Parallel Implementation of the Fluid Particle Model for Simulating Complex Fluids in the Mesoscale

Krzysztof Boryczko, Witold Dzwinel
AGH Institute of Computer Science, al. Mickiewicza 30, 30-059, Kraków, Poland

David A.Yuen[1]
Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, Minnesota 55415-1227, USA

Abstract

Dissipative particle dynamics (DPD) and its generalization – the fluid particle model (FPM) - represent the "fluid particle" approach for simulating fluid-like behavior in the mesoscale. Unlike particles from molecular dynamics (MD) method, the "fluid particle" can be viewed as a "droplet" consisting of liquid molecules. In FPM, "fluid particles" interact by both central and non-central, short-range forces with conservative, dissipative and Brownian character. In comparison to MD, FPM method in 3-D requires two to three times more memory load and three times more communication overhead. Computational load per step per particle is comparable to MD due to the shorter interaction range allowed between "fluid particles" than between MD atoms. The classical linked-cells technique and decomposing the computational box into strips allow for rapid modifications of the code and for implementing non-cubic computational boxes. We show that the efficiency of the FPM code depends strongly on the number of particles simulated, geometry of the box, and the computer architecture. We give a few examples from long FPM simulations involving up to 8 million fluid particles and 32 processors. Results from FPM simulations in 3-D of the phase separation in binary fluid and dispersion of colloidal slab are presented. Scaling law for symmetric quench in phase separation has been properly reconstructed. We show also that the microstructure of dispersed fluid depends strongly on the contrast between kinematic viscosities of this fluid phase and the bulk phase. This FPM code can be applied for simulating mesoscopic flow dynamics in capillary pipes or critical flow phenomena in narrow blood vessels.

Keywords: fluid particles, parallel algorithm, checker-board periodic boundary conditions, phase separation, dispersion, blood flow simulation

Submitted to: *Concurrency: Practice and Experience*, November 2001

[1] *corresponding Author, Minnesota Supercomputing Institute, University of Minnesota, 1200 Washington Av. South, Minnesota, 55415-1227, USA e-mail: davey@krissy.msi.umn.edu* ***fax: 612 624 8861***

1 Introduction

Dynamical processes occurring in the mesoscopic fluids involve many disparate spatio-temporal scales, from microscopic interactions of discrete particles, thermal fluctuations, multiphase mixing and large-scale disturbances. The implementations of these well-known computational techniques such as:

1. molecular dynamics (MD), used in in large-scale simulations [1-7],
2. finite element methods (FEM), employed in direct numerical simulations (DNS) [8-10],

are still too demanding and, in many cases, they are not adequate for resolving fine enough features at the mesoscale.

In the last decade we have witnessed the rapid growth in new approaches for modelling multi-scale phenomena. They are the grid techniques (Lattice-Boltzmann Gas- LBG, cellular automata [11,12] and the meshless fluid particle methods (DPD-dissipative particle dynamics [13] SPH- smoothed particle hydrodynamics, [14,15], direct simulation Monte-Carlo [16]). The fluid particle can be viewed physically as a "droplet" consisting of liquid molecules with an internal structure and with some internal degrees of freedom.

The fluid particle methods have at least four important advantages over the grid techniques.

1. The dynamics of fluid particles develop over continuum space in real time, thus allowing for realistic visualization and physical understanding.
2. Within the context of cross-scaling systems they are homogeneous with both microscopic molecular dynamics and macroscopic smooth particle hydrodynamics techniques [17]. The **transition** from discrete to continuum model is not necessary.
3. The methods employing fluid particles are also homogeneous within the context of solid-liquid simulations for which both solid and liquid are represented by particles [18,19].
4. They are also homogeneous from implementation point of view. Well-known sequential and parallel algorithms from MD simulations can be employed directly.

In [19-23] we demonstrate that the dissipative particle dynamics (DPD) [13] - the method employing fluid particles - fits very well for simulating multi-resolution structures of complex fluids. Typical examples of complex fluids with large molecular structure include miscellar solutions, microemulsions and colloidal suspensions such as blood, ink, milk, fog, paints, magma melts with long silicate chains, and waste products [24].

For complex fluids the gap in the spatio-temporal scales between the smallest microstructures and the largest structures is much smaller than for simple fluids. In [19,22,23] we show that by using moderate number of particles we can simulate in two dimensions multi-resolution structures ranging from micelles, micellar arrays, colloidal agglomerates and large-scale instabilities induced by the global flow. For

realistic modeling of multi-resolution structures in three dimensions, the large-scale simulations involving hundreds of processors working in parallel are necessary for covering the same spatio-temporal scales, as those employed in current high-resolution 2-D simulations.

Fluid particle model (FPM) [17] is a generalization of dissipative particle dynamics method. The FPM method is computationally more intensive than DPD and MD for a constant interaction range. However, FPM has an advantage over DPD for larger scales, in which the fluid particles are adequately large and can interact only with their closest neighbors. In such a case DPD is less efficient because many more particles than for FPM should be involved for creating a drag between the DPD particles. Therefore, we can expect that FPM will enable the eventual simulation of complex fluids in 3-D, with reasonable resources of parallel systems, which are operating in the multi-job mode.

First, we will describe the FPM model. Then we present a parallel algorithm for FPM and discuss its efficiency. We then define the boundary conditions of the code. Next, the parallel clustering procedure for detection of multi-resolution structures is presented. We then show the results of tests from the FPM simulations of phase separation in binary mixture and dispersion of colloidal slab in an elongated computational box. Finally, we summarize our findings and discuss the prospects of employing the FPM parallel algorithm in realistic large-scale simulations, such as flow in constricted blood vessels [25].

2. Fluid particle model

The equations of fluid dynamics describe the motion of mass in both time and space. Particle methods are based on the idea of simulating a fluid flow as a flow of particles, which interact by short-range forces. The total mass is subdivided into a finite number of small mass packets, which are called particles. Their structure is described by particles mass distribution $\phi_i>0$. The particles can move independently of each other. The total mass density is given by:

$$r(x,t)=\sum_{i=1}^{N} f_i\left(x-\mathbf{r}(t)\right) \tag{1}$$

For point-like particles, ϕ_i is the Dirac delta function. The temporal evolution of the particle system is then described by the equations of motion for the particle position $\mathbf{r}(t)$. Fluid particle model (FPM) described here, differs from dissipative particle dynamics (DPD) by Hoogerbrugge and Koelman [13,17,26]. The fluid particles can rotate in space and should be understood as comparatively big mass packets, though they are still particles in the sense of statistical mechanics. The forces of interaction between the particles are pair forces of a finite range, unlike in smoothed particle dynamics (SPH) [14,15], and a broad class of particle methods [27] where they are derived in a canonical manner from the force laws of continuum

mechanics and are directly based on a regularized stress tensor. SPH cannot be applied to the study of mesoscopic system in the Brownian realm, because the implementation of Lagrangian fluctuating hydrodynamics with SPH is not a trivial problem.

The fluid particles [17] are represented by their centers of mass, which posses several attributes, as mass m_i, position $\mathbf{r}_i$, translational and angular velocities and type. The "droplets" interact with each other by forces dependent on the type of particles. This type of interaction is a sum of conservative force $\mathbf{F}^C$, two dissipative components $\mathbf{F}^T$ and $\mathbf{F}^R$ and the Brownian force $\tilde{\mathbf{F}}$, that is:

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^{C} + \mathbf{F}_{ij}^{T} + \mathbf{F}_{ij}^{R} + \tilde{\mathbf{F}}_{ij} \tag{2}$$

The $\mathbf{F}_{ij}$ force components are defined by:

$$\mathbf{F}_{ij}^{C} = -V'(r_{ij}) \cdot \mathbf{e}_{ij} \tag{3}$$

$$\mathbf{F}_{ij}^{T} = -\gamma \cdot m\mathbf{T}_{ij} \bullet \mathbf{v}_{ij} \tag{4}$$

$$\mathbf{F}_{ij}^{R} = -\gamma \cdot m\mathbf{T}_{ij} \bullet \left(\frac{1}{2}\mathbf{r}_{ij} \times (v_i + v_j) \right) \tag{5}$$

$$\tilde{\mathbf{F}}_{ij}\,dt = s\left(\tilde{A}(r_{ij})\,d\overline{\mathbf{W}}_{ij}^{S} + \tilde{B}(r_{ij})\frac{1}{D}tr[d\mathbf{W}]\mathbf{1} + \tilde{C}(r_{ij})\,d\overline{\mathbf{W}}_{ij}^{A} \right) \bullet \mathbf{e}_{ij} \tag{6}$$

where:

r_{ij} – is a distance between particles i and j, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ is a vector pointing from particle i to particle j and $\mathbf{e}_{ij}=\mathbf{r}_{ij}/r_{ij}$, D – is the model dimension, dt – is the timestep, γ - scaling factor for dissipation forces, w - angular velocity, $d\mathbf{W}^S$, $d\mathbf{W}^A$, $tr[d\mathbf{W}]\mathbf{1}$ - are respectively the symmetric, antisymmeric and trace diagonal random matrices of independent Wiener increments and A(r), B(r), C(r), $\tilde{A}(r), \tilde{B}(r), \tilde{C}(r)$, V'(r) – functions dependent on the separation distance $r=r_{ij}$. $\mathbf{T}_{ij}$ – is dimensionless matrix given by:

$$\mathbf{T}_{ij} = A(r_{ij})\mathbf{1} + B(r_{ij})\mathbf{e}_{ij}\mathbf{e}_{ij} \tag{7}$$

$\mathbf{1}$ – is the unit matrix.

As shown in Español [17], the single component FPM system yields the Gibbs distribution as the steady-state solution to the Fokker-Planck equation under the condition of detailed balance, i.e.,

$$s^2 = 2k_B T\gamma \cdot m \tag{8}$$

where: T – is the temperature of particle system, k_B – the Boltzmann constant.

As a consequence, it obeys the fluctuation dissipation theorem, which defines the relationship between the normalized weight functions, which are chosen such that:

$$A(r) = \frac{1}{2}\left[\tilde{A}^2(r) + \tilde{C}^2(r)\right], \quad B(r) = \frac{1}{D}\left[\tilde{B}^2(r) - \tilde{A}^2(r)\right] + \frac{1}{2}\left[\tilde{A}^2(r) - \tilde{C}^2(r)\right], \tag{9}$$

For the dissipative particle dynamics (DPD) method A(r)=0, consequently $\tilde{A}(r),\tilde{B}(r),\tilde{C}(r)=0$ and V'(r)∝B(r), which means that all the DPD forces are central.

The non-central force in FPM, which is proportional to the difference between particle velocities, introduces an additional drag lacking in the DPD model. The non-central force results also in additional rotational friction given by Eq.(5).

The temporal evolution of the particle ensemble obeys the Newtonian laws of motion:

$$\dot{\mathbf{v}}_i = \frac{1}{m_i}\sum_{j;r_{ij}<r_{cut}} \mathbf{F}_{ij}\left(\mathbf{r}_i, n_i, v_i\right) \qquad \dot{\mathbf{r}}_i = \mathbf{v}_i \tag{10}$$

$$\dot{v}_i = \frac{1}{I_i}\sum_{j;r_{ij}<r_{cut}} \mathbf{N}_{ij}\left(\mathbf{r}_i, n_i, v_i\right) \qquad \mathbf{N}_{ij} = -\frac{1}{2}\mathbf{r}_{ij}\times\mathbf{F}_{ij} \tag{11}$$

The FPM method can predict the transport properties of the fluid, thus allowing one to adjust the model parameters by using the equations of continuum limit for the partial pressure *P* [13]:

$$P = \frac{n\cdot\int d\mathbf{r} r V'(r)}{2\cdot D} \tag{12}$$

and formulas in kinetic theory [17] for respectively bulk viscosity ν_b, shear viscosity ν_S and rotational viscosity ν_R:

$$n_b = g\cdot n\left[\frac{A_2}{2D}+\frac{D+2}{2D}B_2\right]+c^2\frac{1}{g\cdot Dn\left(A_0+B_0\right)} \tag{13}$$

$$n_S = \frac{1}{2}g\cdot n\left[\frac{A_2}{2}+B_2\right]+c^2\frac{1}{2gn\left(A_0+B_0\right)} \tag{14}$$

$$n_R = \frac{gnA_2}{2} \tag{15}$$

where:

$$c^2 = \frac{k_B T}{m} \tag{16}$$

and

$$A_0 = \int d\mathbf{r}A(r), \quad B_0 = \frac{1}{D}\int d\mathbf{r}B(r), \quad A_2 = \frac{1}{D}\int d\mathbf{r}r^2 A(r), \quad B_2 = \frac{1}{D(D+2)}\int d\mathbf{r}r^2 B(r) \tag{17}$$

The kinetic theory for FPM has been developed for deriving transport coefficients by assuming that conservative forces are absent. For non-zero pressures, the transport coefficients computed from Eqs.[13-

15] can be used as the first approximation in an iterative procedure, which matches the coefficients of the FPM forces.

Unlike in SPH - the angular momentum is conserved exactly in FPM. The model can be physically interpreted as a Lagrangian discretization of the non-linearly fluctuating hydrodynamic equations.

3. Numerical realization

3.1 Decomposition of computational box

We consider here an isothermal two-dimensional system, which consists of M particles. The particle system is simulated within a rectangular box. The particles of uniform or various types can be distributed randomly in the box, i.e., this multi-component system can be perfectly mixed initially, or separated by a sharp interface (stratified, circle, rectangular, random shape). The particles defined by mass m_i, position $\mathbf{r}_i$, velocity $\mathbf{v}_i$ and angular velocity w_i interact with each other via a two-body, short ranged forces given by Eqs.(3-6).

We assume that the weight functions (Eqs.(6-7)) satisfy the conditions imposed. Due to the choice allowed by the model in selecting the weight functions, we assume that:

$$\tilde{A}(r)=0,\quad A(r)=B(r)=\frac{15}{2p\cdot r_{cut}^3 n}\left(1-\frac{r}{r_{cut}}\right)^2,\ V'(r)=-\Pi\cdot\frac{3}{p\cdot r_{cut}^3 n}\left(1-\frac{r}{r_{cut}}\right) \tag{18}$$

where, r_{cut} – is a cut-off radius, which defines the range of FPM particle interactions. For $r_{ij}>r_{cut}$, $\mathbf{F}_{ij}=0$. The first assumption is recommended in [17]. We postulate the rest of weight functions the same as in DPD [13,26]. Due to additional drag between particles caused by the non-central interactions, we can reduce the computational load by assuming that the interaction range is shorter than for DPD fluid.

As shown in Fig.1, the box is divided onto cubic cells of the edge size $l_C \sim r_{cut}$. For multi-component fluid with different interaction ranges we assume that $l_C \sim \max_k(r_{cut,k})$, where k means the kind of interaction. The forces are computed by using O(M) order *link-list* scheme [28]. The force on a given particle includes contribution from all the particles that are closer than r_{cut} and which are located within the cell containing the given particle or within the adjacent cell (see Fig.1).

Parallel computing requires decomposing the computation into subtasks and mapping them onto multiple processors. The total volume of the box is divided into P overlapping subsystems of equal volume, and each subsystem is assigned to a single processor in a P processors array. By using SPMD paradigm (*single program multiple data*), commonly used for MD code parallelization, each processor

follows an identical predetermined sequence to calculate the forces on the particles within assigned domain.

Among many parallel implementations of molecular dynamics code [1,3,5,6,29] two approaches for particles redistribution between processors are employed.

1. The box is sliced along one coordinate and divided up onto identical sub-boxes (see Fig.1).
2. The system is partitioned into a mesh of sub-boxes in x, y and z directions.

The particle positions and velocities from cells, which are situated on the boundaries between processor domains, are copied to the neighboring processor (see Fig.1). Thus the number of particles located in the boundary cells defines the communication overhead. Let us assume that:

1. The system is confined in a box elongated in z direction and the *x*,*y* cross-section of the box is a square of unit area.
2. The number of processors P, the system size and the length of the box L_z>>1 are constant.
3. The box is partitioned along *z*-axis onto processor domains.

For this case, the communication overhead t_{strips}, which is proportional to the area of the interface between processor domains, is constant and equal to 1. Let us assume that the box is partitioned additionally into n^2 -mesh of identical sub-boxes on *x*,*y* plane. The communication cost t_{box} for *n*>1 is proportional to the area of walls (only half of them) of a single sub-box and is equal to $2L_z/(P/n^2)\times 1/n + 1/n^2$. For sufficiently long boxes with L>>1 the ratio of two overheads $j = t_{box}/t_{strips} = [2L_z/(P/n^2)]\times 1/n + 1/n^2$ is greater than 1. This means that the communication overhead is lower - and consequently calculation communication ratio higher - for the first method consisting in slicing the box onto strips along z coordinate. The value of j is less than one for more regular computational boxes, e.g., a cubic computational box for which $P \sim n^3$ and L_z=1. In this case the second partition method dividing the box onto cubic sub-domains is better.

Slicing the box along the *z*-axis considerably simplifies the routing of messages and enables sending them in unblocking way. Each processor sends the message only in one direction to its closest neighbor. The load balancing is easier and consists in shifting the boundaries of processor domains along one direction, while for the second method the load balancing schemes are very complex requiring irregular mesh. In Fig.2 we present the sequence of computation and communication procedures invoked in parallel implementation of FPM code.

Many parallel implementations of molecular dynamics codes employ neighbor tables for each particle for speeding-up the evaluation of forces. This increases considerably the memory requirements, communication overheads and makes the code more complex. Fluid particle model (FPM) has two-four times greater memory requirements than codes for molecular dynamics. Besides the positions and forces in highly optimized parallel codes for large-scale MD [3] (minimum 6 arrays), additional arrays must be allocated such as: the angular and translational velocities, torques and replicated arrays for velocities

needed for integrating Newtonian equations of motion (see Eqs.(21-24)), that is, minimum 24 arrays. Moreover, the random number generator is invoked 4 times for computation of Brownian forces for each pair of interacting particles.

Therefore, the speed-up expected from application of neighbor tables can be compromised due to the effect of frequent cache misses resulting from its overload. The particles from boundary cells "cached" on the neighboring processors and those migrating from one processor to another must be updated every timestep (see Fig.2). Unlike in MD, the FPM forces (see Eqs.3-6) depend not only on the particle positions but also on translational and angular velocities. Moreover, besides reaction forces, the reaction torques must be updated. Thus, the communication overhead is almost three times greater for FPM than for MD. Because FPM fluid particle interacts only with their closest neighbors, the number of interactions per particles is smaller by factor of 4.5 than for a standard MD code. However, the number of arithmetic operations involved for evaluation of FPM interactions is greater, at least by the same factor, than for calculating the Lennard-Jones forces in MD code. Thus, we may expect that computational load per particle should be similar for these two cases. Summarizing, the high memory load in FPM will result in:

1. greater communication overhead,
2. more frequent occurrence of cache misses,

than in standard implementations of MD method in multiprocessor environment (e.g., in [3,5]).

3.2 Temporal evolution of fluid particles

Integration of the Newtonian equations of motion in the fluid particle model is more complex than in MD and DPD codes. From Eqs.(3-6,10-11) we note that the forces and torques depend not only on particle positions (as in MD) and translational velocity (as in DPD case) but also on angular velocities. Moreover, due to the random Brownian force, the equation of motion are stochastic differential equations (SDE). Numerical integration of SDE by using classical Verlet scheme [30] generates large numerical errors [31] and artifacts, e.g., resulting in unacceptable temperature drift with simulation time. Therefore,very small timesteps should be used to obtain a reasonable approximation to the thermodynamical quantities. On the other hand, predictor-corrector numerical schemes are both very time and memory consuming, which for high memory load for FPM will result in additional overheads. Therefore, we have decided to employ extrapolation schemes, which we used successfully in our 2-D MD-DPD and MD-FPM codes [19]. The schemes are as follows:

$$\mathbf{v}_i^{n+\frac{1}{2}} = \mathbf{v}_i^{n-\frac{1}{2}} + \frac{\Delta t}{m_i} \sum_{j;r_{ij}<r_{cut}} \mathbf{F}_{ij}\left(\mathbf{r}_i^n, \tilde{\mathbf{v}}_i^n, \tilde{\nu}_i^n\right) \tag{19}$$

$$\nu_i^{n+1/2} = 2\nu_i^{n-1/2} - \nu_i^{n-2/3} + \frac{\Delta t}{I_i}\left(\mathbf{N}_i^n - \mathbf{N}_i^{n-1}\right) \tag{20}$$

$$\mathbf{N}_i^n = \sum_{j;r_{ij},r_{cut}} \mathbf{N}_{ij}\left(\mathbf{r}_i^n, \tilde{\mathbf{v}}_i^n, \tilde{\nu}_i^n\right) \tag{21}$$

$$\tilde{\mathbf{v}}_i^{n+1} = \frac{1}{2}\left(3\mathbf{v}_i^{n+\frac{1}{2}} - \mathbf{v}_i^{n-\frac{1}{2}}\right) \quad \tilde{\nu}_i^{n+1} = \frac{1}{2}\left(3w_i^{n+\frac{1}{2}} - \nu_i^{n-\frac{1}{2}}\right) \tag{22}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \mathbf{v}_i^{n+\frac{1}{2}} \cdot \Delta t \tag{23}$$

To assure the numerical stability of the particle system, Eq.(11) representing the conservation of angular momentum is integrated by using the scheme of order $o(\Delta t^4)$ (see Eq.(20)). The coordinates of vectors $\tilde{\mathbf{v}}, \tilde{?}$ in Eqs.(19,21) are extrapolated by using Adams-Bashfoth $o(\Delta t^2)$ scheme (see Eqs(22)).

The size of the timestep Δt should be estimated from the characteristic time scales for both rotational and translational motion. The mean collision time τ_{col} defines the time scale for the translational motion, which is given by:

$$t_{col} = \frac{l}{\langle v_{rel} \rangle} \tag{24}$$

where $\langle v_{rel} \rangle$ is a relative velocity, λ- is the characteristic length scale, which is equal to the average distance between particles.

Both the quality and numerical stability of the model can be estimated from the temporal behavior of the thermodynamic temperature Tth and dimensionless pressure $\delta = k_B \cdot T/(P/n)$. As shown in Fig.3, the temperature Tth of the system, computed as the average kinetic energy of the FPM particle systems, fluctuates no more than 1.5% percent. Its average differs from the temperature T assumed (computed from detailed balance Eq.(8)) on about 0.1%. For comparison, at the similar simulation conditions (but in 2-D) and the same timestep, the equilibrium temperature Tth for DPD simulation of phase separation obtained in [31] is roughly twice its input value. The temperature drift (upward or downward, depending on the hardware and compiler used) caused by the round-off error, which is apparent for large number of timesteps, we have greatly reduced by using 64 bit compiler. The value of δ, which represents the reciprocal of partial pressure *P*th of FPM fluid computed from the viral theorem [30], can also be approximated accurately by the Eq.(12) (see Fig.3).

3.3 Boundary conditions

Periodic boundary conditions (PBC) simulate the system of unlimited number of interacting particles by limited number of interacting lattices where each of them stands for a particle and its replicas. When the distance between a particle and its nearest image is too short, long wavelength phenomena are cut and their energy is passed to the shorter waves, which go through the box generating numerical artifacts. Moreover, the commonly used computational box shape, such as rectangular prism, makes the system highly anisotropic. In [32,33] the minimum image convention is presented for non-cubic boxes such as truncated octahedron, rhombic dedocahedron and hexagonal prism. In spite of the more symmetric geometry and savings in CPU time due to increase of the nearest image distance, the non-cubic boxes are still not popular in particle simulations. There are at least two basic problems with non-cubic boxes for simulating large particles ensembles.

1. Non-cubic boxes involve non-cubic cells in the linked-cells algorithm. This makes the code very clumsy (especially in 3-D) due to greater number of walls, edges and vortices in non-cubic cells than for cubic ones, thus involving complicated nearest image convention schemes [33].
2. Domain decomposition is difficult for non-cubic boxes.

In [34], a method for uniformization of the periodic box shape for small particle system was presented. As shown in Figs.4a,b, the periodic box can be divided onto two, black and white, rectangles of the same size. Unlike for the periodic square, the box replicas are shifted creating checker board picture (see Fig.4). For properly selected box sizes Lx, Ly and Lz one can reproduce different shapes. For example, the periodic hexagon can be simulated assuming that $Lx/Ly=1/\sqrt{3}$ (see Fig.4a) while the box with Lx=1, Ly=1 and Lz=2 (see Fig.4b) corresponds to periodic rhombic dodecahedron [34].

In Fig.5 we compare the largest circles inscribed in a hexagonal and square boxes of the same area, which diameter represent the distance between a particle and its nearest image. Diameter of the circle inscribed in hexagon is about 7% greater than in the square. In three dimensions, the sphere inscribed in a rhombic dodecahedron is 15% larger than the largest sphere inscribed in a cubic box of the same volume. For keeping the same distance between the particles and their nearest images, one can employ periodic rhombic dodecahedron with particle ensemble 40% smaller than those for the cubic box.

The possibility of application of linked-lists method with cubic cells for non-cubic periodic boxes is the great advantage of using the checker-board PBC. Below we present the translation scheme for renumbering the cell coordinates: Nx, Ny and Nz, from the border of the computational box by replicating periodic rombic dodecahedron.

```
ix = INT ( (float(Nx)/N) +1)    -  1
iy = INT ( (float(Ny)/N) +1)    -  1
iz = INT ( (float(Nz)/N) +1)    +  INT ( float (Nz)/N) – INT (float(Nz)/(2*N)) – 2
iz = iz * [(abs(ix) + abs (iy) +abs(iz) – 1 ) mod 2]

Nx = Nx – N*ix
Ny = Ny – N*iy
Nz = Nz – N*iz
```

The parallel code for the checker board periodic boundary conditions is relatively easy to implement by assuming that the box is decomposed by segmenting it along x or y coordinate. For boxes elongated in *z* direction, such the decomposition will increase communication time due to thin layers of domains and larger interface area between neighboring processor domains. Slicing the box along *z*-axis (see Fig4b) may generate even more serious problems with communication. The processors will communicate not only with their neighboring processors, as it is for periodic rectangle shown in Fig.1, but also with the distant processors. In this situation, communication time may depend strongly on the architecture and memory access time of the parallel system.

For simulating the flow in an elongated and periodic capillary, we have employed the hexagonal prism PBC shown in Fig.4a. The checker board PBC are realized only on *x,y* plane. This preserves more circular shape of the capillary section than for a periodic rectangular prism and allows us to employ the same strategy of domain decomposition as shown in Fig.1. We simulate the box with circular section in *x,y* plane with reflecting or dissipative boundaries in *x* and *y* directions by filling white space in Fig.5 with heavy or motionless particles.

3.4 Clustering procedure

The patterns created in macrosopic flows, for which a homogeneous physical process dominates in multiple spatio-temporal scales, have typically self-similar fractal structures. In [19-23] we show that the strong heterogeneities of the flow in the mesoscale co-produce complex multiresolution patterns [24]. The creation of micelles, colloidal arrays, colloidal agglomerates and large-scale instabilities in fluid are the consequence of the competition between two coupled non-linear processes: global motion of particle ensembles and local interactions between particles. These multiscale structures are complex due to the inflexibility of the description level with varying scale of observation. The detection of particle clusters for controlling their temporal behavior represents a very important aspect in visualizing and extracting the complex patterns.

We have solved the problem for detecting clusters by using efficient O(M) algorithm inscribed in parallel structure of FPM code. The algorithm is based on the mutual nearest neighborhood (MNN) concept. The algorithm is outlined as follows:

1. Find the list $\boldsymbol{L}_i$ of K nearest neighbors j of each particle i in $\mathbf{R}_{clust}$ radius and sort out the list in ascending order according to the distance between i and j particles. Thus $\boldsymbol{L}_i(k)=j$ and k is the position of the particle j in the list. This procedure can be performed in parallel along with computation of forces in FPM code. To reduce the communication overhead, we use the parallel clustering algorithm *off-line* after simulation.
2. Assuming that $\boldsymbol{L}_i(k)=j$ and $\boldsymbol{L}_j(m)=i$, compute $MNN(i,j)$ distances defined as: $MNN(i,j) = m+k$. The maximum MNN distance is less than $2K$.
3. Begin a classical agglomerative clustering algorithm (e.g., *nearest linkage* [35]) with the *linked-lists* concept, starting from the smallest $MNN(i,j)$=2 value.
4. This is terminated upon reaching the greatest value of MNN.

The value of $\mathbf{R}_{clust}$ should be somewhat larger than the spacing between particles in aggregates ($\mathbf{R}_{clust} \approx 0.2\text{-}0.3\,\boldsymbol{r}_{cut}$), and K value should be between 3-8. In Fig.6 we show the clusters of complicated shape from non-linear aggregation process. This event is detected by using MNN algorithm [35].

3.5 Tests for computational efficiency

The FPM code was written in FORTRAN 95 and was implemented on the MPI interface for both the IBM SP and SGI/Origin 3800 platforms. We performed our tests on IBM SP with WinterHawk+ nodes consisting of 4 Power3+/375MHz processors with 4GB of memory per node. For comparison we present the benchmarks for SGI/Origin 3800 system with R14000/500 CPUs.

Our tests were performed in production run mode, sharing communication switch with other users. The maximum number of nodes we use was 8 (4 CPUs per node) both for IBM SP and for SGI/Origin 3800. The timings obtained for parallel jobs we compare with the CPU time measured for the serial version of the FPM code. The periodic rectangular prism was decomposed along z axis (see Fig.1). The test parameters are summarized in Table 1.

In Fig.7a we depict the CPU times and speed-ups obtained for fixed number of particles (M=1,048,576). The speed-ups for parallel runs refer to the CPU time per step per particle measured for the serial version of the FPM code. Each point on the plots represents the average from ten runs and the first 100 timesteps of simulation. The superlinear speed-up observed in Fig.7a results directly from the

cache. For **pfpm0** and **pfpm0_origin** runs, the box is very thin. By increasing the number of processors, the fraction of computations involving cache increases (the number of cache misses decreases). The cache effect is more distinct for IBM SP machine with Power3+ CPUs, whose cache size is smaller (4MB) than that of the R14000 processor (8MB).

Making the computational box wider in *x*,*y* plane and proportionally shorter in z direction (in **pfpm1** the number of cells remains the same as in **pfpm0**) the communication overhead increases proportionally to the increase of the interface area between processor domains. Moreover, the cache misses become more frequent. Because the particles that are the physical neighbors should also be closer one another in the computer memory, to avoid frequent cache misses the particles are renumbered every some period of time. In result the particles residing in the same cell have consecutive numbers. However, the gap between particle numbers still exists for the particles from different cells. This is due to the sequential numbering of particles in domains. Let us assume that the particles are numbered first along x, then y and finally z directions. By increasing 4 times the sizes of computational box in *x*,*y* plane, the gap between particle numbers from the neighboring cells in *z*-direction increases also 4 times. Thus, the respective **r**, **v** and w coordinates of two interacting particles from these cells can be very distant in memory generating cache misses.

For 32 processors we observe a decrease in the speed-up for all the tests. This is caused by the small number of cell layers residing in processor domains and the degradation of computation/communication ratio. When decomposition goes along the shorter side of the box (**fpmy**), this overhead is much larger.

In Fig.7b we compare the two scalable runs performed on IBM SP and SGI/Origin 3800 computers. The number of particles increases proportionally to the number of CPUs, from 500,000 to 16 millions on 32 processors. The computational box increases only in one (*z*) direction. This keeps the communication overhead constant, due to unblocking and bi-directional communication between domains. For larger number of CPUs than 8, we observed the rapid degradation of code performance on IBM SP machine. This may come from communication delay between processors belonging to different IBM SP *frames*, which involve switches between the frames. The network is shared between other users. The machine remains very busy. Thus communication between processors from different frames (supernodes) may be much slower than in a single node or inside the frame.

From Fig.7b we find that our code runs more than 2 times faster on a single R14000/500 processor than on Power3+/375. This effect can be a combination of two factors: greater peak performance of MIPS processor (1 Gflops, i.e., about 30% more than Power3) attained by implementing a new MIPS-IV 64-bits instruction set and aggressive optimization strategy of f90 compiler. We expect, however, that the second factor is crucial in the case of FPM model. As shown in [43], for parallel version of clustering procedure it

appeared that IBM SP is two times faster than SGI/Origin 3800 in spite of a slower clock speed. However, in the case of clustering the code has much modest memory requirements and is much simpler than for FPM model, which involves time consuming floating point computations of interparticle forces and many nested loops.

For the number of nodes greater than 2, the FPM code achieves better scalability with the number of CPUs on SGI/Origin 3800 than on IBM SP. This difference is not observed for parallel clustering [43] involving only 10% of memory requirements and communication bandwidth of those demanded by the FPM code. Therefore, better scalability of FPM code on SGI/Origin 3800 must be the consequence of faster communication between nodes on the SGI machine than on IBM SP. IBM SP is a distributed memory machine, while SGI/Origin 3800 is ccNUMA (cache coherent non uniform memory access) machine with virtual shared memory and with highly optimized distant memory calls. They must be optimized due to the calls to the slow, distant memory are the main source of the overheads on ccNUMA systems. The FPM code, which employs MPI communication interface, uses both the high memory bandwidth of ccNUMA architecture and the procedures which forces locality of the data by placing neighboring domains on neighboring processors. However, for very long boxes, the calls to distant memory from extreme processors, can produce overheads observed in Fig.7b.

5. Simulation results

In the FPM code we employ dimensionless program units collected in Table 2. We set arbitrary partial pressure P - defined in Eq.(12) - divided by the number density n, as a reference point for computing the energy unit δ. Larger value of δ means the greater contribution of thermal fluctuations. The scaling coefficient Π for conservative FPM forces (Eqs.3,18) can be computed directly from Eqs.(12). It is responsible for the compressibility of FPM fluid and is chosen such that the FPM particle system exhibits liquid ordering (see [21]). The scaling factor for dissipative forces (Eq.4,5) is computed from the value of Ω (see Table 2), which stands for the dimensionless kinematic viscosity of FPM particle system [36]. This value represent the ratio of the time taken by a particle covering r_{cut} distance at the thermal velocity c and the time γ^{-1} associated with friction. The value of σ - scaling factor for Brownian forces - is computed from the detailed balance equations Eqs.(8).

We present here sampling simulation results obtained by using FPM parallel code from:

1. Phase separation (symmetric quench) in binary fluid.
2. Dispersion of colloidal slab in an elongated pipe.

4.1 Phase separation

The growth kinetics of binary immiscible fluid and phase separation in two dimensions have been investigated with dissipative particle dynamics by Coveney and Novik [31]) and Dzwinel and Yuen [21]. It was shown that time-dependent growth of average domain size $\boldsymbol{R}(t)$ in two dimensions, follows algebraic growth laws of the form $\boldsymbol{R}(t)=t^b$ where $b=1/2$ for Brownian regime and 2/3 for inertial regime. In the absence of Brownian diffusion of interfaces and droplets the growth proceeds by the Lifshitz-Slyozov mechanism [37] and the power-low index b is set to 1/3.

We have simulated two immiscible FPM particle fluids in 3-D assuming that the particles are perfectly mixed at the beginning of simulation. We define [21] the immiscibility factor to be $DP=P_1-P_{1,2}$ where $P_1=P_2$ are the partial pressures in fluid 1 and 2 respectively. The value of $P_{1,2}$ denotes the pressure computed from Eq.(12) for scaling factor $\Pi_{1,2}$ of conservative forces between two types of particles representing different fluids. Important property of detailed balance for multi-component DPD particle ensemble is satisfied as for one-component system [38]. Here we presume that this is also valid for the FPM. In Table3 we display the principal physical and numerical parameters employed in the simulations.

As we have depicted in Fig.8, in 3-D FPM the lamellar phase resulting from the Lifshitz-Slyozov mechanism (b =1/3) [31,37] can be observed in the initial stages of separation. The lamellas are destroyed quickly by the thermal fluctuations. The value of b changes to 1 for the diffusive regime and b =2/3 for an average domain size greater than hydrodynamic length. From FPM simulation of phase separation - shown in Fig.9 - we have obtained the three regimes and additional one with b =1/2, which was observed before, but only in two dimensions.

4.2 Dispersion of colloidal slab

The principal parameters for the simulations of the slab accelerated in the periodic hexagonal prism, elongated in z direction are presented in Table. 4. The particle system consists of two types of particles with the same size. The particles forming initially a rectangular slab are accelerated in a solvent, which is made up of particles, which are 5 times lighter.

In Fig.10 we present the snapshots from FPM simulations and the results from clustering, which reveal cluster structures creating during dispersion. As shown in Fig.11, this structure changes depending on the viscosity contrast between slab Ω_S and bulk of fluid Ω_S. The slab shape resembles a comet in appearance for Ω_B=10 and Ω_S=100 with the dense cluster in the tip and the tail consisting of smaller structures.

For $\Omega_B=\Omega_S$=100, the head of slab becomes distinctly smaller and clusters create the streaks at the end of the tail. In the case of higher viscosity in the bulk fluid (Ω_B=100 and Ω_S=10), the head of slab

disappears and smaller clusters collectively create long streaks by the large shear developed along the flow field.

5 Conclusions

Fluid particle model (FPM) is a very interesting physical paradigm, which can be used successfully for simulating mesoscopic fluid dynamics. The deployment of fluid particles in realistic 3-D cross-scaling simulations requires resolving a few fundamental issues.

1. **Scale matching** - matching particle interactions to the properties of simulated fluid in the spatio-temporal scale under interest.
2. **Coupling –** combining particles of different types and length-scale (e.g., defining interactions between them)
3. **Scales bridging –** defining the rules of splitting fluid particles into particles from larger to smaller spatio-temporal scale and combining them vice versa. The problem with multiple timestepping should be solved.
4. **Implementation –** efficient numerical implementation of the model in a parallel environment.

In the paper we have discussed the last item. For investigating the structures of multiple scales created in complex fluids, the problems of coupling and bridging can be partly overcome by the bottom-up approach. This approach involves millions of particles and an efficient parallel code for simulating their temporal evolution.

We have proposed here an algorithm for parallel implementation of the fluid particle model (FPM) and we have presented the results of its implementation on the two parallel platforms of different architecture. We have used distributed memory IBM SP machine and ccNUMA SGI/Origin 3800 system. We have shown that due to much greater memory load than in classical parallel MD codes, the optimal use of cache memory becomes crucial for obtaining efficient scalability of the parallel FPM code on the IBM SP machine. Moreover, the slow communication between distant processors – assigned to the extreme domains in an elongated computational box - results in a bad scalability of the code for the number of nodes greater than 2 (i.e. 8 CPUs). The ccNUMA architecture of SGI/Origin 3800 appeared to be more efficient than the IBM SP due to the different organization of the cache and a faster memory access. We have obtained the speedup of 26 on 32 processors of SGI/Origin 3800. Further increase of efficiency can be achieved by optimizing distant memory calls between domains located at the two ends

of the elongated box. Our code executed on R14000/500 processor is more than 2 times faster than on Power3+/375. This result is somewhat surprising in light of a higher peak performance of the Power3+ resulting from its 4-way architecture. In order to speed-up the calculations on IBM SP by reducing cache misses, the particles that are the physical neighbors should also be close to one another in the operational memory. Better scalability can be obtained by renumbering the particles in the neighboring cells and by splitting up the loop in which the forces are evaluated.

However, constant tuning of the code can make it too complicated for rapid modifications because of adjustment to the physics. This can increase considerably the design and testing time. Our parallel code can be employed for production runs involving reasonable computational resources, i.e., up to 32 processors simulating about 20 million particles in 5,000-10,000 timesteps.

The FPM code can be applied for simulating vascular fluid flow in capillary pipes [39] or blood flow in small vessels. A periodic grid of long boxes fits well for modeling the flow in bunch of capillary pipes carrying fluid by employing capillary forces. Blood flow in small vessels during a rapid heart-attack or a rapid stroke developed by deep vein thrombosis require more complicated boundary conditions. The computational boxes should have more complex shapes and also elastic boundaries. In this connection the checker-board boundary conditions are helpful for reducing the superfluous space.

The axisymmetric pulsatile flows and flows subject to acceleration in blood vessels have been investigated both experimentally and numerically for a long time (see e.g. [10,39–42]) by using Navier-Stokes equations with proper substitution of the blood rheological properties. Up to now, there has not been much progress made in the field concerning the flow interactions between the microstructural dynamics and the larger-scale flow. The modelling of the dispersion of drugs and thrombosis along tiny blood vessels demands a completely different approach.

Parallel implementation of the fluid particle model is also a good starting point for simulating the physical and chemical processes involving nano to mesoscale structures, which are essential to critical phenomena that govern the trapping and release of nutrients, contaminants and pathogens, such as anthrax .

Acknowledgments

We thank Professor Dr Jacek Kitowski from the AGH Institute of Computer Science and Dr Dan Kroll from Minnesota Supercomputing Institute for their contribution to this work. Support for this work was provided by the Energy Research Laboratory Technology Research Program of the Office of Energy Research of the U.S. Department of Energy under subcontract from the Pacific Northwest National Laboratory and partly by the Polish Committee of Scientific Reasearch (KBN), AGH Institute of Computer Science and ACK Cyfronet, Kraków, Poland.

6. References

1. Abraham F, Broughton, J, Q, Bernstein, N, Kaxiras, E. 1998. Spanning the Length Scales in Dynamic Simulation. *Computers in Physics* 1998; **12**(6):538-546.
2. Alda W, Dzwinel W, Kitowski J, Moscinski J, Pogoda M, Yuen, D.A. 1998. Complex Fluid-Dynamical Phenomena Modeled by Large-Scale Molecular Dynamics Simulations. *Computers in Physics* 1998; **12**(6):595-600.
3. Beazley D M, Lomdahl P S, Gronbech-Jansen N. Giles R, Tomayo P. 1996. Parallel Algorithms for Short Range Molecular Dynamics. *Annual Reviews of Computational Physics III*. World Scientific:Singapur, 1996; pp.119-175.
4. Holian B L, Ravelo R. 1995. Fracture Simulation Using Large-Scale Molecular Dynamics. *Phys. Rev. B* 1995;**51**(17):11275-11285.
5. Moscinski J, Alda W, Bubak M, Dzwinel W, Kitowski J, Pogoda M, Yuen D. 1997. Molecular Dynamics Simulations of Rayleigh-Taylor Instability, in: *Annual Review of Computational Physics*, **V**, World Scientific:Singapur, 1997: pp.97-136.
6. Nakano A, Bachlechner M E, Campbell T, Kalia R K, Omaltchenko A, Tsuruta K, Vashishta P. 1998. Atomistic Simulation of Nanostructured Materials, *IEEE Computational Sci. and Engineering* 1998. October-December, 68-7.
7. Vashishta P, Nakano A. 1999. Dynamic fracture analysis. *Computing in Science and Engineering* 1999, Sept/October, 20-23.
8. Glowinski R, Pan T W, Hela T I, Joseph D D, Priaux J. 2000. A Fictitious Domain Approach to the Direct Numerical Simulation of Incompressible Viscous Flow Past Moving Rigid Bodies: Application to Particle Flow. *University of Minnesota Supercomputing Institute Research Report*, UMSI 2000/68, April 2000.
9. Singh P, Joseph D D, Hesla T I, Glowinski R, Pan T-W. 2000. A distributed Langrange multiplier/fictitious domain method for viscoelastic particulate flows, *J. Non-Newtonian Fluid Mech* 2000; **91:**165-188.
10. Taylor C A, Hughes T J R, Zarins C K. 1998. Finite element modeling of blood flow in arteries. *Meth.Appl. Mech. Eng.*1998;**158**(1-2):155-196.
11. Chopard B, Droz M. *Cellular Automata Modelling of Physical Systems*. Cambridge University Press, 1998.
12. Rothman D H, Zaleski S. *Lattice-Gas Cellular Automata: Simple models of complex hydrodynamics*. Cambridge University Press, 1997.

13. Hoogerbrugge P J, Koelman JMVA. 1992. Simulating Microscopic Hydrodynamic Phenomena with Dissipative Particle Dynamics. *Europhysics Letters*1992;**19**(3):155-160.
14. Gingold R A, Monaghan J J. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. Astr. Soc*. 1997;**181**:375-389.
15. Libersky L D, Petschek A G, Carney T C, Hipp J R, Allahdadi F A. 1993. High Strain Lagrangian Hydrodynamics, *J. Comp. Phys.*1993;**109**(1):67-73.
16. Bird G A. *Molecular Dynamics and the Direct Simulation of Gas Flow*. Oxford Science Publications: Oxford, 1994.
17. Español P. 1998. Fluid particle model. *Physical Review E* 1998; **57**(3):2930-2948.
18. Dzwinel W, Alda W, Yuen D, A. 1999. Cross-Scale Numerical Simulations Using Discrete Particle Models. *Molecular Simulation* 1999; **22**:397-418.
19. Dzwinel W, Yuen D A. 2001. Dispersion of Colloidal Agglomerate Modelled by a Hybrid Fluid Particle Model. *University of Minnesota Supercomputing Institute Research Report UMSI* 2001/23, accepted for publication in *J. Colloid and Interface Sci.*, October 2001.
20. Dzwinel W, Yuen D A. 2001. Mixing Driven by Rayleigh-Taylor Instability in the Mesoscale Modeled with Dissipative Particle Dynamics. *International J. of Modern Physics C* 2001;**12**(1):91-118.
21. Dzwinel W, Yuen D A. 2000. Matching macroscopic properties of binary fluid to the interactions of dissipative particle dynamics. *International Journal of Modern Physics C* 2000;**11**(1):1-25.
22. Dzwinel W, Yuen D A. 2000. A two-level, discrete-particle approach for simulating ordered colloidal structures. *J. Colloid Interface Science* 2000; **225**(1):79-190.
23. Dzwinel W, Yuen D A. 1999. Dissipative particle dynamics of the thin-film evoluation in mesoscale. *Molecular Simulation* 1999;**22**:369-395
24. Larson R G. *The structure and Rheology of Complex Fluids*. Oxford University Press:New York 1999.
25. Perkold K, Rappitsch G. 1995. Computer simulation of local blood flow and vessel mechanics in a compliant carotid artery bifurcation model. *J.Biomech*.1995;**28**:845-856.
26. Marsh C, Backx G, Ernst M H. 1997. Static and dynamic properties of dissipative particle dynamics, *Physical Review E* 1997; **56:**1976.
27. Yserentant A. 1997. A new class of particle methods, *Numerische Mathemetik* 1997;**76**:87-109, (1997)
28. Hockney R W, Eastwood J W. *Computer Simulation Using Particles*, McGraw-Hill Inc.1981.
29. Rapapport D C. *The art of molecular dynamics simulation.* Cambridge Univ.Press:Cambridge, 1995.

30. Haile P M. *Molecular Dynamics Simulation*. Wiley:New York, 1992.
31. Coveney P V, Novik K E. 1996. Computer simulations of domain growth and phase separation in two-dimensional binary immiscible fluids using dissipative particle dynamics. *Physical Review E* 1996;**54**(5):5134-5141.
32. Adams D,J. 1983. Alternatives to the periodic cube in computer simulation. *CCP5 Information Quarterly for Computer Simulation of Condensed Phases* 1983; **10**:30 (Informal Newsletter, Daresbury Laboratory, England).
33. Smith W. 1989. The minimum image convention in non-cubic MD cell. *CCP5 Information Quarterly for Computer Simulation of Condensed Phases* 1989;**30**:35 (Informal Newsletter, Daresbury Laboratory, England).
34. Dzwinel W, Kitowski J, and Moscinski J. 1991. „Checker board" periodic boundary conditions in molecular dynamic codes. *Molecular Simulation* 1991;**7**:171-179.
35. Jain D, Dubes R C. 1988. *Algorithms for Clustering Data*. Prentice-Hall Advanced Reference Series, 1988.
36. Español P, Serrano M. 1999. Dynamical regimes in DPD. *Phys.Rev.E* 1999;**59**(6):6340-7.
37. Gonnella G, Orlandini E, Yeomans J M.1997. Spinodal Decomposition to a Lamellar, Phase: Effects of Hydrodynamic Flow. *Phys. Rev. Lett.* 1997;**78**(9):1695-1698.
38. Coveney P V, Español P. 1997. Dissipative particle dynamics for interacting multicomponent systems. *J.Phys.A:Mathematical and General* 1997; **30**:779-784.
39. Quarteroni A, Tuveri M, Veneziani A. 2000. Computational vascular fluid dynamics: Problems models and methods. *Comput. Visualization Sci* 2000;**2**:163-197.
40. Gueraoui K, Hammoumi A, Zeggwagh G.1998. A theoretical model of pulsatile flow of an inelastic fluid through anisotropic porous viscoelastic pipes. *Computes Rendus de l Academie des Sciences Serie II Fascicule B-Mecanique Physique Chimie Astronomie* 1998;**326**(9):561-8.
41. Latinopoulos P, Ganoulis J. 1982. Numerical simulation of pulsatile flow in constricted axi-symmetric tubes. *Applied Mathematical Modelling* 1982;**6**(1):55-60.
42. Misra J C, Sahu B K. 1988. Flow through blood vessels under the action of a periodic acceleration field: a mathematical analysis. *Computers & Mathematics with Applications* 1988;**16**(12):993-1016.
43. Boryczko K, Dzwinel W, Yuen D.A. 2001. Parallel Extraction and Visualization of Clusters from Large-Scale Data Sets, *University of Minnesota Supercomputing Institute Research Report, UMSI 2001/87*, submitted for publication in *Parallel Computing*

Figure captions

Fig.1 The box decomposition onto cells and processor domains.

Fig.2 The sequence of computation and communication procedures invoked in FPM parallel code.

Fig.3 Dimensionless pressure $\delta=k_BT/(P/n)$ and thermodynamic temperature Tth of the FPM particle system in 3-D (number of particles $M=1.3\times10^5$) with time. The initial values for $\delta=0.021$ (in dimensionless units) and for an assumed temperature $T=100$ K.

Fig.4 a) Checker-board periodic boundary conditions in 2-D. Computational box simulating the hexagonal periodic boundary conditions along with its replicas are depicted.

b) Checker board periodic boundary conditions in 3-D. Computational box of side lengths Lx=1, Ly=1 and Lz=2 represents periodic boundary conditions for rhombic dodecahedron (see [34]). The domain decomposition onto CPU units is shown. The arrows show the communication paths among the processors.

Fig.5 The largest circle inscribed in a hexagonal box compared to the largest circle inscribed in a square box of the same area.

Fig.6 The result of clustering procedure detecting colloidal agglomerates in 2-D. The largest cluster is shown in black.

Fig.7 Speed-ups and CPU time per step per particle for benchmark on IBM SP and SGI/Origin 3800.

Fig.8 Two snapshots from 3-D FPM simulation of phase separation in binary fluid involving 8 million FPM particles. Cross-sections are depicted. A) lammelar phase at t=400 ($\beta=1/3$) B) the regime with $\beta=1/2$, at t=4000.

Fig.9 The growth of average domain size in time in symmetric quench.

Fig.10 The snapshots from FPM simulation of a slab accelerated in the particle fluid. Only the slab is shown. The contrast in viscosity between slab and solvent is 10:1. The pictures from a) comes from Data Explorer and show droplet's surface. In the following figures the raw particle positions are shown. Multiresolution structures are detected from the clustering procedure. The light gray tip of slab is the largest cluster extracted. The blue particles create the smallest clusters (consisting of at most 2 particles). The red particles represent medium scale clusters, i.e., the streaks are created due to shear. In solvent (c) low density cluster is shown in blue. This situation can be applied to flow in narrow blood vessels.

Fig.11 Different viscosity contrast between solvent Ω_B and a slab Ω_S. a) $\Omega_B=10$, $\Omega_S=100$ b) $\Omega_B=\Omega_S=100$ c) $\Omega_B=100$, $\Omega_S=10$. The first three pictures come from Data Explorer and show clusters surface. The pictures on the right depict the multi-scale features extracted with the clustering MNN algorithm. These situations can be applied to flow in narrow blood vessels.

Tables

Table 1. Parameters of the efficiency tests. Number of particles in a single unit cell is equal to 4.

Test	**Number of particles M**	**Namber of cells in x,y and z direction**	**Number of processors $P=2^k$**	**Platform**
Serial FPM, **sfpm**	1,048576	64×64×64	1	IBM SP, Origin 3800
Parallel FPM **pfpm1**	1,048576	64×64×64	2-32	IBM SP
Parallel FPM **pfpm0** **pfpmp_origin0**	1,048576	32×32×256	2-16	IBM SP, Origin 3800
Parallel FPM **pfpmy**	1,048576	32×256×32	2-16	IBM SP
Scalable parallel FPM **spfpm_ibm2-4-8-16-32**	1,048576 to 16,777216	64×64×64×P/2	2-32	IBM SP
Scalable parallel FPM **spfpm_origin2-4-8-16-32**	1,048576 to 16,777216	64×64×64×P/2	2-32	Origin 3800

Table.2. Program units

VALUE	**UNIT**
Length l	the average distance between the neighboring fluid particles $l = 1$
Mass m	dimensionless - mass of the lightest fluid particle $m = 1$
Time Δt	in $t_{ref} = l/c$ where $c^2 = k_B T/m$ l -unit of length
Energy δ	in $k_B T/(P/n)$ where P-partial pressure defined in Eq.(12)
Viscosity Ω	$\Omega = gr_{cut}/3 \cdot c$ where γ is the scaling factor of dissipative forces

Table.3. Principal physical and numerical parameters employed in FPM simulation of phase separation in 3-D

General parameters	Values
$\mathbf{k_B T}$ /(P/n)	0.015
Number density n	1.0 (per cube of volume l^3)
Viscosity (in Ω)	25
$D\boldsymbol{P/P=P_1-P_{1,2}/P}$	5%
Number of particles	8.2×10^6
D**t** (in t_{ref})	0.01
Cut-off radius r_{cut} (in l)	2.0
Computational box	periodic rhombic dodecahedron
Box size (in cells)	80×80×160

Table.4. Principal physical and numerical parameters employed in FPM simulation of dispersion of colloidal slab.

General parameters	Values
$\mathbf{k_B T}$ /(P/n)	0.015
g ($l/\Delta t^2$) (accel.)	10
Particle masses	$m_{SLAB}=5$, $m_{SOLVENT}=1$
Number density n	1.0 (per cube of volume l^3)
Viscosity (in Ω)	10 and 100
$D\boldsymbol{P/P=P_1-P_{1,2}/P}$	5%
Number of particles:	*Total:* 1.76×10^6
	Solvent: 1.49×10^6 *Slab:* 171,000
D**t** (in t_{ref})	0.01
Cut-off radius r_{cut} (in l)	1.58
Computational box	periodic hexagonal prism
Box size (in cells)	44×50×200

Figures

Fig.1

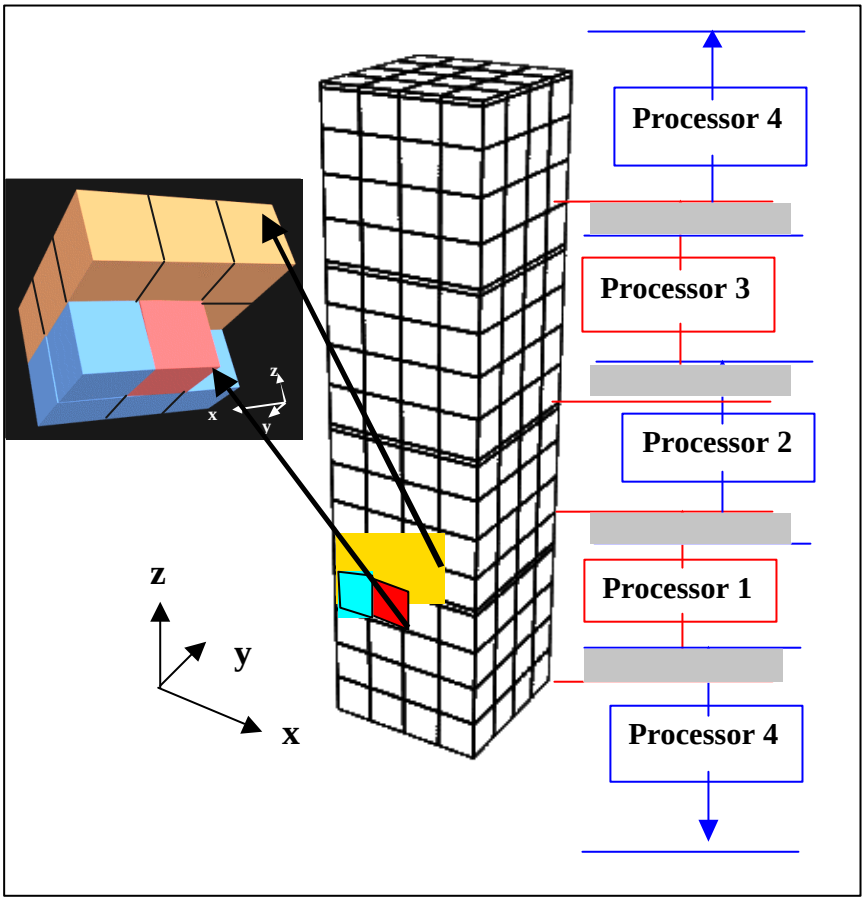


Fig.2

PARALLEL LOOP

For all P = 1, N

SENDfrom_P-1_to _P_particles_from_boundary_cells **($\vec{\mathbf{r}}$, ?, ? , Nkind);**

Compute_INTERACTIONS_in_P **($\vec{\mathbf{F}}_{ij}$, $\vec{\mathbf{N}}_{ij}$);**

SENDfrom_P_to_P-1_reactions_on_boundary_particles_in-P-1 **($\vec{\mathbf{F}}_{ij}$, $\vec{\mathbf{N}}_{ij}$);**

Update_INTERACTIONS_for_boundary_particles_in-P-1 **($\vec{\mathbf{F}}_{ij}$, $\vec{\mathbf{N}}_{ij}$);**

MOVE_particles_in_P **($\vec{\mathbf{r}}$, ?, ?);**

SENDfrom_P_to_P-1_outcoming_particles **($\vec{\mathbf{r}}$, ?, ? , Nkind);**
SENDfrom_P-1_to_P_incoming_particles **($\vec{\mathbf{r}}$, ?, ? , Nkind);**
SEND_totals_from_P_to_master_processor **(Virial, Ekin, *etc.*);**

endfor

Fig.3

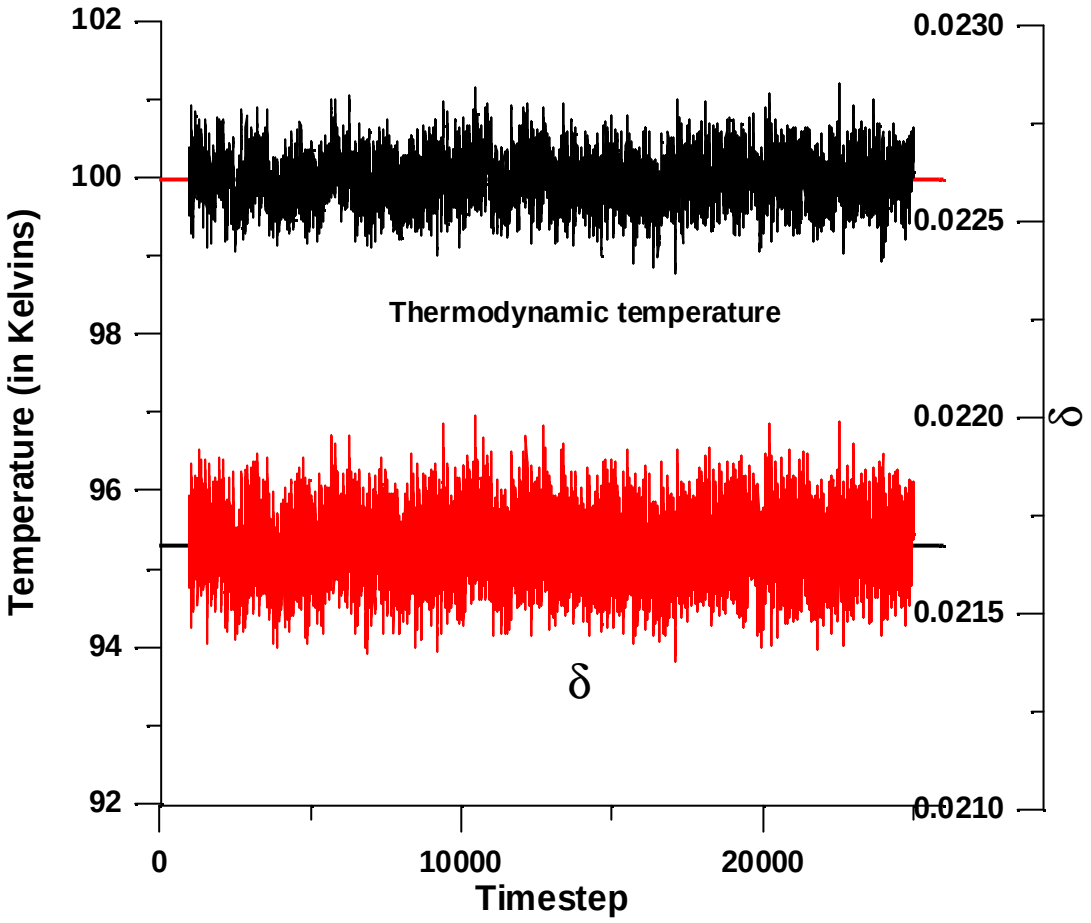
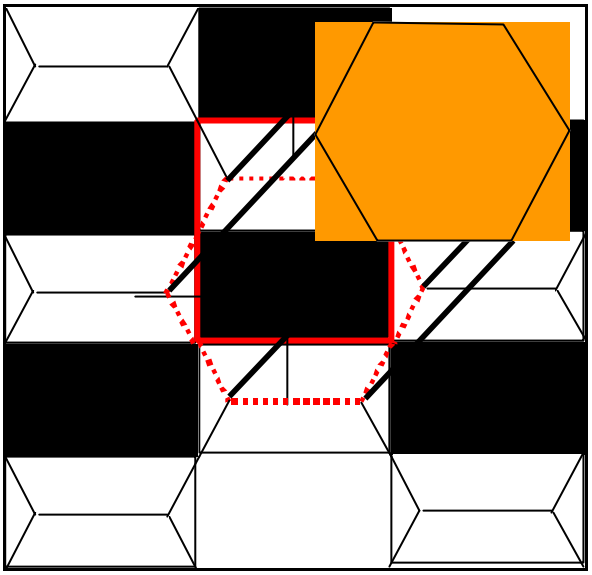

Temperature (in Kelvins)
102
100
98
96
94
92
0.0230
0.0225
0.0220
0.0215
0.0210
δ
Thermodynamic temperature
δ
0
10000
20000
Timestep

Fig.4



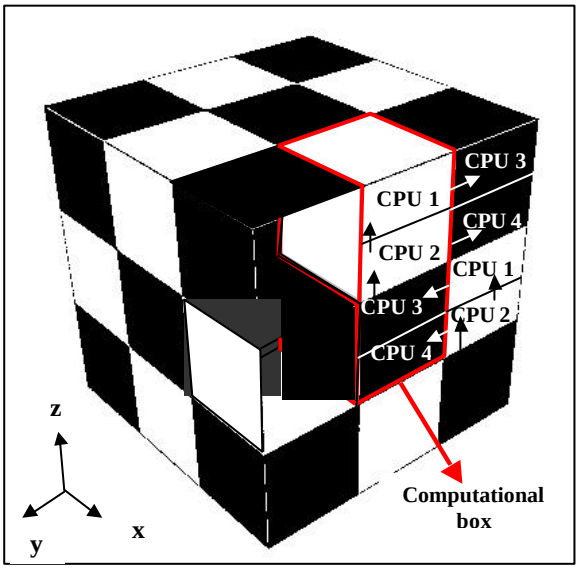

CPU 3
CPU 1
CPU 4
CPU 2
CPU 1
CPU 3
CPU 2
CPU 4
z
x
y
Computational box

Fig.5

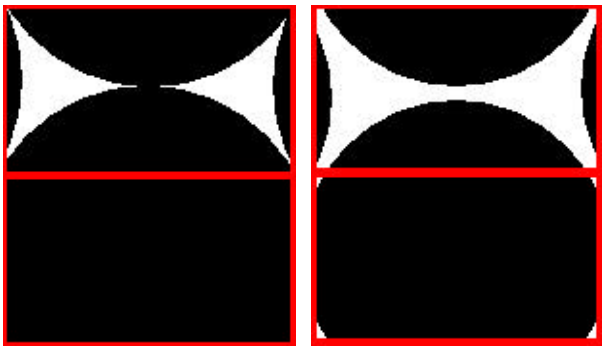


Fig.6

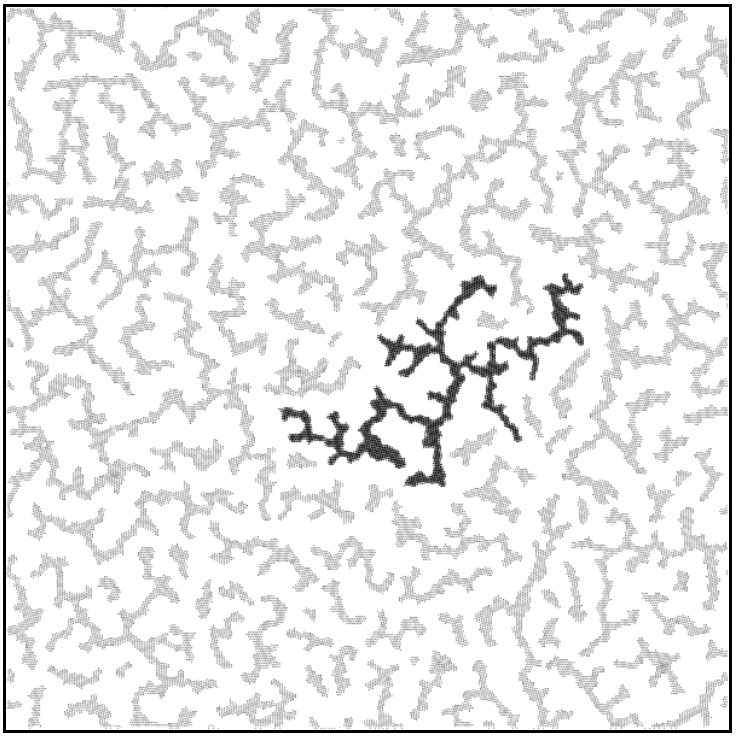


Fig.7a

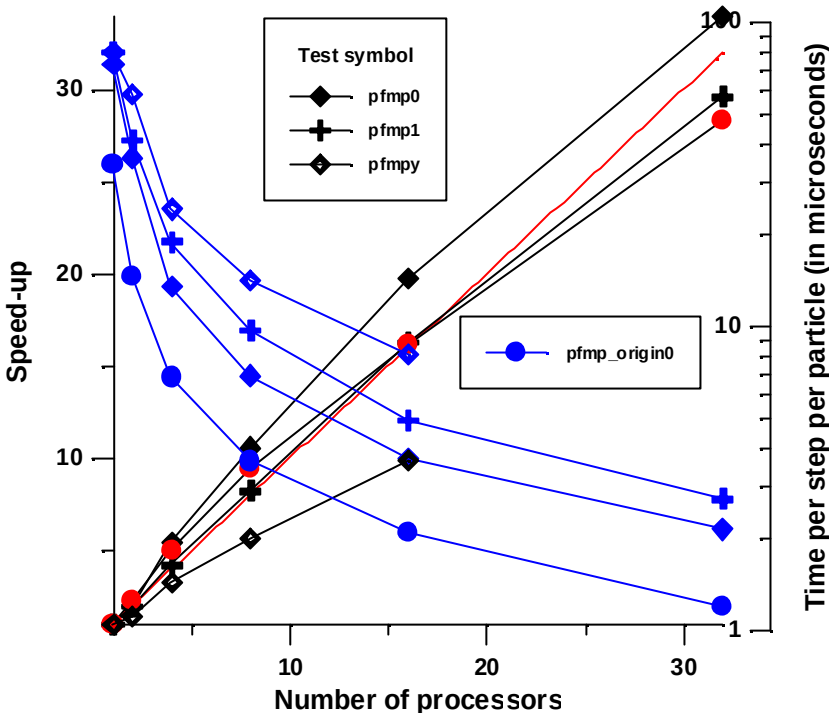

Test symbol
pfmp0
pfmp1
pfmpy
pfmp_origin0
Speed-up
Time per step per particle (in microseconds)
Number of processors
30
20
10
100
10
1
10
20
30

Fig.7b

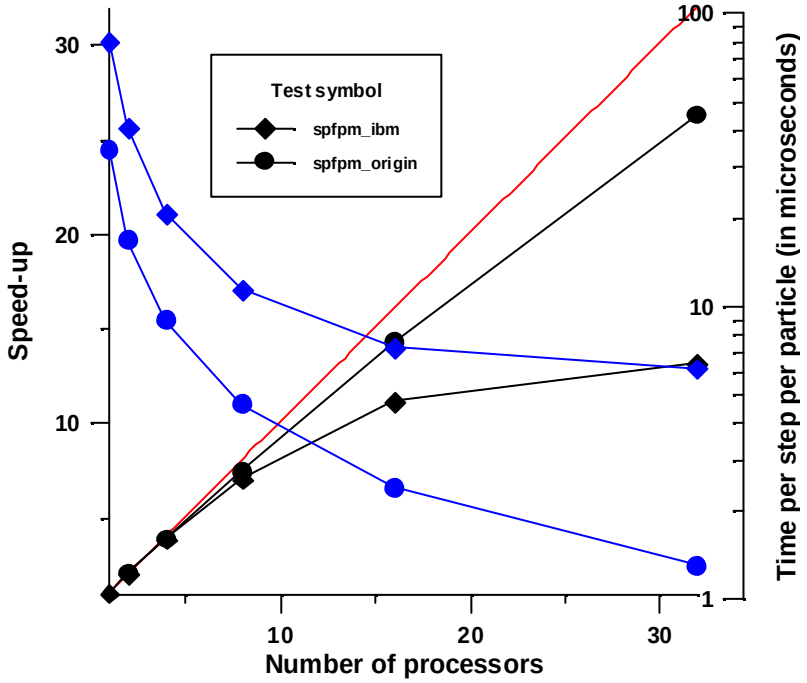
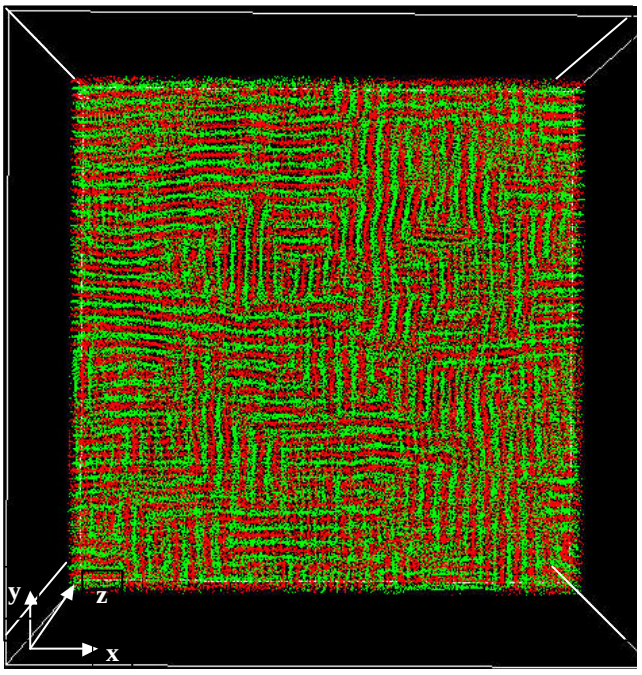

Test symbol
spfpm_ibm
spfpm_origin
Speed-up
Time per step per particle (in microseconds)
Number of processors
30
20
10
100
10
1
10
20
30

Fig.8



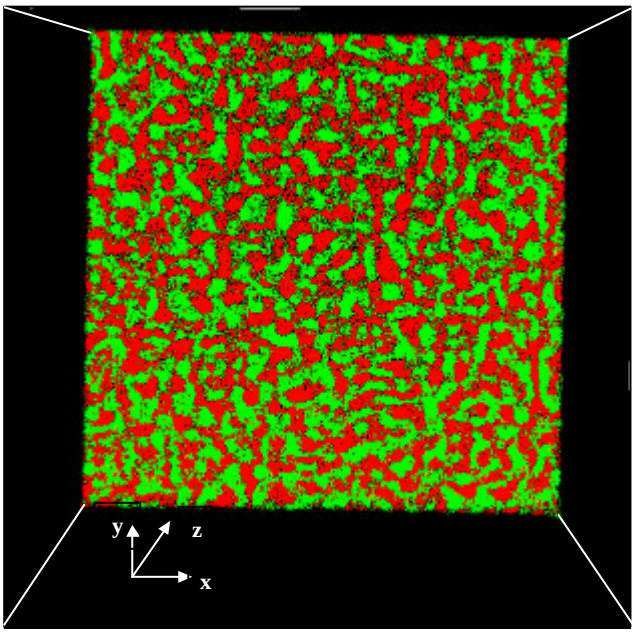


Fig.9

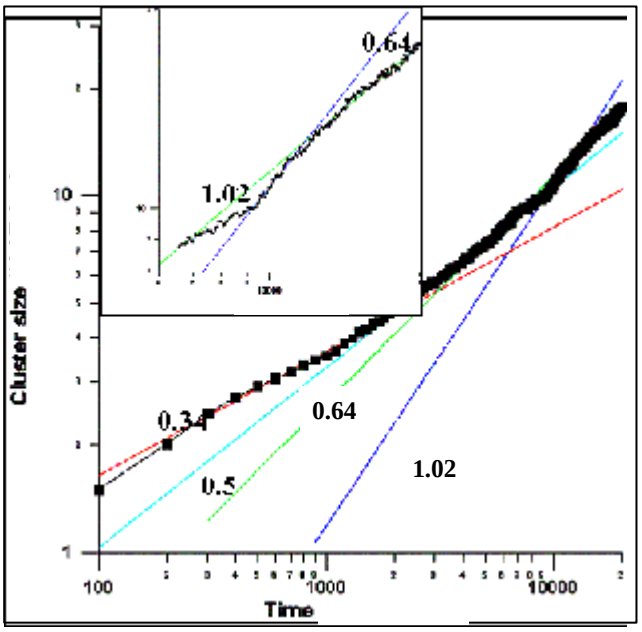


Fig.10

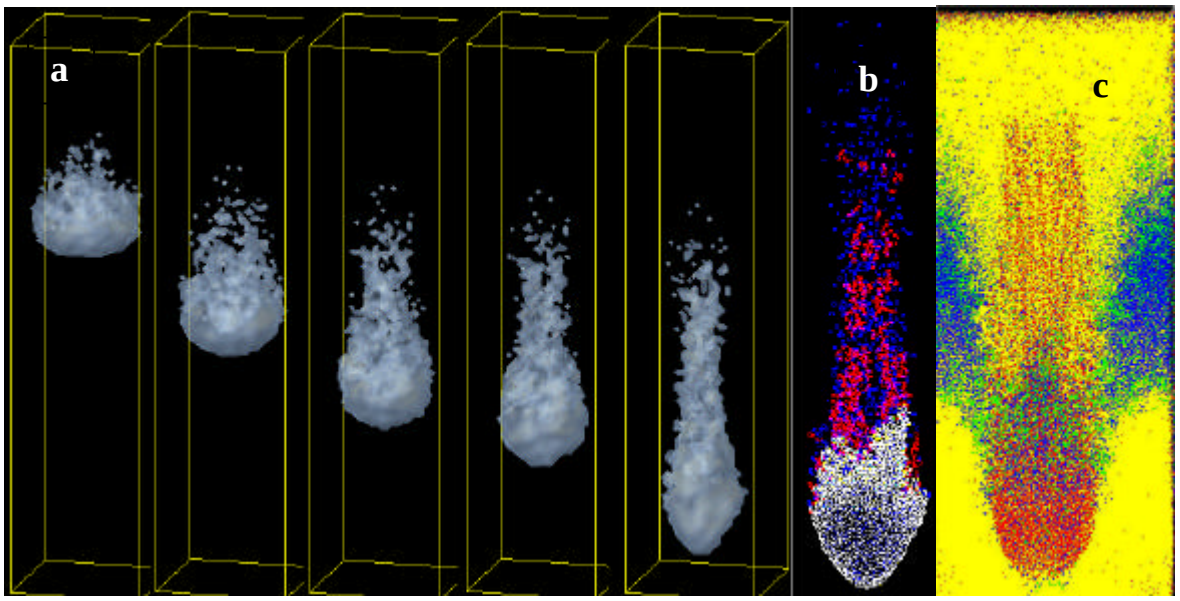
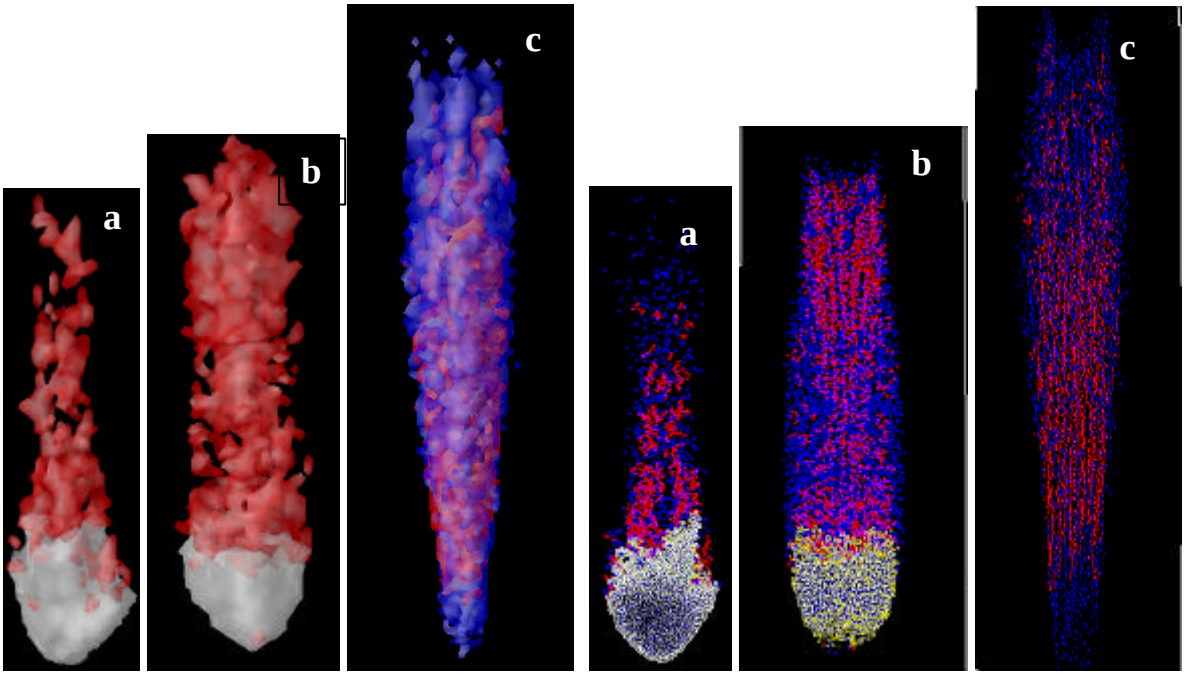


Fig.11



About authors

David A. Yuen is a professor of geophysics and scientific computation at the University of Minnesota, Twin Cities and a fellow of the Minnesota Supercomputing Institute. He is interested in modelling of all scales, ranging from microscopic to large-scale mantle convection and fluid dynamics problems. He is also working on problems related to multidimensional wavelets and 3-D feature extraction using wavelets.

Witold Dzwinel is a professor of computer science at the University of Mining and Metallurgy (AGH), Institute of Computer Science, Kraków, Poland. His main fields of interest are simulations using particles, natural solvers and evolutionary systems. He is also working on application of pattern recognition systems in diagnostics of nuclear reactors and in geophysical prospecting.

Krzysztof Boryczko is an assistant professor at the University of Mining and Metallurgy (AGH), Institute of Computer Science, Kraków, Poland. He is a specialist in parallel programming and large-scale computing. He is interested in application of scientific visualization systems in feature extraction and clustering data from large-scale simulations with particles.