a local network. Even the simulation of mixing in 3D transient flows can be efficiently performed using a parallel virtual machine, containing only one host with relatively large RAM size and a (large) number of other computers available at the same time.

3 Description of a deforming volume

An example of 3D distributive mixing simulation is presented. A blob, with an initially spherical shape is placed in the flow domain. The evolution in time of the shape of the blob gives information on the mixing properties of the flow. If, for example, the flow has a high mixing efficiency, a blob can be stretched exponentially in time. Since the blob is assumed to have the same mechanical properties as the rest of the fluid, a new shape of the blob can be determined by tracking only its surface.

For this purpose, the surface is covered with a grid of points (nodes), forming a mesh. The deformed mesh is refined when necessary, taking into account the changing size of the mesh cells and surface curvature. Different types of meshes were tested for this purpose. Structured meshes turned to be of limited usefulness. They are simple in use and post-processing but have the significant disadvantage of not permitting local refinement. Unstructured meshes, composed of triangles are much more useful in this case. An example of an unstructured mesh, covering a sphere, is shown in figure 1. When the blob is deformed, relatively flat parts of the surface can be covered by large mesh cells, while in strongly curved regions the size of the mesh cells should be sufficiently smaller. That is why curvature analysis is used to reduce the amount of computational work. To evaluate the curvature, the angles between normals to the adjacent cells and the size of the mesh cells are used.

For transient flows it is hardly possible to keep a complete record of the flow field evolution. Instead, the data from a few, last time steps are kept and used.
cers (when their presence is underestimated), the motion of each pair marker. This gives the computational task a local network. At nation about all markers of data usage of PVM [3] is very stationary flow, the ve flow, the ions. In this last case the most powerful worksta-sses, while the raw work ed to other hosts of the

Fig. 1. Unstructured mesh, covering a sphere.

Fig. 2. 4-step walls motion protocol in a cubic cavity.

Then, when mesh refinement is necessary, decisions are made using the data from the last time level, but actual insertion of new nodal points is performed at the earliest time level still available. The new nodes are then tracked to the current time level, using the known history of the flow. If necessary, this refinement cycle is repeated. The advantage of such approach is a reasonable accurate representation of the surface (the mesh is refined at the earlier time level where it was less deformed) without storing the complete history (coordinates of markers and velocity field) of the flow and deforming blob.

After the refinement is completed, local transformations of the mesh topology (re-connection of the nodes) is performed to improve the quality of the mesh (highly stretched cells are avoided in the mesh, if possible). These transformations are performed only in less curved zones of the surface, where they don’t affect the surface shape significantly. The “quality” of a mesh cell is characterized with

\[ q = 12\sqrt{3} \frac{s}{p^2}, \]

where \( s \) is the area and \( p \) is the perimeter of the triangle. The value \( q = 1 \) represents a perfect triangle and \( 0 < q < 1 \) otherwise.

4 Implementation of the parallel algorithm

For the implementation of the parallel algorithm a master-slave [3] scheme was chosen. The master programme stores, saves on disk and retrieves the mesh data, performs all kinds of mesh transformations and distributes the extensive computational work (differential equations integration) among slaves. One more auxiliary programme is developed to handle the velocity field data, broadcasting them to slaves on request of the master programme. In case of transient flows this programme should actually compute the velocity field at each new time level. In case of quasi-steady flow (which is presented as an example here) it
only handles the choice of time steps between mesh refinements and data saving
and controls the order of the storage of data from different time levels.

Within each time step, the main programme requests for a velocity update
and next distributes all markers over the slaves. After tracking is completed
the mesh refinement is done by the master programme and the task of advection
of the new inserted markers is distributed over the slaves. This step is repeated
until all mesh refinement conditions are fulfilled. The workload is distributed
dynamically [3]: a limited amount of markers is assigned to each available slave
processes and then, each slave which returns updated data, gets a new assign-
ment, until the pool of markers to move is empty. This way all slave processes
are kept busy until the requested work is done.

5 Results

We examine the time-periodic 3D mixing flow of viscous incompressible fluid
in a cubic cavity. Flow is considered in a Stokes approximation where inertia
terms are neglected. This flow is a 3D generalization of a 2D lid-driven cavity
flow, extensively used in the both theoretical and experimental study of fluid
mixing mechanisms (see, for example [2], [4], [5], [6]). The flow is induced by
the successive motion of two opposite (front and back) walls in perpendicular
directions. This 4-step wall motion protocol is illustrated in figure 2. The system
under study is very convenient as a prototype mixing flow: a relatively simple
geometry yields accurate numerical solutions for the velocity field (which is very
important for studying the mixing flows), but the fluid motion is essentially 3-
dimensional. The results presented here correspond to the wall displacements of
2.5 times the cube edge length.

The key item for evaluation of the mixing ability of time-periodic flow is the
presence and type of periodic points (that is, points which return to their original
position after a number of periods of motion) [5], [6]. In 3D flows, periodic lines
composed of periodic points can be formed. A numerical technique was used that
gives the possibility to find and diagnose the type of periodic structures. This
technique (its complete description is beyond the scope of current presentation)
involves the analysis of the total displacement of material particles over one
period of the flow and the analysis of the local deformation. A system of periodic
lines (period-1) determined for the flow under investigation is shown in figure 3.

For testing purposes the initial blob was centered around part of a hyperbolic
periodic line, where the material stretching ratio over 1 period exceeds 10 (see
figure 4). This ensures that the test volume will be strongly deformed in the flow
under study. The diameter of the initial blob is equal to 10% of the cavity size.

The motion of the blob over 3 complete periods of the flow was simulated.
The resulting shape of the blob is shown in figures 5-7. The area of the blob
surface increases more than 50 times. Whereas the initial shape was described
with 258 markers, about $10^4$ markers were necessary to represent the blob shape
after 3 periods. The maximum allowed size of the mesh cells in “flat” zones of
the surface was set equal to the initial radius of the blob, while the size of the
nements and data saving. The velocity update
strategy is completed the task of advection this step is repeated. The workload is distributed
to each available slave data, gets a new assign-
ses way all slave processes

ous incompressible fluid
ximation where inertia
a 2D lid-driven cavity
ment study of fluid
The flow is induced by
walls in perpendicular
d in figure 2. The system
flow: a relatively simple
ocity field (which is very
motion is essentially 3-
the wall displacements of

time-periodic flow is the
return to their original
3D flows, periodic lines
 technique was used that
periodic structures. This
of current presentation)
terial particles over one
A system of periodic
ition is shown in figure 3.
und part of a hyperbolic
1 period exceeds 10 (see
ugly deformed in the flow
0% of the cavity size.
the flow was simulated.
7. The area of the blob
tial shape was described
represents the blob shape
cells in “flat” zones of
ob, while the size of the

Fig. 3. Periodic lines, found in the core of the flow under study (10% of the cavity size away from the walls. Thin lines on the plot denote hyperbolic periodic lines.

elements in curved zones is twice smaller. Estimations show that, to achieve the same quality of surface shape description by means of specifying an initial very fine mesh and not performing mesh refinement and connectivity transformation, would require about $5 \times 10^8$ markers.

Fig. 4. Initial blob. Periodic lines are also shown.

Fig. 5. Blob after 1 period of motion.

Computational cost of these simulations is sometimes difficult to evaluate, because typically no hosts were assigned exclusively for these computations. Nevertheless, the particular case, illustrated in figures 4-7, was computed with 4 slave processes on a SGI Power Challenge ($4 \times 196$ MHZ R10000 processors) within 6 hours. The master programme, not performing any extensive computa-
Fig. 6. Blob after 2 periods of motion. Fig. 7. Blob after 3 periods of motion.

...tions, was assigned to a Pentium Pro PC, running under Linux. During this run, the data on the blob shape was saved 100 times a period for creating animated visualization. Analogous computations were performed for different initial blob locations. The results were combined and post-processed using AVS [1] to create an animation of the advection of the blobs.

Problems, involving heavily massive computations can strongly benefit from parallelization, but in the case that the amount of computational work is strongly varying with time (caused by nature of the problem itself), the efficiency may significantly reduce. This was evaluated with two test problems using a different amount of hosts. The virtual machine consisted of identical Pentium Pro PCs, connected by a 100 Mbit/s local network and running under Linux 2.07.

<table>
<thead>
<tr>
<th>Number of hosts</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t1/Nt</td>
<td>t</td>
<td>t1/Nt</td>
<td>t</td>
<td>t1/Nt</td>
<td>t</td>
</tr>
<tr>
<td>“test 1”</td>
<td>1223.100</td>
<td>685.0.89</td>
<td>326.0.75</td>
<td>205.0.60</td>
<td>144.0.43</td>
<td>117.0.21</td>
</tr>
<tr>
<td>“test 2”</td>
<td>1330.100</td>
<td>720.0.92</td>
<td>342.0.78</td>
<td>206.0.65</td>
<td>140.0.48</td>
<td>100.0.27</td>
</tr>
</tbody>
</table>

Table 1. Dependence of the duration of computations t (in minutes) for two test problems on the number of hosts. Total elapsed time, including start-up, loading and saving data (not just CPU time!) is shown. The ratio t1/Nt, where t1 is the elapsed time for the same job executed on 1 host, gives the evaluation of hosts usage efficiency.

The first test problem ("test 1") consisted in tracking of the same blob as shown in figures 4-7 during 2 periods. The number of markers increased during tracking from 258 in the beginning to 25670 after 2 periods. In the second test problem ("test 2") the blob, which was previously advected for 2 periods using a moderately fine mesh, was tracked back to the initial position, using a reversed motion protocol. As the programme was not allowed to throw away any
of the already existing markers, their number raised during tracking from 9992 to 15421.

The results of these tests are summarized in the table 1. Although the computational time reduces with adding more hosts, the efficiency of their use declines. The reason for this is that during some particular stages of tracking (for example, when tracking new markers after mesh refinement) the amount of available work is not enough to use all hosts efficiently and sometimes even not enough to distribute to all of them. The tendency, illustrated by table 1 for this class of problems, should be taken into account.

6 Conclusions

It is concluded that the use of parallel algorithms, based on PVM, is highly beneficial for the discussed class of fluid dynamics problems. Parallelization significantly reduces in this case the duration of computations (although not the sum of consumed CPU times). This gives the freedom to handle different sets of data etc. within the same limited time. This is of great importance for future work, when the mixing of two rheologically different fluids and break-up of a fluid domain will be investigated (and thus the number of initial domains will increase in time). Dynamical distribution of workload gives a possibility to use efficiently the resources of a local network, where the load of hosts is varying with time. Slave processes perform simple operations with limited amount of data and require a moderate amount of RAM. Due to this, the temporarily idle slave processes don’t need to be canceled in the case of “bottle necks” in the simulations. These arise when, as a result of rapidly converging mesh refinement processes, the available pool of tasks is not enough to distribute them over all slave processes.

The authors would like to acknowledge support by the Dutch Foundation of Technology (STW), grant no. ETW44.3453. We also gratefully acknowledge the work done by system engineers P.E.J. van Brakel and L.H.G. Wouters.

References