INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **41**:1253–1271 (DOI: 10.1002/fld.423)

Numerical simulation of disperse multiphase flows with an application in power engineering

K. Bernert, Th. Frank, H. Schneider and K. Pachler*,†

Chemnitz University of Technology, Faculty of Mechanical Engineering and Process Technology, Research Group of Multiphase Flow, Reichenhainer Straße 70, 09107 Chemnitz, Germany

SUMMARY

This paper deals with the numerical simulation of two-phase flows based on the solution of the Navier–Stokes equations with a $k-\varepsilon$ turbulence model for the gas phase and a particle tracking model of the disperse phase fulfilling the framework of the Eulerian–Lagrangian (PSI-Cell) approach. The numerical procedures for the two phases are based on the domain decomposition method applied to a block-structured grid. The complete code is parallelized for computers of MIMD architecture. The paper gives a description of the numerical methods with special attention to the parallelization. Some test calculations demonstrate the performance of the code. The numerical simulation of a flow splitter from the field of power engineering is presented as an example for a real world application of the method.

KEY WORDS: multiphase flows; numerical simulation; Eulerian–Lagrangian approach; parallelization; particle transport in power engineering

1. INTRODUCTION

Disperse multiphase flows are very common for processes in mechanical and thermal process technology (e.g. gas–particle or gas–droplet flows, coal combustion, pneumatical conveying, erosion phenomena). Furthermore processes for the separation of solid particles from gases or fluids and for the classification and particle size analysis are an important field of interest in process technology.

The numerical simulation of multiphase flows includes both the calculation of the continuous phase and the calculation of a high number of particle traces as a basis for deriving statistical quantities as particle concentration, mean particle velocity, etc. As a first step the continuous phase can be calculated independently of the disperse phase, later the interaction with the disperse phase is to be included in the right hand side of the equations of motion in an iterative way. For the disperse phase the authors apply the Lagrangian (PSI-Cell) approach,

* Correspondence to: K. Pachler, Fakultät für Maschinenbau und Verfahrenstechnik, Reichenhainer Strasse 70, 09107 Chemnitz, Germany.
† E-mail: klaus.pachler@arcor.de

Received 3 March 2002
Revised 13 August 2002

i.e. discrete particle trajectories are calculated. Each calculated particle represents a large number of physical particles with the same physical properties.

Even a unique calculation of the continuous phase followed by the calculation of the disperse phase can be very time-consuming. More than ever the iterative coupling of the two phases demands clearly the use of parallel computation. In contrast to a strong activity in developing efficient parallel codes for flow calculation there are only few publications on particle tracking by the Lagrangian method on parallel computer systems. All methods known to the authors suffer from a restricted parallelism by using the shared memory concept [1] respectively copying the complete flow field to all processors of the parallel machine [2, 3] or from the risk of a low parallel efficiency on massive parallel computers of MIMD architecture, resulting from a naive static domain decomposition [4].

Section 2 gives a short description of the physical and mathematical background for calculating the fluid and particle phases.

Sections 3 and 4 explain the solving methods and parallelization strategies. For the calculation of the disperse phase two parallelization strategies are presented. The first algorithm applies a static assignment of grid partitions to the processors of the parallel machine (SDD—static domain decomposition). As our results show, the parallel efficiency of such a parallelization method can be dramatically deteriorated. In the second parallelization method—the so-called dynamic domain decomposition (DDD)—a dynamic assignment of grid and fluid flow information to processor nodes is used, leading to a considerably increased parallel efficiency and a higher degree of flexibility in the application of the computational method to different flow conditions.

Section 5 presents numerical experiments concerning the performance of the code and is focused on parallel efficiency. Most of the calculations have been performed on parts of the Chemnitz Linux Cluster (CLiC) consisting of 528 Intel Pentium III 800 MHz processors and a FastEthernet network.

In Section 6 the direct Eulerian–Lagrangian approach is applied to a flow in the field of power engineering. The appliance which is investigated numerically is a flow splitter with complex interior guiding vanes called bifurcator. It is used for pneumatical transport of coal particles and the split of the overall coal particle mass flow rate from the coal mills to the burners of a coal-fired power plant.

The results show an efficient operation of our code and give a deeper understanding of the flow structure in the bifurcator.

2. PHYSICAL AND MATHEMATICAL FUNDAMENTALS

2.1. Basic equations of fluid motion

The fluid phase considered here is assumed to be Newtonian and to have constant physical properties. The fluid flow is three-dimensional, steady, incompressible, turbulent and isothermal. Fluid turbulence is modelled using the standard $k-\varepsilon$ model and neglecting the influence of particle motion on fluid turbulence. Under these assumptions the time-averaged equations describing the motion of the fluid phase are given by the following form of the general transport equation:

$$\frac{\partial}{\partial x_j}(\rho_{\mathrm{F}} u_j^{\mathrm{F}} \Phi) - \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \Phi}{\partial x_j}\right) = S_{\mathrm{F}} + S_{\mathrm{P}} \tag{1}$$

Table I. Flow variables, transport coefficients and source terms for the basic equations.

Equation for	Φ	Γ	S_F	S_P
Continuity	1	0	0	0
Momentum	u_i^F	μ_{eff}	$\frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial u_j^F}{\partial x_i}\right) - \frac{\partial p}{\partial x_i} + \rho_F f_i$	$S_{u_i}^P$
Turbulent kinetic energy	k	$\frac{\mu_t}{\sigma_k}$	$P_k - \rho_F \varepsilon$	0
Dissipation of k	ε	$\frac{\mu_t}{\sigma_\varepsilon}$	$\frac{\varepsilon}{k}(c_{\varepsilon_1} P_k - c_{\varepsilon_2} \rho_F \varepsilon)$	0

$P_k = \mu_t \frac{\partial u_i^F}{\partial x_j}\left(\frac{\partial u_i^F}{\partial x_j} + \frac{\partial u_j^F}{\partial x_i}\right)$; $\mu_{eff} = \mu + \mu_t$, $\mu_t = \rho_F c_\mu \frac{k^2}{\varepsilon}$

$c_\mu = 0.09$, $c_{\varepsilon_1} = 1.44$, $c_{\varepsilon_2} = 1.92$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$

Here Φ is a general variable, Γ a diffusion coefficient, S_F a general source term and S_P the source term due to momentum exchange between the fluid and the particle phase. The variables u_1^F, u_2^F and u_3^F represent the fluid velocity components, k is the turbulent kinetic energy and ε the rate of dissipation of k. Generally index F indicates *Fluid* and P indicates *Particle*. A detailed description of all terms and their correlations is shown in Table I. In this table ρ_F is the fluid density and μ is the laminar viscosity.

2.2. Equations of motion of the disperse phase

The disperse phase is treated by the application of the Lagrangian (PSI-Cell) approach, i.e. discrete particle trajectories are calculated. Each calculated particle represents a large number of physical particles of the same physical properties. This is achieved by a particle number flow rate $\dot{N}_P$ prescribed to each calculated trajectory. The prediction of the particle trajectories is carried out by solving the ordinary differential equations for the particle location and velocities. Assuming that the ratio of fluid density to particle density is small ($\rho_F/\rho_P \ll 1$) these equations read

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_P = \boldsymbol{u}_P \tag{2}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_P = \frac{3}{4}\frac{\rho_F}{(\rho_P + \frac{1}{2}\rho_F)\,d_P}\Bigg(&u_{rel} C_D(Re_P)(\boldsymbol{u}_F - \boldsymbol{u}_P) \\ &+ \frac{u_{rel}}{\omega_{rel}} C_M(\sigma)(\boldsymbol{u}_F - \boldsymbol{u}_P) \times (\omega - \Omega) \\ &+ \frac{2\nu^{1/2}}{\pi|\Omega|^{1/2}} C_A(\boldsymbol{u}_F - \boldsymbol{u}_P) \times \Omega\Bigg) + \frac{\rho_P - \rho_F}{\rho_P + \frac{1}{2}\rho_F}\boldsymbol{g}\end{aligned} \tag{3}$$

with

$$\Omega = \mathrm{rot}\,\boldsymbol{u}_F, \quad Re_P = \frac{d_P u_{rel}}{\nu}, \quad Re_\omega = \frac{1}{4}\frac{d_P^2 \omega_{rel}}{\nu}$$

$$\sigma = \frac{1}{2}\frac{d_P \omega_{rel}}{u_{rel}}, \quad u_{rel} = |\boldsymbol{u}_F - \boldsymbol{u}_P|, \quad \omega_{rel} = |\omega - \Omega|$$

where the rotation of the particle can be calculated from the following equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\,\omega = -\frac{15}{16\pi}\,\frac{\rho_\mathrm{F}}{\rho_\mathrm{P}}\,\omega_\mathrm{rel}\,\xi_\mathrm{m}(Re_\omega)(\omega - \Omega) \tag{4}$$

In these equations v is the fluid kinematic viscosity, d_P the particle diameter and ω_rel the absolute value of the relative rotational velocity between fluid and particle. The terms on the right hand side of (3) represent the drag force exerted on the particle by the fluid, the lift force due to particle rotation (Magnus force), the lift force due to fluid velocity shear (Saffman force), the gravitational and added mass forces, respectively. The values for the coefficients C_D, C_A, C_M and ξ_m can be found in References [5, 6]. Inter-particle collisions are described by collision probability models due to Oesterle [7, 8]. Particle–wall interaction was modeled according to the virtual-wall model by Sommerfeld [9] in the modified wall roughness formulation of Frank *et al.* [10]. The effect of fluid turbulence on the motion of the disperse phase is modelled by the Lagrangian stochastic-deterministic (LSD) turbulence model. The particle's influence on the fluid phase is modelled by the PSI-Cell (particle-source-in-cell) method proposed by Crowe *et al.* [13]. A more detailed description of all particular models involved in the Lagrangian particle trajectory calculation can be found in References [6, 11–13].

3. SOLUTION ALGORITHM

For the numerical solution of the equations described in the preceding section the physical space has to be discretized. Therefore a boundary-fitted, non-orthogonal numerical grid is used. The grid is block-structured and consists of hexahedral cells. The equations of fluid motion (1) are numerically solved on the basis of a collocated finite volume discretization. The SIMPLE method with convergence acceleration by the multigrid technique method is applied, see Reference [14]. When a converged solution for the fluid flow field has been calculated, the prediction of the particle motion is carried out. Therefore Equations (3) and (4) are solved using a standard 4th order Runge–Kutta scheme. In case of two-way-coupled multiphase flow systems the source terms S_P according to the PSI-Cell method are predicted simultaneously during trajectory calculation. After all particle trajectories are calculated the source terms are included in the fluid momentum equations and a new solution for the fluid flow field is computed. In the case of neglectable phase interaction (the so-called one-way coupling) a single iteration step is sufficient to obtain the solution for the fluid and particle motion.

The iterative algorithm for the numerical simulation of the coupled two-phase flow can be summarized as follows:

1. calculation of a first solution for the fluid flow field without taking the source terms of the disperse phase S_P into account,
2. tracing a large number of particles through the flow field and computing the source terms S_P simultaneously,
3. recalculation of the fluid flow field including the source terms S_P of the disperse phase,
4. repeating Steps 2 and 3 until the solution of the coupled equations has converged.

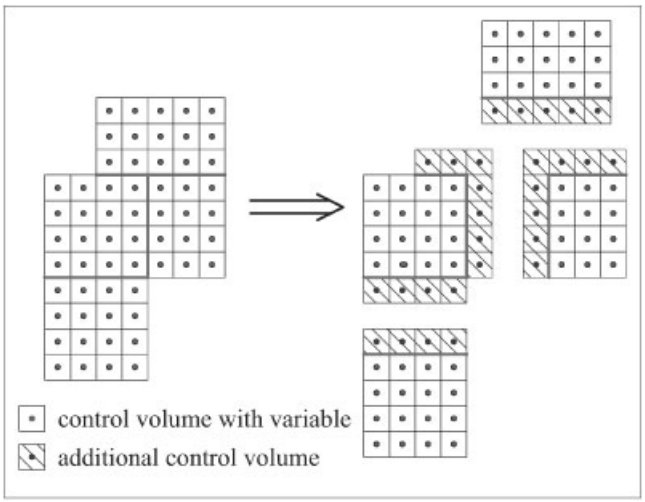

Figure 1. Domain decomposition for the numerical grid.

4. PARALLELIZATION METHODS

4.1. Parallel algorithm for fluid flow calculation

The parallelization of the solution algorithm for the set of continuity, Navier–Stokes and turbulence model equations is carried out by the domain decomposition or grid partitioning method. Using the block structure of the numerical grid the flow domain is partitioned into a number of subdomains (Figure 1). Usually the number of grid blocks exceeds the number of processors, so that each processor of the parallel machine (PM) has to handle a few blocks. The grid-block-to-processor assignment is given by a heuristically determined block–processor allocation table and remains static and unchanged over the time of fluid flow calculation process. Fluid flow calculation is then performed by individual processor nodes on the grid partitions stored in their local memory. Flow characteristics along the grid block boundaries which are common to two different nodes have to be exchanged during the solution process by inter-processor communication, while the data exchange on common faces of neighbouring grid partitions assigned to the same processor node can be handled locally in memory.

4.2. Parallel algorithms for the Lagrangian approach

The prediction of the motion of the disperse phase is carried out by the application of the Lagrangian approach as described in Section 2.2. In the following we consider two parallelization strategies. The static domain decomposition (SDD) method is the most obvious and simplest one. It calculates all trajectory segments at the processors, where the fluid flow data are available from the flow calculation step. The method is easy to implement but can lead to poor load balancing, because the effort for calculating particle trajectories in general is not uniformly distributed in the flow domain even if there is a uniform particle distribution at the inlet. An efficient parallelization of Lagrangian solution algorithm has to take into consideration that the distribution of the work load is not known *a priori.*

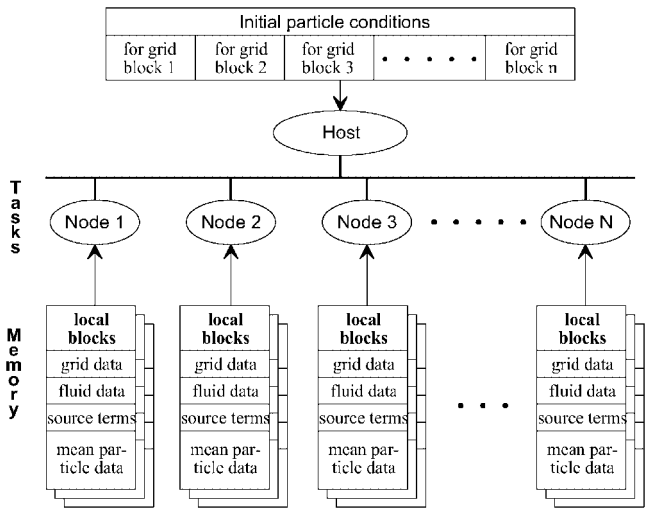

Figure 2. Static domain decomposition (SDD) method for the Lagrangian solver.

The dynamic domain decomposition (DDD) method points out how this can be achieved by a more flexible concept for calculating particle trajectory segments including grid and flow data provision.

Method 1: static domain decomposition method. The first approach in parallelization of Lagrangian particle trajectory calculations is the application of the same parallelization scheme as for the fluid flow calculation to the Lagrangian solver as well. In this approach geometry and fluid flow data are distributed over the processor nodes of the parallel machine (PM) in accordance with the block–processor allocation table as already used in the fluid flow field calculation of the Navier–Stokes solver. Furthermore an explicit host-node process scheme is established as illustrated in Figure 2. The trajectory calculation is done by the node processes whereas the host process carries out only management tasks. The node processes are identical to those that do the flow field calculation. Now the basic principle of the SDD method is that in a node process only those trajectory segments are calculated that cross the grid partition(s) assigned to this process. The particle state (location, velocity, diameter, etc.) at the entry point to the current grid partition is sent by the host to the node process. The entry point can either be at an inflow cross section or at a common face/boundary to a neighboring partition. After the computation of the trajectory segment on the current grid partition is finished, the particle state at the exit point (outlet cross section or partition boundary) is sent back to the host. If the exit point is located at the interface of two grid partitions, the host sends the particle state to the process related to the neighboring grid partition for continuing trajectory computation. This redistribution of particle state conditions is repeatedly carried out by the host until all particle trajectories have satisfied certain break condition (e.g. an outlet cross section is reached). During the particle trajectory calculation process the source terms for momentum exchange between the two phases are calculated locally on the processor nodes $1,\ldots,N$ from where they can be passed to the Navier–Stokes solver without further processing.

Poor load balancing can be a serious disadvantage of the SDD method, as shown later for the presented test cases. Reasons for this behavior can be

1. Unequal processing power of the calculating nodes, e.g. in a heterogeneous workstation cluster.
2. Differences in particle concentration distribution throughout the flow domain. Situations of poor load balancing can occur e.g. for flows around free jets/nozzles, in recirculating or highly separated flows where most of the numerical effort has to be performed by a small subset of all processor nodes used.
3. Multiple particle–wall collisions. Highly frequent particle–wall collisions occur especially on curved walls where the particles are brought in contact with the wall by the fluid flow multiple times. This results in a higher work load for the corresponding processor node due to the reduction of the integration time step and the extra effort for detection and calculation of the particle–wall collision itself.
4. Flow regions of high fluid velocity gradients/small fluid turbulence time scale. This leads to a reduction of the integration time step for the Lagrangian approach in order to preserve accuracy of the calculation and therefore to a higher work load for the corresponding processor node.

Method 2: dynamic domain decomposition method. This method has been developed to overcome the disadvantages of the SDD method concerning the balancing of the computational work load. In the DDD method there exist three classes of processes: the host, the servicing nodes and the calculating nodes (Figure 3). Just as in the SDD method the host process distributes the particle initial conditions among the calculating nodes and collects the particle's state when the trajectory segment calculation has been finished. The new class of servicing nodes uses the already known block–processor assignment table from the Navier–Stokes solver for storage of grid and fluid flow data. But in contrast to the SDD method they do not perform trajectory calculations but delegate that task to the class of calculating nodes. So the work of the servicing nodes is restricted to the management of the geometry, fluid flow and particle flow data in the data structure prescribed by the block–processor assignment table. On request a servicing node is able to retrieve or store data from/to the grid partition data structure stored in its local memory.

The calculating nodes are performing the real work on particle trajectory calculation. These nodes receive the particle initial conditions from the host and predict particle motion on an arbitrary grid partition. In contrast to the SDD method there is no fixed block–processor assignment table for the calculating nodes. Starting with an empty memory structure the calculating nodes are able to obtain dynamically geometry and fluid flow data for an arbitrary grid partition from the corresponding servicing node managing this part of the numerical grid. The correlation between the required data and the corresponding servicing node can be looked up from the block–processor assignment table. Once geometry and fluid flow data for a certain grid partition has been retrieved by the calculating node, this information is locally stored in a pipeline with a history of a certain depth, which can be limited by an adjustable parameter. It has further to be mentioned that a servicing node process does not have to be executed on a separate physical processor, since the work load is quite neglectable. In current MPI implementations the servicing node process is implemented as separate node process and is executed in parallel to the corresponding calculating node process on the same physical processor. Furthermore the host process is also executed on one of the N processors of the

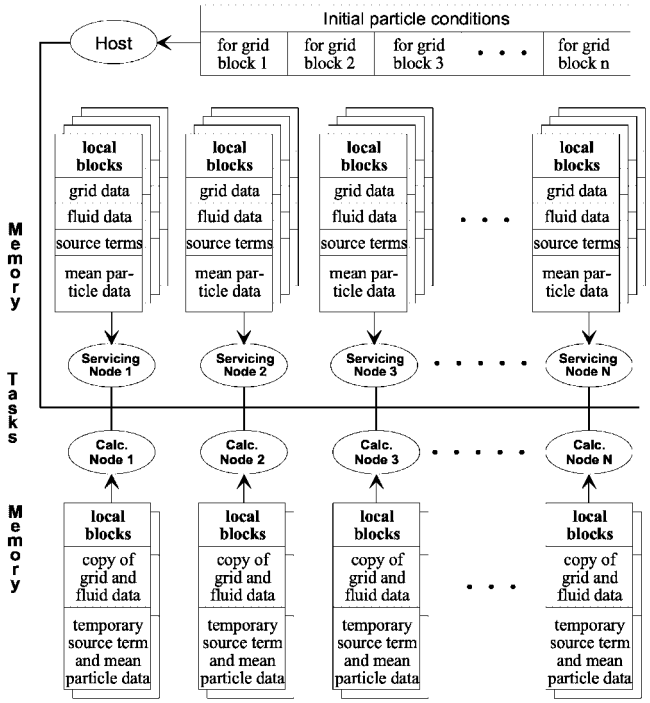

Figure 3. Dynamic domain decomposition (DDD) method for the Lagrangian solver.

PM keeping the number of used processors constant in comparison with the Navier–Stokes solver.

5. NUMERICAL EXPERIMENTS

5.1. MIMD computer architectures and MPI implementations

The different parallelization methods are based on the paradigm of a MIMD computer architecture with explicit message passing between the node processes of the PM. The implementation uses an encapsulated communication layer which can operate on top of standard MPI or PVM communication libraries (the latter from historical reasons). Usable communication libraries have to be in compliance with MPI 1.1 or PVM 3.2 standard.

Investigations presented in this paper were carried out on three different computer architectures. Most of the calculations have been performed on an AMD/Athlon PC cluster or on the Chemnitz Linux Cluster CLiC. For performance comparison we used the CRAY-T3E at the Dresden University of Technology. Table II summarizes the most important properties of the three computer systems. The calculations on the PC clusters were performed with MPI distributions of MPICH 1.2.0 and LAM-MPI 6.3.2. On the CRAY we used the message passing toolkit MPT 1.2.1.0 containing MPI and PVM message passing libraries.

Table II. Parallel computing systems.

Computer platform	CPU	Memory in MB	Network
PC-Cluster	12 AMD/Athlon, 600 MHz	12×512	Fast ethernet
CLiC	528 Intel P III, 800 MHz	528×512	Fast ethernet
CRAY-T3E	64 DEC Alpha 21164300 MHz	64×128	Gigaring

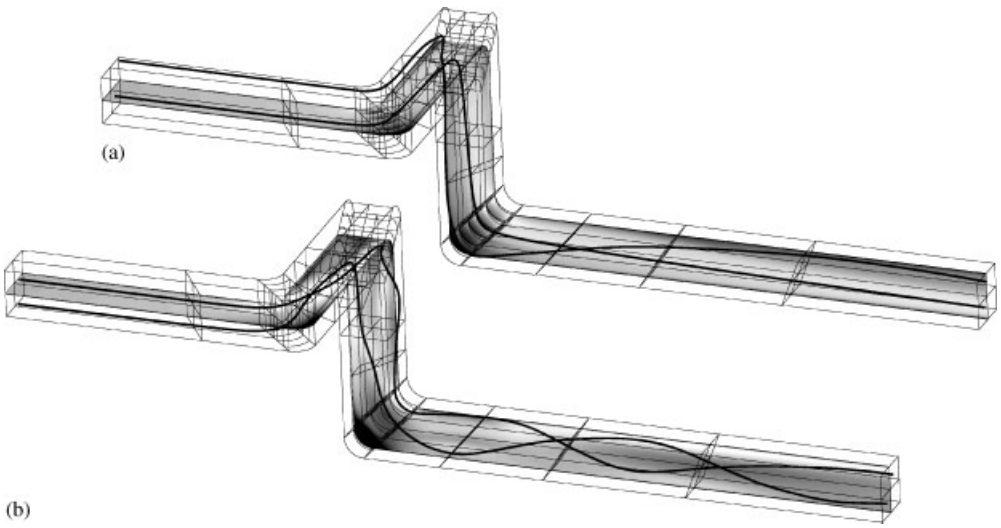

Figure 4. Flow through a bent duct: grid blocks, absolute velocity and two particle trajectories: (a) bent duct with blades; (b) bent duct without blades.

5.2. Description of the test cases

Two test cases are investigated. The first test case is a dilute gas–particle flow in a three times bent channel with square cross section of $0.2 \times 0.2\,\mathrm{m}^2$. In all three channel bends 4 corner vanes are installed, dividing the cross section of the bend in 5 separate corner sections (see Figure 4). The vanes are modelled as infinitely thin solid walls within the flow region (non-slip condition). The duct has been subdivided into 64 blocks, the number of finite volumes for the finest grid is $80 * 80 * 496 = 3\,174\,400$; because of the blades no more than three coarser grids can be used for the multigrid method. This means that the coarsest grid with only two cells between the blades has $10 * 10 * 62 = 6\,200$ finite control volumes. At the inlet a plug velocity profile with 10.0 m/s is given ($Re_\mathrm{F} = 156\,000$), at the outlet a zero gradient condition is implemented. The particle phase with particle diameters of $d_\mathrm{P} = 4, \ldots, 20\,\mu\mathrm{m}$ and a density of $\rho_\mathrm{P} = 2500\,\mathrm{kg/m}^3$ has initially a uniform concentration distribution over the inlet cross section of the duct. For each of the test case calculations 5000 particle trajectories have been calculated. Similar configurations are used for e.g. pneumatical conveying of granular material in channels and pipes.

The second test case differs from the first one by omitting the vanes in the channel bends. This leads to the development of a counter clockwise swirling fluid flow and due to the centrifugal forces acting on the particles to a strong separation of the particle phase from the fluid flow. Demixing of the particle phase starts immediately after the first bend and leads to

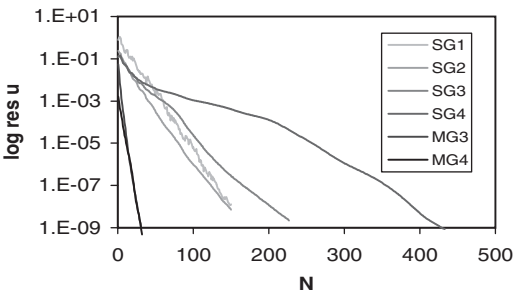

Figure 5. Convergence of the single- and multi-grid method on a sequence of refined grids.

the formation of a particle rope after the third channel bend. This test case has been introduced as an example of a strongly separated gas–particle flow in order to prove the suitability and performance of the developed load balancing algorithms for such kind of separated multiphase flows leading to poor parallel efficiency with the so far used SDD method.

5.3. Results

5.3.1. Flow calculation. The first test compares the convergence of two algorithms which use the multigrid (MG) method in different way. The first algorithm, denoted as single—grid method, is the common SIMPLE method where only the pressure calculation is accelerated with the MG method. Although this inner MG method leads to a much faster convergence of the iteration for the complete system the dependence of the number of iterations needed for a prescribed accuracy on the number of unknowns cannot be overcome in this way. The second algorithm, denoted as multigrid method, applies the MG-method to the complete system of equations and uses the SIMPLE-algorithm as smoothing method and coarse grid solver. The MG-acceleration for the pressure calculation is used additionally.

In the test the flow through the bent duct with blades is calculated on a sequence of refined grids. Figure 5 includes single-grid runs on four grids (SG1-SG4) with $10 * 10 * 62$ (coarsest grid) up to $80 * 80 * 496$ finite volumes and multigrid runs starting on the two finest grids (MG3, MG4). N denotes the number of SG iterations or MG cycles. The curves show that the number of iterations for the SG method increases while the number of MG cycles remains constant if the grid is refined. The total calculation times are decreased by the multigrid technique up to 1% of the original SIMPLE method.

By the way the calculations on the four grids pointed out that the solutions on the finest grid can be considered to be mesh-independent.

Figure 6 summarizes the parallel efficiency of the MG method on the CRAY-T3E and on the CLiC for a series of runs on an increasing number of processors. All calculations are performed for the bent duct with 64 blocks and 396 800 or 3 174 400 finite volumes on the finest grid. Generally the parallel efficiency is better for the calculations on the CRAY. This is due to the faster communication network. Table III shows calculation times T_cal and the times needed for data exchange T_e for the runs on the coarser grid. While with a growing number of nodes the ratio T_e/T_cal remains constant on the CRAY it grows from 0.42 to 0.8 on the CLiC. The run on the fine grid demonstrates that on the CLiC the parallel efficiency is fairly good as long as the number of finite volumes per node is not too small.

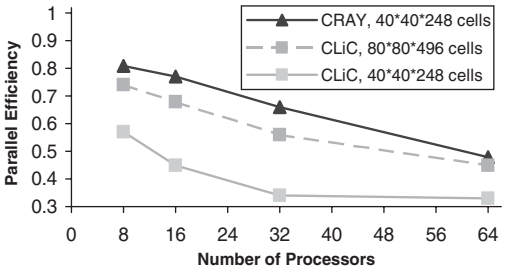

Figure 6. Parallel efficiency of the MG algorithm on CLiC and on a CRAY-T3E.

Table III. Calculation times and data exchange times for an increasing number of nodes.

Number of nodes	8	16	32	64
CRAY: T_cal	1661	863	499	349
T_e	396	199	103	80
T_e/T_cal	**0.24**	**0.23**	**0.21**	**0.23**
CLiC: T_cal	1022	669	449	249
T_e	434	400	330	200
T_e/T_cal	**0.42**	**0.60**	**0.73**	**0.80**

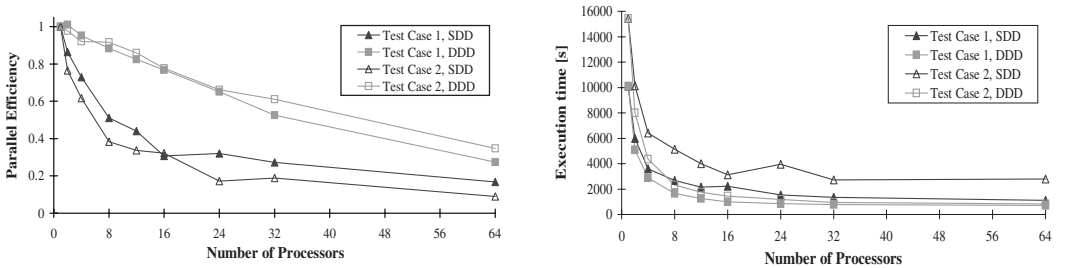

Figure 7. Parallel efficiency and total calculation times vs. number of processor nodes; comparison of parallelization methods for both test cases.

More details of the parallelization method and further results for fluid flow calculation can be found in Reference [2].

5.3.2. Particle calculation. For the test case calculations the total execution time, calculation time, communication time and I/O time have been measured for the execution of one iteration cycle of the Lagrangian solver. This means the calculation of 5000 particle trajectories. All calculations in this experiments have been carried out on the second finest grid level with $40 * 40 * 298 = 396.800$ CVs.

Figure 7 shows the parallel efficiency and total calculation times for calculations on both test cases with SDD and DDD methods vs. the number of processor nodes.

It can be seen from the figure that the DDD method has a clear advantage over the SDD method. The advantage is less remarkable for the first test case than for the second one. This is due to the fact, that the gas–particle flow in the first test case is quite homogeneous in respect to particle concentration distribution which leads to a more balanced work load distribution in the SDD method. So the possible gain in performance with the DDD method is not as large as for the second test case, where the gas–particle flow is strongly separated and we can observe particle roping and sliding of particles along the solid walls of the channel leading to a much higher amount of numerical work in certain regions of the flow. Consequently the SDD method shows a lower efficiency and the highest execution times for the second test case due to poor load balancing between the processors of the PM.

The efficiency of the work load balancing introduced into the Lagrangian approach by the DDD method is clearly reflected in the calculation and communication times measured for the processors of the PM. We consider two runs for test case 2 on eight processors (without a figure). In the SDD method two nodes use the largest amount of calculation time (about 90%) while the other nodes show up to 80% communication (waiting) time. For the DDD method the distribution of work load becomes very uniform with calculation times of about 90% on all physical processors. This balanced work load behavior remains unchanged also for larger numbers of processors and leads to the significantly increased parallel performance in comparison with the traditional SDD method.

6. NUMERICAL SIMULATION OF THE FLOW IN A FLOW SPLITTER

As an example we present the application of the method to a flow in the area of power engineering. The device of interest is a so-called bifurcator which is used in large coal-fired power plants. This bifurcator belongs to the very complex pipework between coal mills and burners that is necessary to ensure a preferably uniformly distributed supply of the burners with pulverized fuel from the coal mills even in the case when a single mill will turn out. Because of the complexity, the pipework consists among other things of a number of bends that cause the emergence of particle ropes mainly as a result of centrifugal forces.

These ropes would influence the distribution of the pulverized fuel to the burners in a negative manner. The result of such an unequal supply of the burners with coal is a lower efficiency and higher output of pollutants and must be avoided. That's why special attention is turned to the disintegration of the ropes. For these purposes the bifurcator contains a very complex fixture called a riffle box (see Figure 8(a)). In detail it consists of a system of 64 differently inclined channels and directly attached to these a system of vanes that lead the flux alternating to the both legs of the bifurcator. If a rope meets the riffle box, it is dispelled by the checkerboard like system of channels in several parts that are then distributed uniformly to the both legs because adjacent channels always lead to different legs.

The investigation of the real object is very difficult because experiments in a power plant at work are impossible and experiments for a model in original scale would be too expensive due to the large dimensions of the rope splitting device. Experiments are realized by A. Aroussi at the University of Nottingham on a one-third scale model. A part of the experiments has been performed with all internals (the complete riffle box). In this case the particle mass flow through the two outlets was measured. Velocity distributions were measured at three places in the pipes up and downstream the bifurcator without the internals.

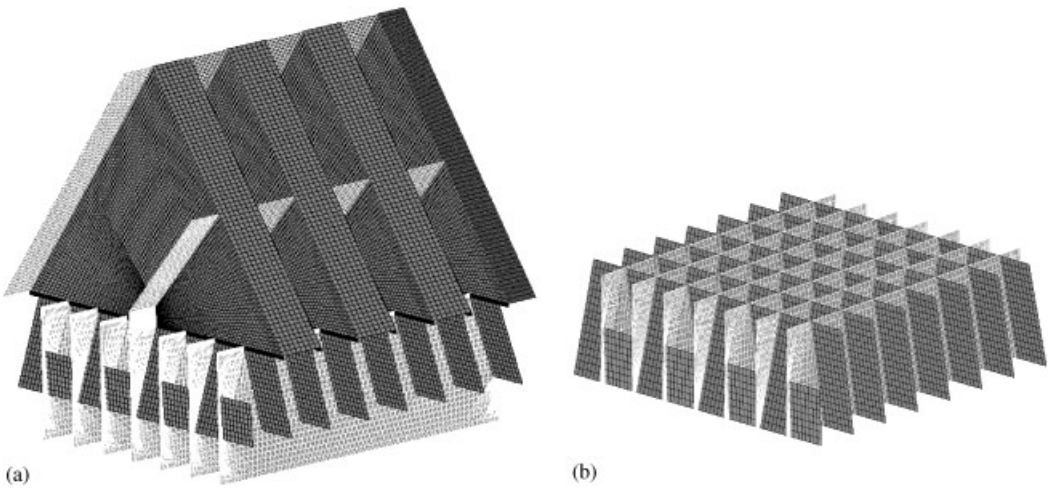

Figure 8. (a) Whole riffle box and (b) Riffle box, Lower part.

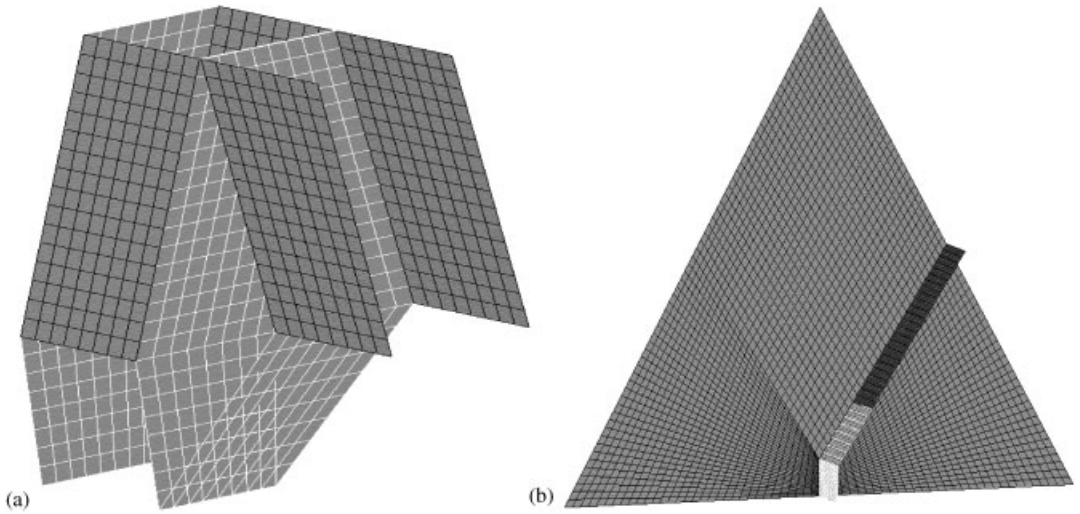

Figure 9. Riffle box: (a) Lower part, detail; (b) Upper part, detail.

The construction of a numerical grid is quite difficult because of the complex structure mainly of the riffle box. First of all the 64 channels inclined against each other (see detail of the grid in Figure 9(a)) are a serious problem especially for a structured grid that consists of hexahedrons (see detail of the grid in Figure 9(a)). So the grid in this region has to split alternating into different branches that pass either a left inclined or a right inclined channel. To realize this, a thin layer between the sets of left and right inclined channels is not belonging to the gridded area and form a thin wall of finite thickness. After the passing of the channels the grid unites again and must map to the guiding vanes that lead alternating

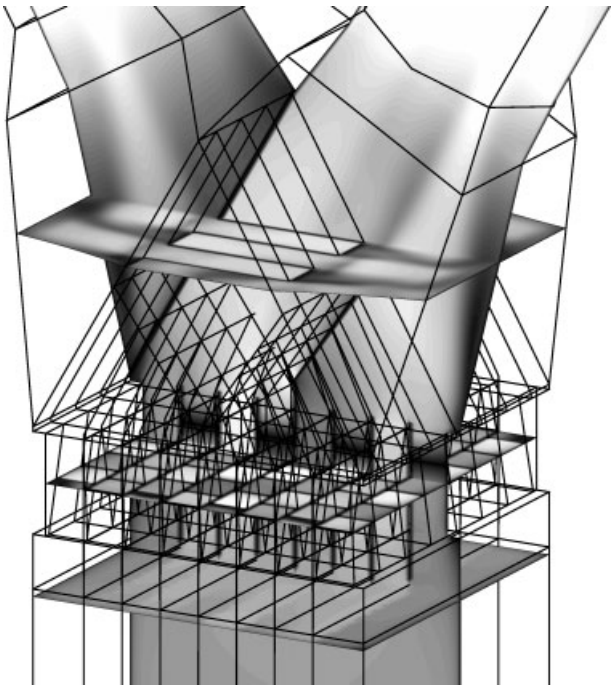

Figure 10. Shaded contours of the absolute velocity in the bifurcator, (black: 0 m/s; white: 36 m/s).

to the left and the right leg of the bifurcator and forms a triangular structure (see detail of the grid in Figure 9(b)), which is also difficult to grid with hexahedrons. In this case the triangle is divided into 3 quadrangles. The shape of these quadrangles is determined by the guiding vanes sitting on the triangles that lead either to the right as seen in Figure 9(b) or (alternating) to the left. The position of the guiding vanes marks the block boundaries in the interior of the triangle that cause another difficulty. Since the grid is structured, the subdivisions on the longer (outside of the triangle) edge are to find on the opposite edge. That means a considerable contraction of the grid within these blocks and simultaneously the so-called bad angles that are a general problem in the complex bifurcator geometry.

The result of the grid generation process are grids with about 2 millions of cells depending on the concrete case of application. The examples differ for instance from the form of the incoming flow region. So the inlet channel geometries had been varied for different real installation conditions of the bifurcator (straight or bent inlet; varying inlet channel length).

The computations were performed on a cluster of 12 PC with Athlon processors running under the operating System LINUX. The predictions start with an uniformly distributed inlet velocity of 30 m/s.

Results of the computations can be seen in Figures 10 and 11. Figure 10 gives an impression of the complex block structure especially in the region of the rope splitting riffle box. On the other hand there are slices through the geometry that show the contour of the field of the resulting absolute velocity, where the velocity increases from black to white (black: 0 m/s;

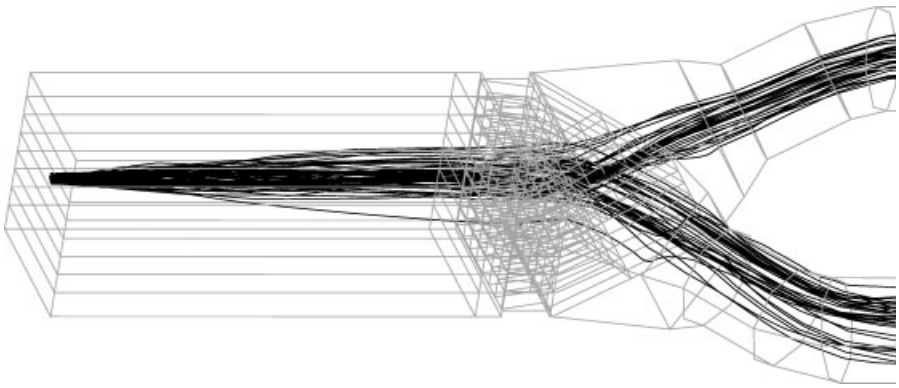

Figure 11. Particle trajectories in the flow splitter.

white: 36 m/s). Good to observe is the flow through the 64 inclined channels. The flow through the upper part of the riffle box can be seen in the highest of the three horizontal planes and shows the alternating distribution to the left and the right leg of the bifurcator. The illustration of the complex flow is completed by the slice in a plane perpendicular to the others.

In Figure 11 the operation of the riffle box is illustrated by drawing a number of particle trajectories. The particles were injected in the stream in a cross section that forms a distinct rope, which would leave the bifurcator completely through one leg in absence of the riffle box. But the rope is distributed relatively uniform to both legs due to two effects: First the rope is dispersed only under the influence of the turbulent motion in the flow and therefore enters more than one of the 64 channels. Secondly the rope is distributed by the riffle box to the both legs, because a particle is guided through the other leg, when it enters an adjacent channel even in the case when the rope is situated very close to one wall (or leg). This situation can be found e.g. when a bend is close to the inlet of the bifurcator. In addition to the trajectories in Figure 11 a number of statistical quantities of the particle phase was investigated. For example the distribution of the mean particle diameter (Figure 12) and the particle number density (Figure 13) was predicted by tracking $450 * 450 = 202500$ particles through the geometry. To obtain a non-uniform distribution of the basic flow and the particle phase in front of the riffle box, a 90° bend is added in the input region that causes an enrichment of particles on the left hand side of the cross-section. Especially the particles of higher diameters were carried to the left because of their higher mass and the resulting higher inertia (see Figure 12). The principle function of the bifurcator with the riffle box is good to observe. Although the particles were found mostly on the left of the cross-section in front of the bifurcator, the riffle box makes sure that the particle mass flow is subdivided nearly uniformly to both legs of the bifurcator. Also to observe is the function of the guiding vanes especially in the upper parts of the riffle box. The particles concentrate near these vanes due to inertial effects and form certain ropes (the white areas in Figure 13) that will be redistributed upstream over the whole cross-section by turbulence effects.

The principal function of the rope splitting devices was investigated in a number of experiments at the University of Nottingham. The bifurcator was operated completely without

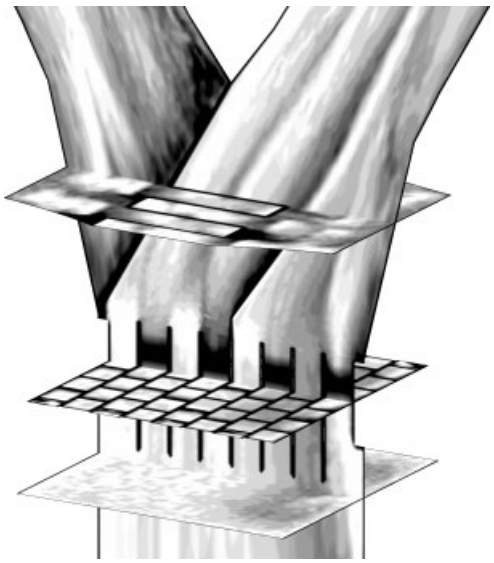

Figure 12. Distribution of mean particle diameter (black: 4 μm; white: 20 μm).

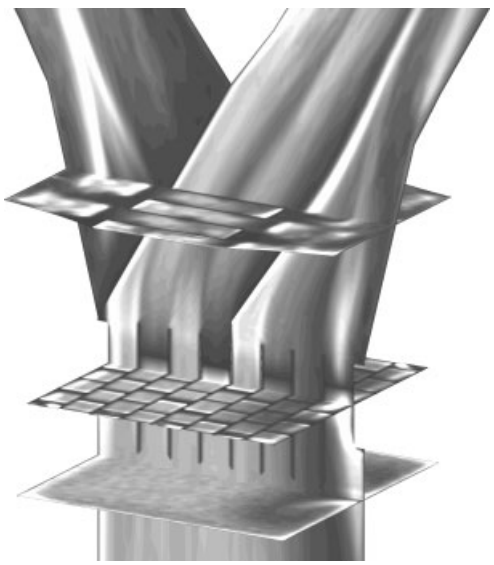

Figure 13. Distribution of mean particle number density (black: $0\,s^{-1}\,m^{-3}$; white: $10^{10}\,s^{-1}\,m^{-3}$).

internals, only with the lower part of the fittings and with the combination of the lower and upper part (the complete riffle box). The mass flow throughout the two outlets was recorded. A good function of the rope splitter was only to observe, when the complete internals were used. This result is not much surprising, because only the combination of lower and upper

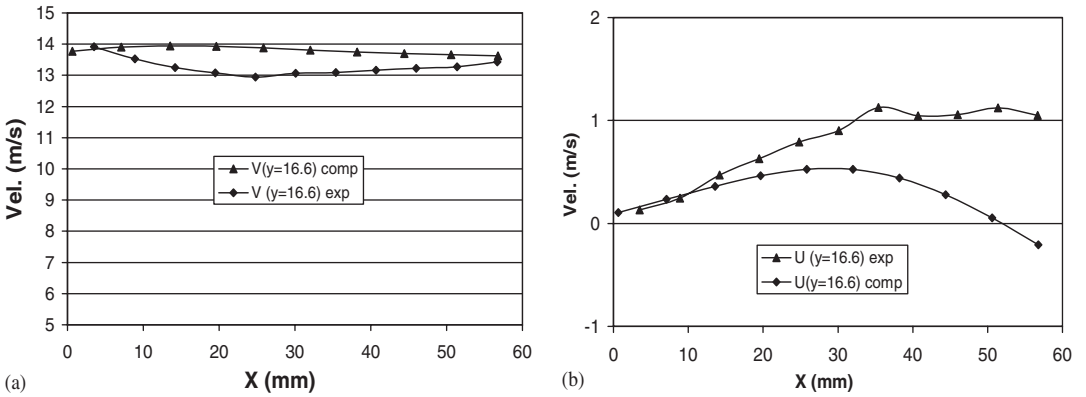

Figure 14. Comparison of velocity components in an outlet of the bifurcator; (a) longitudinal component; (b) transversal component.

part of the riffle box ensures that the mass flow through adjacent fields of the lower part is guided to different legs of the bifurcator and so ropes are splitted.

A major problem in such large 3d Lagrangian particle simulations is the statistical reliability of the obtained results. Even the high number of 202500 particles cannot ensure that every grid cell is crossed by at least one particle. The comparison with a calculation with 102400 particles, however, shows that the chosen number is completely sufficient to point out the behaviour of the particle phase. For integral quantities as the mass flow rate through the outlets still fewer particles give quite reliable results. For the two outlets of the bifurcator we obtained the particle mass flow ratios 48.14:51.86 with 22500 particles, 48.93:51.07 with 102400 particles and 48.74:51.26 with 202500 particles.

The results of the computations are compared with measurements of the flow in a bifurcator without internal fittings (riffle box) and with a long pipe inlet including three bends at the upstream side, because experimental data of the velocity distributions were available only for this case. For the purpose of verification of our computational results the existence of internals is not mandatory.

As an example for the comparison we present velocity distributions in one of the legs of the bifurcator. The measurements were done in a rectangular measuring area situated in the middle of the pipe (viewing window for the PIV measurements), where the lower left corner corresponds with the origin of the X-co-ordinate. The comparison is diagrammed in Figure 14(a) and 14(b). V is the velocity component in the main flow direction and U the appropriate rectangular one.

Generally it is to realize, that the computations characterize the principal behaviour of the flow. So a double helix of particle ropes arising from the bends is fully reproduced also in the numerical simulations whereas the comparison at a certain cross section sometimes shows differences concerning the precise position of the rope. This is due to some inevitable differences in the initial conditions between experiment and computation. (For example the uniform velocity and concentration distributions in the inlet cross section are hardly to obtain in the experiments done on the test rig.) Consequently some more or less different results are to observe between experiment and computation if the velocity distributions along a well-

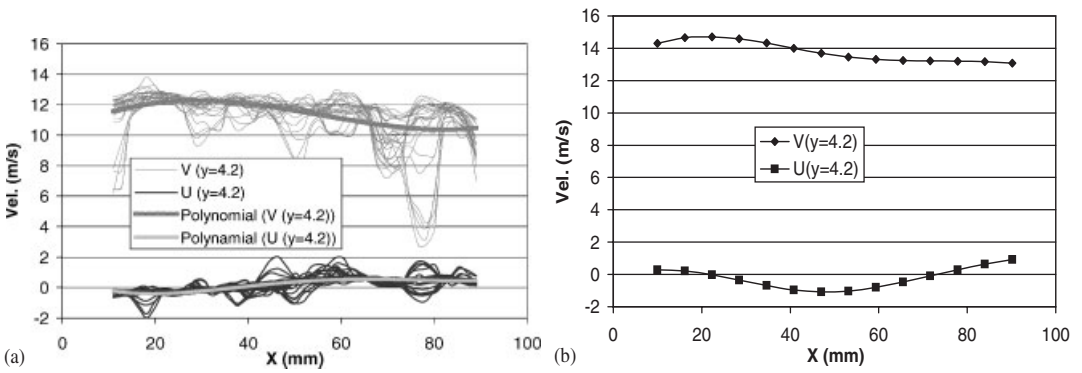

Figure 15. Comparison of (a) measured; and (b) computed velocity components in an outlet of the bifurcator.

defined cross section are compared. This is illustrated in Figure 15(a) and 15(b). The thin lines in 15(a) represent a certain number of measurements and show thereby the fluctuations in the flow. The thicker lines are trendlines and represent a compensation of these fluctuations over the time. Figure 15(a) is therefore also an example for the difficulties and reliability of the measurements in such complex fluid–particle flows. Nevertheless the distributions of longitudinal velocity components from experiments and from the numerical predictions were found to be in qualitatively good agreement. The differences between the absolute values may result from the above mentioned difficulties concerning the consistence of experiment and computation and from uncertain measurements.

7. CONCLUSIONS

The paper presents the parallelized code MISTRAL/PartFlow-3D for the simulation of two-phase flows on computers of MIMD type. The code includes a multigrid accelerated SIMPLE method for the fluid phase and two algorithms for the disperse phase, called static domain decomposition (SDD) and dynamic domain decomposition (DDD) method. Performance results are given for two typical test cases. The CFD code and the DDD method show a fairly good parallel efficiency on a PC cluster with FastEthernet network.

The suitability of the code for solving real problems from engineering is demonstrated with the application of the method to a fluid–particle flow in the field of power engineering. The code is applied to a quite complex technical component, the so-called bifurcator. The function of the riffle box within the bifurcator could be simulated. Solutions could be obtained for the basic gas flow and for the motion of the particle phase.

REFERENCES

1. Tysinger TL, Missaghi M. A combined shared-memory and distributed-memory model for computation of coupled Lagrangian dispersed phase and Eulerian gas phase combustion. *Proceedings of the International Conference on Recent Developments and Advances Using Parallel Computers*, '*Parallel CFD*'97'. Manchester, England, May 19–21, 1997.

2. Yonemura S, Tanaka T, Tsuji Y. Cluster formation in gas–solid flow predicted by the DSMC method. *Proceedings of the International Symposium on Gas–Solid Flows. ASME Fluids Engineering Conference*. Washington DC, USA (FED-1993; **166**:303–309).
3. Tsuji Y. Discrete element modelling of clusters in gas–solid flows. *Proceedings of the 2nd International Symposium on Numerical Methods for Multiphase Flows*, ASME Fluids Engineering Division Summer Meeting, San Diego, CA, USA, July 7–11 (FED-1996; **236**, **1**:3).
4. Byrde O, Sawley ML. Parallel computation and analysis of the flow in a static mixer. *Computers & Fluids* 1999; **28**:1–18.
5. Frank Th. Numerische Simulation der feststoffbeladenen Gasströmung im horizontalen Kanal unter Berücksichtigung von Wandrauhigkeiten. *PhD Thesis*, Techn. University Bergakademie Freiberg, Germany, 1992.
6. Sommerfeld M. Modellierung und numerische Berechnung von partikelbeladenen turbulenten Strömungen mit Hilfe des Euler/Lagrange–Verfahrens. *Berichte aus der Strömungstechnik*. Shaker Verlag, Aachen, Germany, 1996.
7. Fohanno S, Oesterle B. Analysis of the effect of collisions on the gravitational motion of large particles in a vertical duct. *International Journal of Multiphase Flows* 2000; **26**(2):267–292.
8. Sommerfeld M. Inter-particle collisions in turbulent flows: a stochastic Lagrangian model, *Conference on Turbulence and Shear Flow Phenomena*, *1st International Symposium*, Santa Barbara, CA, USA, September 1999.
9. Sommerfeld M. Modelling of particle–wall collisions in confined gas–particle flows. *International Journal of Multiphase Flows* 1999; **18**(6):905–926.
10. Frank Th, Wassen E, Yu Q. A 3-dimensional Lagrangian solver for disperse multiphase flows on arbitrary, geometrically complex flow domains using block–structured numerical grids. *7th International Symposium on Gas–Particle Flows*, ASME Fluids Engineering Division Summer Meeting, Vancouver, BC, Canada, June 22–26, *CD-ROM Proceedings*, FEDSM97-3590, 1997.
11. Frank Th, Schneider J, Yu Q, Wassen E. Experimental and numerical investigation of particle separation in a symmetrical double cyclone separator. *8th International Symposium on Gas–Particle Flows*, ASME Fluids Engineering Division Summer Meeting, San Francisco, CA, USA, July 18–22, *CD-ROM Proceedings*, Paper No. FEDSM99-7865, 1–10, 1999.
12. Frank Th. Application of Eulerian–Lagrangian prediction of gas–particle flows to cyclone separators. *Lecture Series Programme 1999–2000*, "*Theoretical and Experimental Modeling of Particulate Flow*", *VKI*, *Von Karman Institute for Fluid Dynamics*, Bruessels, Belgium, 03.–07, April 2000.
13. Crowe CT, Sommerfeld M, Tsuji Y. *Multiphase Flows with Droplets and Particles.* CRC Press: Boca Raton, 1998.
14. Bernert K, Frank Th, Schneider H, Pachler K. Multi-grid acceleration of a SIMPLE-based CFD-code and aspects of parallelization. *IEEE International Conference on Cluster Computing*—CLUSTER 2000, November 28.–December 2., Chemnitz, Germany, 2000.
15. Bernert K. τ-extrapolation—theoretical foundation, numerical experiment and application to Navier–Stokes equations. *SIAM Journal of Science and Computing* 1997; **18**(2).
16. Brandt A, Dinar N. Multigrid solutions to elliptic flow problems. In: *Numerical Methods for Partial Differential Equations*, Parter S.v. (ed). Academic Press: New York, London, Toronto, Sydney, San Francisco, 1979; 53–147.
17. Caretto LS, Gosman AD, Patankar SV, Spalding DB. Two calculation procedures for steady, three-dimensional flows with recirculation. *Proceedings of the Third International Conference on Numerical Methods in Fluid Dynamics*, Paris, 1972.
18. Ferzinger JH, Perić M. *Computational Methods for Fluid Dynamics.* Springer: Berlin, Heidelberg, 1996.
19. Frank Th, Wassen E. Parallel efficiency of PVM- and MPI-implementations of two algorithms for the Lagrangian prediction of disperse multiphase flows. *JSME Centennial Grand Congress* 1997, *ISAC*'97 *Conference on Advanced Computing on Multiphase Flow*, Tokyo, Japan, July 18–19, 1997.
20. Frank Th, Wassen E, Yu Q. Lagrangian prediction of disperse gas–particle flow in cyclon separators. *ICMF '98–3rd International Conference on Multiphase Flow 1998*, Lyon, France, June 8.–12., 1998. *CD-ROM Proceedings*, Paper No. 217, 1–8.
21. Frank Th, Bernert K, Schneider J. Numerische Untersuchungen der Gas–Partikel–Strömung in symmetrischen Doppelzyklon–Abscheidern, *VDI Berichte*, 1999; (1511).
22. Hackbusch W. *Multigrid Methods and Applications*. Springer-Verlag: Berlin, 1985.
23. Rhie CM, Chow WL. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA J.* 1983; **21**:1525–1532.
24. Stone HL. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM Journal of Numerical Analysis* 1968; **5**:530–558.
25. Stüben K, Trottenberg U. Multigrid methods: fundamental algorithms, model problem analysis and applications, *Lecture Notes in Mathematics*, vol. 960. Springer-Verlag: Berlin, 1982.

