V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

RESOLVING IMPLEMENTATION ISSUES IN THE HYBRID SIMULATION OF TURBULENT COLLISION OF CLOUD DROPLETS

Bogdan Rosa*, Hossein Parishani†, Orlando Ayala†, Lian-Ping Wang† and Wojciech Grabowski‡

*Institute of Meteorology and Water Management,
61 Podlesna St, Warsaw 01-673, Poland
e-mail: bogdan.rosa@imgw.pl

†University of Delaware,
130 Academy St, Newark, DE 19716-3140, USA
e-mail: {hparish, omayalah, lwang}@udel.edu

‡National Center for Atmospheric Research
PO Box 3000, Boulder, Colorado 80307-3000, USA
e-mail: grabow@ncar.ucar.edu

Key words: turbulent collision-coalescence, aerodynamic interactions, cloud droplets, direct numerical simulation, MPI implementation

Abstract. *Collision-coalescence of cloud droplets in a turbulent air flow is an essential step for warm rain precipitation, therefore, the parameterization of turbulent collision-coalescence rate of cloud droplets is central to the modeling of cloud dynamics in particular and to weather prediction in general. In the last few years, we have developed a hybrid direct numerical simulation approach (Ayala et al. 2007, Wang et al. 2009)[1,2], to simulate the collision of cloud droplets in a turbulent background flow and to incorporate local aerodynamic interaction of cloud droplets. The present paper extends our previous work in several directions. First, we have completed the MPI implementation and validated the results of the new MPI HDNS code. Second, in order to consider a polydisperse suspension of cloud droplets, we combine the direct numerical integration of droplet equation of motion for finite droplet inertial response time and an asymptotic representation for small inertial response time (Maxey 1987)[3]. It was shown that the results are consistent with those obtained by direct integration. Third, to prepare for next-generation scalable supercomputers, a new FFT interface is being developed, using 2D domain decomposition strategy. The scalability of our new FFT interface was found to be quite promising. These advances are being used to perform HDNS for large domain size and for a polydisperse droplet size distribution.*

1 INTRODUCTION

Reliable weather and climate prediction at both local and global scales depends on our understanding of microphysical processes (i.e., droplet activation, condensational growth, collision-coalescence growth, and drop breakup) and the small-scale cloud dynamics (i.e., entrainment, mixing, multiscale turbulent transport, and thermodynamics). Here we are interested in developing a quantitative approach for turbulent collision-coalescence of cloud droplets – the last step to warm rain initiation. In recent years, increasing evidence has been accumulated suggesting that air turbulent could significantly enhance the growth of cloud droplets by collision-coalescence. A significant progress in this direction has been made through the computational approach known as the direct numerical simulation (DNS). DNS of small-scale air turbulence has played a central role in our understanding of turbulent collision-coalescence of cloud droplets [4,5,6].

DNS is a bottom up approach of turbulent flow. In DNS, turbulent air motion at the dissipation-range scales (mm to cm scales) and a limited range of inertial-subrange scales – currently up to $\mathcal{O}(10$ cm) – are resolved, but larger-scale motion is represented by a forcing scheme. In the last few years, our group has developed a hybrid direct numerical simulation (HDNS) approach to simulate the collision rate of cloud droplets in a turbulent air where both the inertia and sedimentation of cloud droplets are considered[1]. This approach extends the usual point-particle based simulation[2] to include the effect of droplet-droplet aerodynamic interactions so that the collision efficiency between cloud droplets in a turbulent flow can be studied.

One of the limitations of this HDNS so far is the limited range of resolved flow length scales (or equivalently the small computational domain size). As the droplet size increases, so does the inertial response time of the droplets. A larger range of scales of flow motion may contribute to the dynamics of cloud droplets. Therefore, it is desirable to include more inertial range scales, or equivalently, to increase the computational domain size from $\mathcal{O}(10$ cm) to, say, $\mathcal{O}(1$ m). Increasing the domain size implies that a lager number of cloud droplets will need to be simulated. It is then necessary to make use of a larger number of processors on a distributed memory machine. The first part of this paper will address the MPI implementation of the HDNS code, to prepare for higher resolution simulations.

In recent years, experimental measurements of statistics of inertial particles and droplets have been made to understand clustering and sedimentation of particles in a turbulent flow, as well as to probe into relative motion and collision of inertial particles. Advances in this direction provide opportunities to compare results from our HDNS with those from experimental observations. One complication in the experiment is that the droplets are polydisperse containing a range of different sizes. Representing a range of droplet sizes in HDNS calls for a more efficient algorithm for treating small droplets, which will be discussed in the second part of this paper.

Currently, our HDNS code is based on 1D domain decomposition for parallel implementation. This limits the maximum number of processors that the code can utilize. To

prepare for future petascale computing, we must consider a more flexible domain decomposition such as 2D and 3D domain decomposition. The third part of the paper will explore the use of 2D domain decomposition, starting from FFT of 3D field data.

These issues may seem to be somewhat disconnected, but together they will lead to a much more flexible HDNS code that can eventually be used to address a variety of multiscale coupling questions in cloud microphysics. Our ultimate goal in the near future is to be able to simulate $\mathcal{O}(10^6) \sim \mathcal{O}(10^7)$ droplets suspended in a turbulent flow of Taylor microscale Reynolds number R_λ of the order of 200 to 500. This will allow us to extend our HDNS data on kinematic pair statistics such as radial distribution function and radial relative velocity so a better analytical parameterization of the turbulent collision kernel can be developed.

2 HYBRID DNS

In this section, we will focus on the MPI (Message Passing Interface) implementation of aerodynamic interactions of cloud droplets moving in turbulent flow using HDNS approach. The basic ideas and algorithms for the HDNS approach have been presented in [1,2]. The previous implementation developed by Ayala *et al.*[1] was based on loop-level parallelization using OpenMP. Such approach assumes shared memory and is limited to the use of processors (typically 32 or less) within the same computational node. In terms of the problem size we are dealing with, parallelization with OpenMP could handle the problem efficiently up to 128^3 flow grid resolution with $\mathcal{O}(10^5)$ droplets, giving a maximum Reynolds number of $R_\lambda = 72.4$, λ being the Taylor-microscale of the background turbulence. In order to tackle problems with a larger number of droplets or flows at higher Reynolds numbers, we parallelize the code using MPI. The MPI parallelization does not pose a limitation on the maximum number of processors that can be utilized. Therefore, a higher speed up could be achieved, this along with the larger available memory will make it possible to treat larger problem sizes.

For MPI implementation, we decompose the cubic computational domain into slabs in the direction perpendicular to gravity in order to minimize the number of droplets crossing the slab boundaries. This will in turn reduce the amount of time spent in data communication between the slabs. Since this is a spatial decomposition, we will make use of MPI to communicate data between the slabs when droplets are crossing the boundaries or whenever data must be communicated between processes.

We consider a dilute suspension of droplets in a background turbulent air flow $\mathbf{U}(\mathbf{x}, t)$ solved by the usual pseudo-spectral method in a periodic domain. The flow is resolved with N^3 grid points. Then, the fluid velocity at the location of the $k-$th droplet of radius $a^{(k)}$, denoted by $\mathbf{U}(\mathbf{Y}^{(k)}(t), t)$, will be interpolated from the grid using the 6-point Lagrangian interpolation in each spatial direction, where $\mathbf{Y}^{(k)}(t)$ is the location of that droplet. The velocity of the k-th droplet will be denoted by $\mathbf{V}^{(k)}(t)$.

The disturbance velocity at the center of $k-$th droplet due to other droplets in the system is denoted by $u^{(k)} \equiv \mathbf{u}(\mathbf{Y}^{(k)}(t))$, which must be solved from a coupled linear

system[1] of dimension $3N_p$, as shown by Eq. (1) in Table 1. Note that Eq. (1) is derived from the requirement that the composite flow field (background flow plus the disturbance flow fields caused by droplets) satisfies, on average, the no-slip boundary condition on the surface of each droplet. It is evident that the disturbance velocity felt by a given droplet depends on the background turbulent flow, and positions and velocities of all other droplets in the system. Other equations needed to describe the motion of droplets are summarized in Table 1. The most computationally demanding step in the HDNS approach is to solve Eq. (1). In MPI implementation each processor is assigned to compute the RHS of equation (1) for the droplets inside its own slab and when it is needed, data associated with droplets located in the neighboring slabs can be obtained by non-blocking communication (MPI_Isend and MPI_Irecv).

Table 1: Key equations for the droplets in the HDNS approach (from top to bottom): (1) the linear system of dimension $3N_p$ used to solve for the disturbance velocities felt by droplets; (2) the definition of Stokes disturbance flow; (3) The viscous drag experience by the k-th droplet, (4) the equation of motion, and (5) the kinematic equation for droplet location.

$$\mathbf{u}^{(k)} = \underbrace{\sum_{m=1}^{N_p}}_{m\neq k} \mathbf{u}_S\left(\mathbf{Y}^{(k)}(t) - \mathbf{Y}^{(m)}(t); a^{(m)}, \mathbf{V}^{(m)} - \mathbf{U}(\mathbf{Y}^{(m)}, t) - \mathbf{u}^{(m)}\right), \qquad k = 1, 2, ..., N_p \tag{1}$$

$$\mathbf{u}_S(\mathbf{r}^{(k)}; a^{(k)}, \mathbf{V_p}^{(k)}) = \left[\frac{3}{4}\frac{a^{(k)}}{r^{(k)}} - \frac{3}{4}\left(\frac{a^{(k)}}{r^{(k)}}\right)^3\right]\frac{\mathbf{r}^{(k)}}{(r^{(k)})^2}(\mathbf{V_p}^{(k)} \cdot \mathbf{r}^{(k)}) + \left[\frac{3}{4}\frac{a^{(k)}}{r^{(k)}} + \frac{1}{4}\left(\frac{a^{(k)}}{r^{(k)}}\right)^3\right]\mathbf{V_p}^{(k)} \tag{2}$$

$$\mathbf{D}^{(k)}(t) = -6\pi\mu a^{(k)}\left[\mathbf{V}^{(k)}(t) - \left(\mathbf{U}(\mathbf{Y}^{(k)}(t), t) + \mathbf{u}^{(k)}\right)\right] \tag{3}$$

$$\frac{d\mathbf{V}^{(k)}(t)}{dt} = -\frac{\mathbf{V}^{(k)}(t) - \left(\mathbf{U}(\mathbf{Y}^{(k)}(t), t) + \mathbf{u}^{(k)}\right)}{\tau_p^{(k)}} - \mathbf{g} \tag{4}$$

$$\frac{d\mathbf{Y}^{(k)}(t)}{dt} = \mathbf{V}^{(k)}(t) \tag{5}$$

The computation of the summations in equation (1) is expensive, since for every droplet this sum should be carried out over all other droplets in the domain. In other words, the computational cost for (1) grows as ${N_p}^2$. As long as the collision efficiency is concerned, Ayala *et al.*[1] showed that we could truncate the summation and restrict the radius of influence of the droplet disturbance velocity to $d/a^{(k)} = H_{trunc}$ without a significant effect on the computed collision efficiency. Their experiments with $d/a^{(k)} = H_{trunc}$ showed

that the computed collision efficiency is insensitive to H_{trunc} if H_{trunc} is larger than 35. In this paper we will use dimensionless truncation radius of $H_{trunc} = 50$ which is more conservative than $H_{trunc} = 35$.

In order to efficiently locate the droplets and their immediate neighbors inside the truncation sphere, we make use of cell-index method and the concept of linked lists[7]. This is numerically implemented by dividing the slabs into smaller cells and keeping track of droplet indices inside each cell. This speeds up the process of finding neighboring droplets and it does not affect the converged solution of system (1).

When the disturbance velocities $\mathbf{u}^{(k)}$ are computed, droplets are advanced by solving their equation of motion, Eq. 5.

2.1 Results

In the study of droplet collision and coalescence, one is usually interested in the dynamic collision kernel which is defined as:

$$\Gamma = \frac{\dot{N} V_{box}^2}{n_{pairs}} \tag{6}$$

where n_{pairs} is the total number of droplet pairs, $\dot{N}$ is the number of collisions per unit time per unit volume. For a monodisperse system of N_p droplets, $n_{pairs} = N_p(N_p - 1)/2$, while for a bidisperse system of N_{p_1} size-1 droplets and N_{p_2} size-2 droplets, $n_{pairs} = N_{p_1} N_{p_2}$. In our HDNS code, collision kernel Γ can be determined dynamically by counting the number of collisions between droplets and using equation 6 for different types of colliding pairs (i.e., 1-2, 1-1, 2-2 pairs).

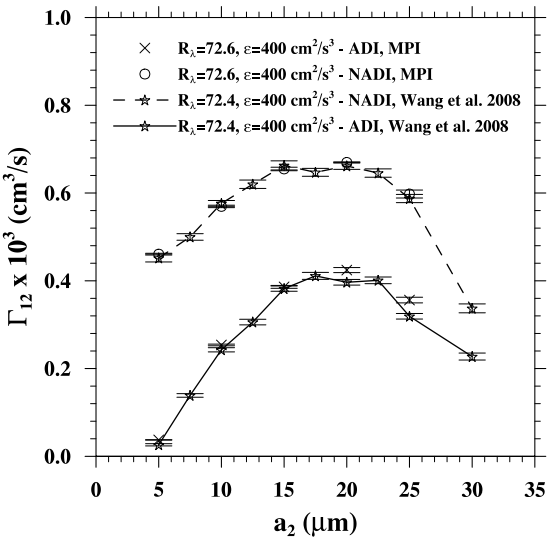

Figure 1: Dynamic collision kernel from MPI implementation compared with the results from previously developed OpenMP code[6]

Furthermore, kinematic pair statistics are also of interest, such as the average radial relative velocity $\langle |w_r| \rangle (r = (a_1 + a_2))$ and the radial distribution function $g_{12}(r = (a_1 + a_2))$ [8]. More details regarding physical and mathematical aspects of collision kernel, radial distribution function and radial relative velocity can be found in[6]. When aerodynamic interactions are considered, non-overlapping corrections for both kinematic properties are needed as explained in [8].

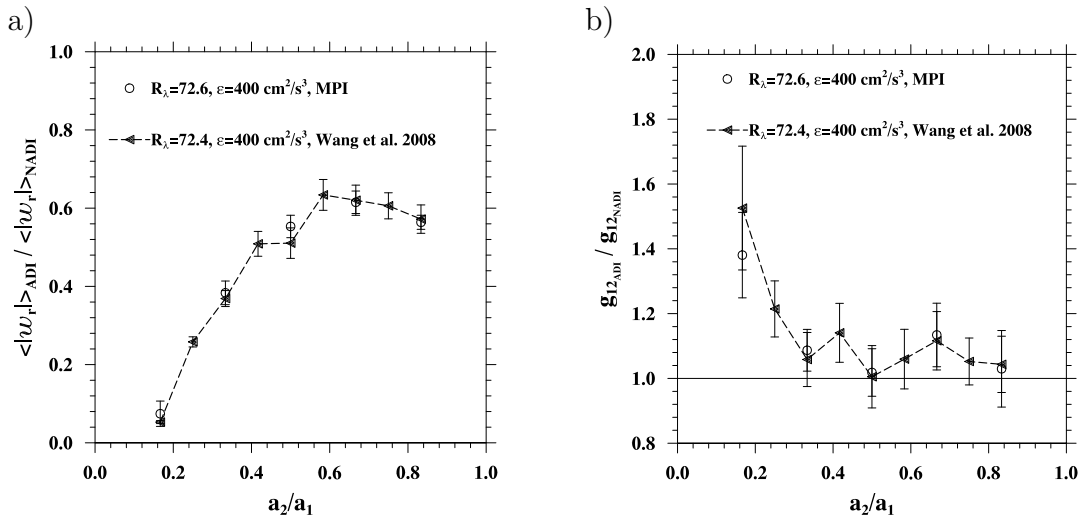

Figure 2: *a*) Radial relative velocity compared with the results from previously developed OpenMP code[6] *b*)Radial distribution function from ADI1 compared with the results from previously developed OpenMP code[6]

More simulations are underway to extend the current collision efficiency, collision kernel, RDF and relative velocity tables to higher R_λ and a wider range of droplet-size combinations.

3 EFFICIENT INTEGRATION SCHEME FOR TRACKING LOW INERTIA PARTICLES MOVING IN TURBULENT FLOW

Integration of trajectories of a large number ($\geq \mathcal{O}(10^6)$) of particles moving in turbulent flow is a computationally intensive task. In order to preserve high computational efficiency, time step size has to be set optimally. There are at least three different issues which introduce limitations on the time step length. The first is related to handling collisional interaction. Too large time step will not allow to capture of all collisional events, particularly it will not allow an accurate computation of kinematic properties of the suspension as for example relative velocity or radial distribution function. The second issue is related to the interaction of the particles with the background turbulent flow. This restriction strictly depends on particle response time and nature of the flow. Time step size has to be significantly smaller than Kolmogorov time and particle response time. Wang *et al.* [9] suggested that $\mathrm{dt} = \min(0.2\tau_p, 0.2\tau_k)$ is appropriate for tracking monodisperse

system. For polydisperse suspension, this restriction implies using different time steps for different sets of particles which complicates collision detection process. Finally the time step size is limited by the stability condition (CFL number) of the flow solver. In our simulation background turbulent flow is computed with spectral accuracy by solving incompressible Navier-Stokes equations. In order to preserve numerical stability in the pseudo-spectral method the CFL number must be of order of 0.3. For testing purpose the turbulent flow has been computed on uniform grid with 128^3 points and time step size was set to dt=0.002. Average value of CFL number was around 0.21. More flow statistics including rms fluctuating velocity u', the longitudinal integral length scale L_f, the energy dissipation rate ϵ, the Taylor-microscale flow Reynolds number R_λ, spatial resolution parameter $k_{max}\eta$, Kolmogorov scales (length η and time τ_k), the skewness $\mathcal{S}$ and the flatness $\mathcal{F}$ are given in Table 2.

N	ν	u'	L_f	ϵ	R_λ	$k_{max}\eta$	CFL	η	τ_k	$\mathcal{S}$	$\mathcal{F}$
128	0.004	0.0127	1.542	0.192	101.6	1.505	0.21	0.024	0.145	-0.469	5.018

Table 2: Parameter setting and average flow statistics. The flow was used for testing accuarcy of the new integration scheme for computing particle trajectories.

In order to link the DNS scales, namely length, time and velocity to corresponding scales in realistic turbulence some value of energy dissipation rate has to be assumed. In our simulations energy dissipation rate was set to 400 $\mathrm{cm}^2/\mathrm{s}^3$. Now, the size of the particles in DNS can be explicitly defined. In this study we focus on collision-coalescence of cloud droplets with radii in the range (1-60 μm). Fllowing the criterion of Wang *et al.*[9] dt = min(0.2τ_p, 0.2τ_k) is straightforward to estimate appropriate dt for every particle size. Such simple evaluation leads us to conclude that only for particles larger than 12 μm we can use the same time step size both for the flow and for integration of particle trajectories. For smaller particles for example 5 μm time step size has to be more than four times shorter while for 1 μm droplets this time step must be hundred times shorter. This brings along undesirable consequences. Namely, running the code with hundred times smaller time step we will need to run the code hundred times longer to achieve the same physical time. Moreover for the small droplets ($\sim$ 1 μm), less collisional events occur due to smaller area of their cross-sections. Therefore, in order to obtain reliable statistics, the simulation has to be run at a longer time interval. To solve this complex problem we proposed new, numerically efficient algorithm for tracking low inertia particles. The new algorithm is based on computing of the particle velocity from background turbulent flow and its acceleration at the instant particle location. This treatment allows using larger time steps and preserve satisfactory accuracy.

Since the inertia of the droplets with radii 1-10 μm is very low, the droplets respond rapidly even if the background turbulent flow is changing dynamically. Nevertheless, the low but non-zero inertia does not allow the particles follow precisely the fluid elements. This in turn has significant impact on kinematic properties of the system i.e. both on

relative velocity between particles and theirs preferential concentration. In the new algorithm the effect of non zero inertia was included to equation of motion directly by subtracting from the fluid velocity term with fluid acceleration computed at the instant particle location. Such treatment allows to compute particle velocity directly and does not require additional integration approaches.

In this algorithm particle velocity is computed based on the asymptotic expansion of Maxey[3] as follows

$$\bar{V}(t+dt) \approx \bar{u}\left(\bar{x}=\bar{Y}(t+dt), t+dt\right) + \bar{W} - \tau_p \frac{D\bar{u}}{Dt}\bigg|_{\bar{x}=\bar{Y}(t+dt)} \tag{7}$$

The first term $\bar{u}$ is the fluid velocity evaluated at the particle location $\bar{Y}$. This term can be computed using 3D interpolation method from the regular grid. The second term $\bar{W}$ is the particle terminal velocity in stagnant air and is given analytically by Wang *et al*[9]. The third term includes contribution from the fluid acceleration where τ_p is particle response time. Local fluid acceleration in the 3rd term can be computed in the following way:

$$\frac{D\bar{u}}{Dt} = \frac{\partial \bar{u}}{\partial t} + \bar{u}\cdot\nabla\bar{u} = \frac{\partial \bar{u}}{\partial t} - \bar{\omega}\times\bar{u} + \nabla\left(\frac{\bar{u}^2}{2}\right) \tag{8}$$

where $\bar{\omega}$ is fluid vorticity. The first two terms are known explicitly in the spectral algorithm. The last term contains differential operator and is not explicitly known. The simplest way to compute the third term is to transfer $\bar{u}^2$ to Fourier space where differentiation reduces to arithmetic operation. Then, three additional FFT's have to be done in order to transfer back all three components of the acceleration term to the physical space.

3.1 Results

Accuracy of the new integration scheme has been examined through comparing values of kinematic properties of the bidisperse system computed using both new improved algorithm and the conventional direct interaction with a small time step[9]. In two separate runs we were tracking trajectory of 1M non aerodynamically-interacting particles with radii a_1= 5 μm and a_2=10 μm. Stokes numbers of the particles are 0.0158 and 0.063, respectively. For the first run the new efficient Lagranigian algorithm has been used, in which the particle velocities are evaluated by equation 7. Time step size was set the same for flow and particles ($\mathrm{dt}_{DNS} = 0.002$). The second simulation was performed using the direct integration[9] with $\mathrm{dt}_{DNS} = 0.00046$.

It is worthy to mention that both runs start from the same initial flow field and the same particle location. The flow is forced using a deterministic scheme[10] in which energy is applied in Fourier space only to two wavenumber shells (i.e., $0.5 < \mathrm{k} < 1.5$ and $1.5 <$

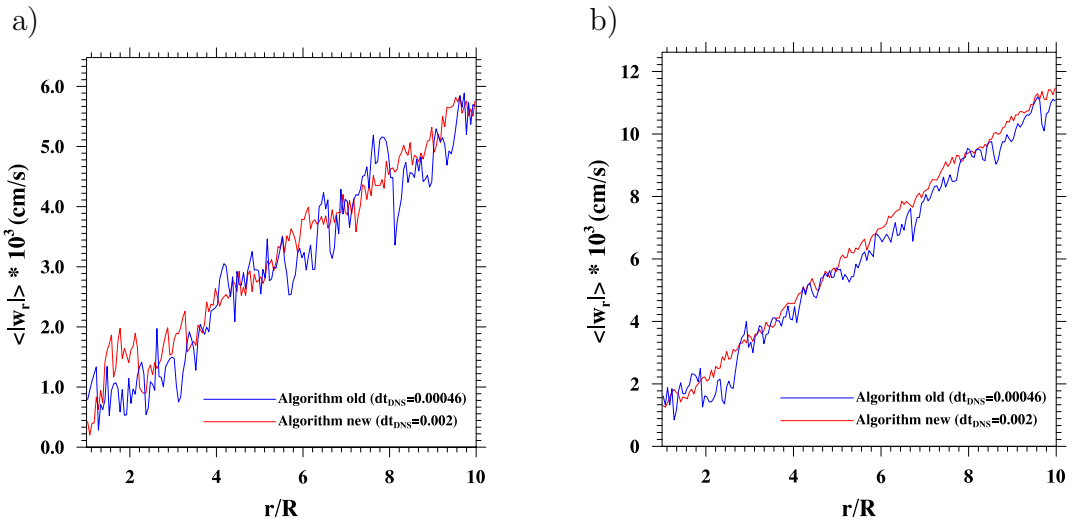

Figure 3: Relative velocity between *a*) $5\mu m$ and *b*) $10\mu m$ droplets in function of separation distance computed using both old and new optimized alghoritms

k < 2.5). Both runs took 10 eddy turnover times and data have been collected only after 3 eddy turnover times. The initial period has been skipped in order to exclude the effect of the particle relaxation from the initial random setting.

Radial distribution function and relative velocity in function of separation distance from both runs are presented in figures 3-5. The separation distance is normalized by the corresponding collision radius i.e. $R = 2a_1$ or $2a_2$ for monodisperse and $R = a_1 + a_2$ for bidisperse systems. Figure 3 shows relative velocity between particles with the same size. From the figure 3 we can conclude that both algorithms yield quantively similar results. Larger oscillations observed in the figure 3a can be attributed to small number of particles at this separation distance.

Figure 4 presents radial distribution function computed based on spherical formulation of Wang *et al.*[9]. Again both algorithms give comparable results, however at close separation the new algorithm underestimates the RDF. It is hard to asses if this is a characteristic feature of the new scheme or the difference result from small number of numerical data. It is necessary to run the code for longer time and obtain smaller statistical uncertainty.

Kinematic properties of the bidisperse system are presented in figure 5. The figure 5a shows relative velocity between particles with different sizes versus normalized separation distance. Radial distribution function g_{12} computed using formulae[11] is plotted in figure 5b.

Figure 5a shows that relative velocity computed using both schemes are in quantitative agreement. Again radial distribution function (figure 5b) seems to be a little underestimated in the new algorithm.

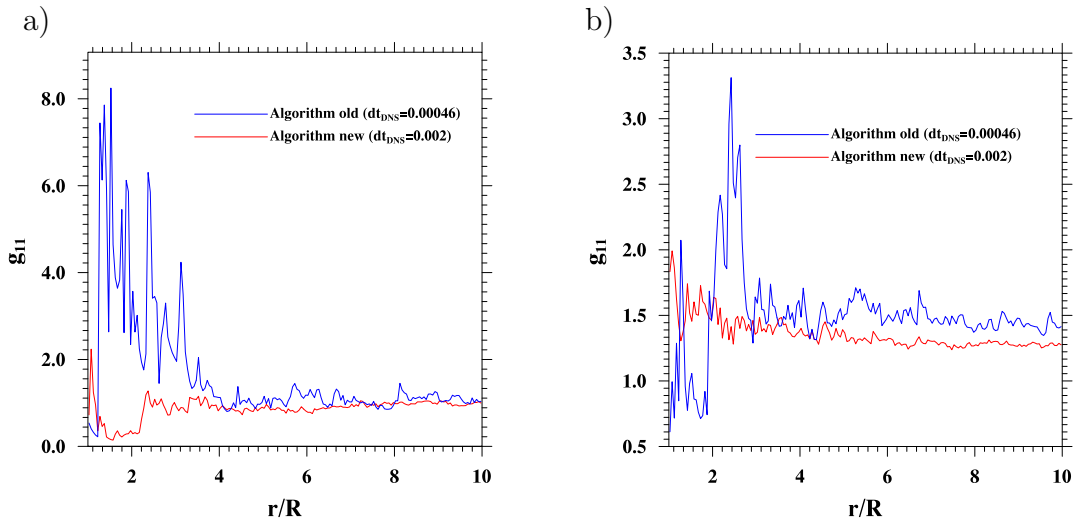

Figure 4: Radial distribution function of the monodisperse system with *a*) $5\mu m$ and *b*) $10\mu m$ droplets in function of separation distance. Blue line represents result computed using old algorithm. The red line shows results from the new improved algorithm

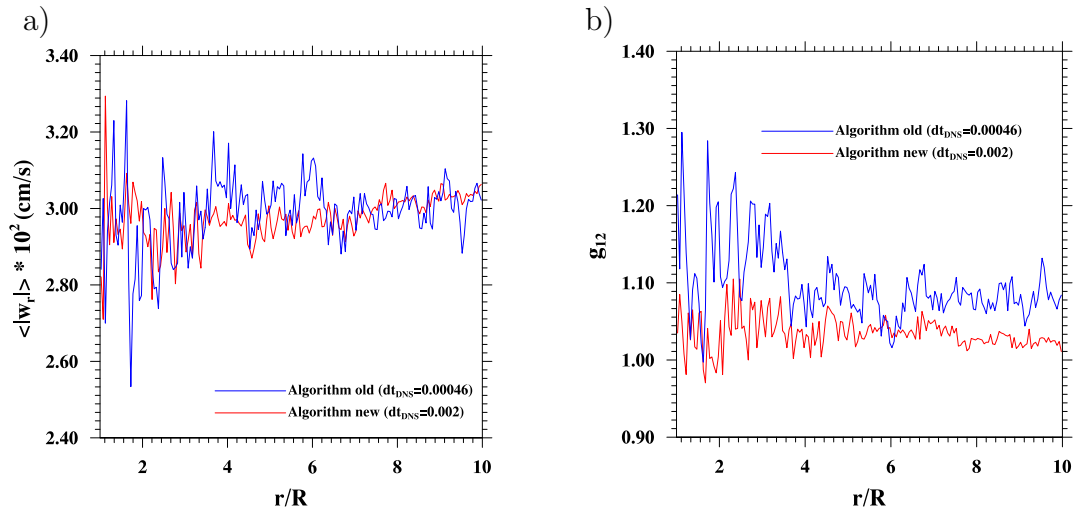

Figure 5: Relative velocity *a*) and radial distribution function *b*) of the bidisperse system ($a_1 = 5\mu m$, $a_2 = 10\mu m$

4 THREE DIMENSIONAL FFT IMPLEMENTATION

Direct numerical simulations of homogeneous isotropic turbulence using the standard pseudo-spectral method provide accurate kinematic and dynamic flow characteristics for inertial and dissipation subrange scales, as well as a basis for studying the interactions of inertial particles with turbulent flow structures. To achieve high flow Reynolds numbers, one has to perform high-resolution simulations demanding both intensive computation and large amount of memory. Parallel computing tools have to be developed, the most

important of which is the MPI implementation of 3D Fast Fourier Transform.

MPI implementations on parallel computers rely on domain decompositions in order to handle large grid resolutions. Here, we aim specifically at simultaneous domain decomposition in two of the three spatial dimensions to prepare us for near-future petascale computers with $\mathcal{O}$(10,000) processors. Previously developed 1D decomposition of Dmitruk *et al.*[12] does not permit the use of a large number of processors and 3D decomposition of (e.g., Eleftheriou *et al.*[13,14]; Fang *et al.*[15]) requires excessive communication. Preliminary attempts on 2D-decomposition FFTs using different communication/implementation strategies for array transpose operation have been made by Plimpton[16], Pekurovsky[17], and Takahashi[18]. For example, Plimpton's strategy was to pack and send as much data as possible in a multistage algorithm using MPI_Send and MPI_Irecv commands. Pekurovsky simply used the traditional MPI command MPIAlltoallv while Takahashi used MPIAlltoAll to communicate data.

Our first objective is to extend the 'round robin' approach used by Dmitruk *et al.*[12] for efficient communication within 1D decomposition, to 2D decomposition. Our second objective is a systematic study on scalability and efficiency using various 2D decompositions so their performances can be compared and potentially optimized. Our preliminary analysis showed that this approach was better than traditional MPI commands that are currently used for 2D decomposition (MPIAlltoallv or MPIAlltoall).

4.1 2D decomposition FFT strategy

A Three Dimensional FFT can be computed by taking a sequence of three One Dimensional FFTs along each direction of the three dimensional block data. On parallel computation, this data can be distributed among all processors using a two dimensional decomposition which implies slicing the computational domain in two directions, e.g. y_{matrix} and z_{matrix} (see figure 6a). While with this decomposition, the 1D FFT along the x direction of the data can be easily performed on each processor, the FFTs along y and z directions can not be done because data are distributed on different processors. Thus, special transposes have to be introduced.

In figure 6 is shown schematically the 2D Decomposition Strategy (we only show the Real to Complex 3D FFT, although the same strategy applies to Complex to Real transform). The data domain is decomposed into columns for each processor (figure 6a) slicing the data along the y_{matrix} and z_{matrix} directions. In this sample figure, there are a total of 16 processors, 4 on each y_{matrix} and z_{matrix} directions (it could be different numbers of processors along the two directions).

Then, the 2D Decomposition FFT strategy follows 5 subsequent steps. (1) One-dimensional real to complex FFT is performed along the x direction (figure 6b) in parallel on each processor. For this first FFT we need to add two additional yz plane of data at the end of the x direction because the output of a real-to-complex FFT provides $(N_x/2+1)N_yN_z$ complex numbers which represents $(N_x+2)N_yN_z$ elements ((N_x, N_y, and N_z are the number of data along each direction). Those elements have to be stored in

the matrix, half of them are the real components of the complex numbers and the other half are the imaginary components of the complex numbers.

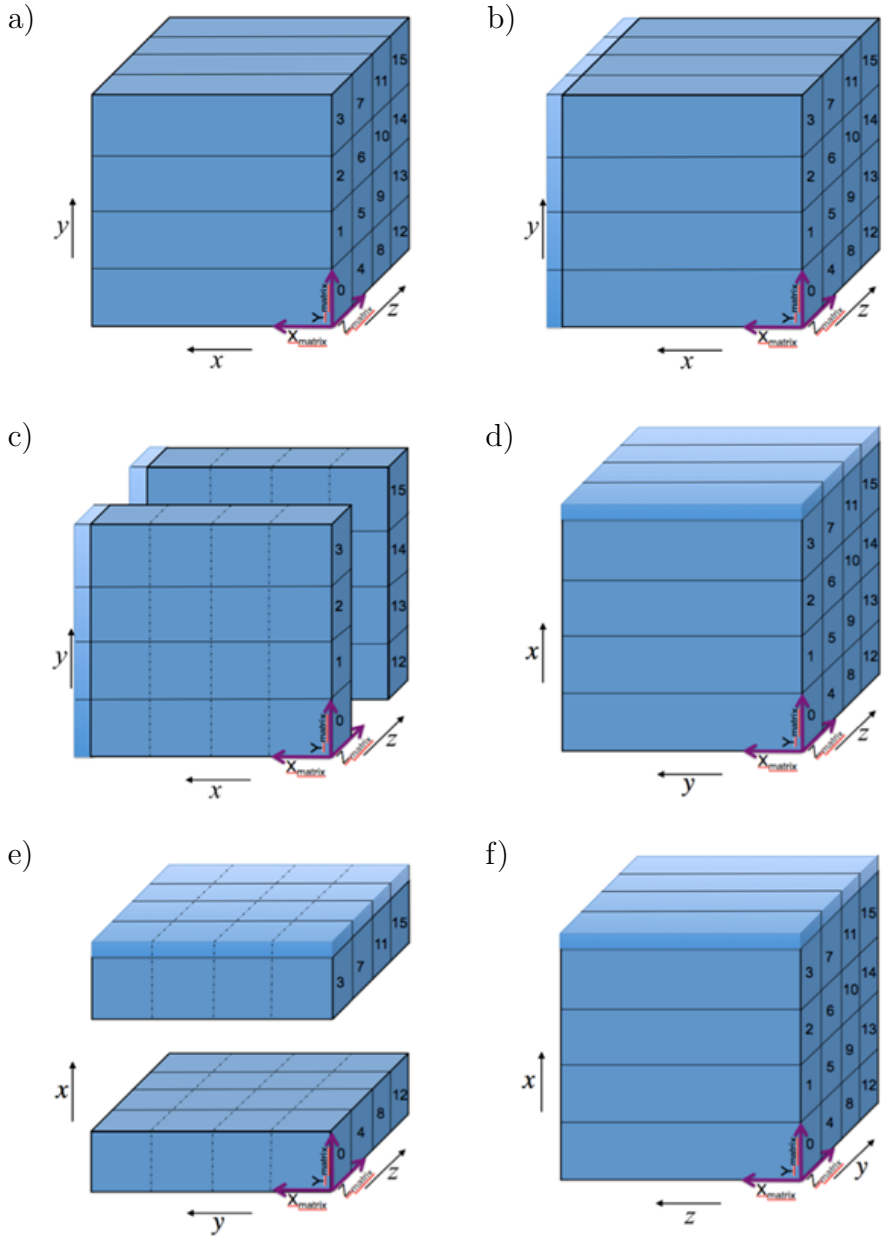

Figure 6: Three Dimensional FFT Real to Complex using the 2D Decomposition Strategy: a) Real Array, b) 1D FFT along x, c) Transpose between x and y directions, d) 1D FFT along y, e) Transpose between y and z directions, f) 1D FFT along z

(2) The data is transposed between x and y directions following the same strategy proposed by Dmitruk *et al.*[12]. The idea is to switch the data in a way that the elements

along the y direction are now stored along the x_{matrix} direction of all processors. It is important to point out that for this transpose, the communications among processors are only needed for those in the same slab of processors as shown in figure 6c. Each slab of processors can perform their communications at the same time. (3) A complex-to-complex one-dimensional FFT is carried along the y direction of the data, i.e., along the x_{matrix} direction (figure 6d). (4) The data is transposed now between the y and z directions (figure 6e) in a similar fashion as the transpose in step (2) to store the data along the z direction in the x_{matrix} direction. (5) Finally, the last one-dimensional complex-to-complex FFT is done along the z direction of the data which is stored along the x_{matrix} direction.

The strategy was implemented on 'Bluefire' - IBM Power 575 cluster (4064 POWER6 processors running at 4.7 GHz) at NCAR's supercomputing center. We also run the subroutine by Pekurovsky (2009) using 1D and 2D decomposition, as well as the 1D decomposition strategy by Dmitruk *et al.*[12]. Figure 7 shows the results of the speed up from the timing tests for a 512^3 problem. For comparison, we also include in the figure the results obtained by Takahashi[18] for a 256^3 problem in a different machine (Tsukuba system). We used it as reference as we expect better timing results as the problem size increase. However, we can notice that the 1D decomposition strategy starts to worsen for 32 processors. Furthermore, the 2D decomposition by Pekurovski also start to worsen around same number of processors. On the other hand, we can see a much better scalability in the new implementation, better than all the other approaches.

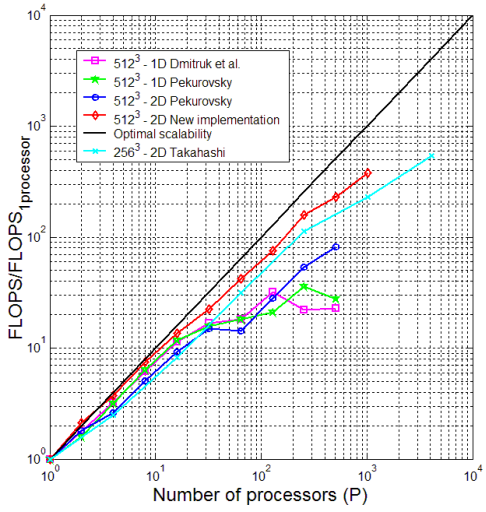

Figure 7: Comparison of the scalability results from our 2D decomposition strategy and others for a 512^3 real data

4.2 Complexity analysis

We developed an approximate quantitative model for execution time of the message passing parallel 3D FFT on a square grid of size N^3 using P processors. This complex-

ity analysis helps to study theoretically the execution time of the different strategies. Dmitruk *et al.*[12] showed that the total time taken for a machine to perform a Three Dimensional FFT is the sum of the computational time and the communication time. For 1D decomposition strategy proposed by them, the computational time is given by

$$T_{COMP} = \frac{5}{2}\frac{N^3 \log_2(N^3)}{P} t_c + \left[2\left(\frac{(N+2)N^2}{P}\right)\frac{1}{P} + (P-1)\left(\frac{(N+2)N^2}{P}\right)\frac{1}{P}\right] t_a \quad (9)$$

and the communication time is

$$T_{COMM} = 2(P-1)\left(\frac{(N+2)N^2}{P}\right)\frac{1}{P} t_w + 2(P-1)t_s \quad (10)$$

where N is the data size along any of the directions, P is the total number of processors, t_c is the computation time per floating point operation, t_a is the memory-to-memory copy time per word, t_w is the time for transmission of a single word between processors, and t_s is the startup or latency time.

Our new implementation is an extension of Dmitruk's[12] strategy. Based on the same analysis done by Dmitruk *et al.*[12], we obtained the computational time and communication time to perform a Three Dimensional FFT:

$$T_{COMP} = \frac{5}{2}\frac{N^3 \log_2(N^3)}{P_y P_z} t_c + 2\left[2\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y} + (P_y-1)\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y}\right] t_a \quad (11)$$

$$T_{COMM} = 2\left[2(P_y-1)\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y} t_w + 2(P_y-1)t_s\right] \quad (12)$$

where the total number of processors P is $P_y * P_z$. P_y is the number of processors along the y_{matrix} direction and P_z is the number of processors along the z_{matrix} direction. It is important to point out that equation 12 is similar to the one presented by Takahashi (2009).

To discuss the performance of the FFT algorithms, the elemental times involved in the estimates given by equations 9 through 12 (t_c, t_a, t_w, and t_s) are needed. Those values can be obtained empirically through some simple test runs on the machine (Bluefire - IBM cluster in this case). t_c can be obtained by doing repeatedly some floating point operations. t_a is determined by running standard loops of memory-to-memory copy on a single processor. The communication times can be obtained by timing while doing several simple communications (send and receive commands) of different array sizes between two processors only.

The floating-point operation chosen for the experiment was 'multiplication', and the average time for this was 1.5 ns/FLOP. Table 3 shows all other results from the performed

tests on Bluefire. The memory-to-memory copy time varies with the array size to copy. However, the copy time stabilizes for large array sizes, to an average time of 1.3 ns/word. As for the communication times, Table 3 shows the wall clock time it takes to send and received a message of different sizes. The average slope of this time corresponds to a time per word communicated (t_w) and its value is $t_w \approx 0.6$ ns/word. The latency time is the time to communicate a message of zero size which can be obtained by extrapolating the data shown to a zero size message ($t_s \approx 6\mu$s). Both communication timings are close to what is reported by NCAR Supercomputer Center under optimal conditions ($t_w \approx 0.2$ ns/word and $t_s \approx 1.3\mu$s). It is important to mention that the Bluefire is a machine with 128 nodes each with 32 processors (machine architecture). Thus, the communication time between processors in different nodes may be different. The results on Table 3 correspond to two processors communicating within the same node. We also perform experiments with two communicating processors at different nodes. The results are shown in Table 4, it can be noticed that the communication time increases when the communicating processors are not in the same node due to the Bluefire's machine architecture.

Array Size	t_a (ns/word)	communicating time (μs)
2^3	10.490	6.008
4^3	3.472	6.1035
8^3	2.132	7.391
16^3	1.485	14.9
32^3	1.208	91.98
64^3	1.079	198.0
128^3	1.443	1288.0

Table 3: Timings from numerical experiments performed on Bluefire to obtain t_a, t_w, and t_s

Array size	ONE NODE Processors 00-01 (μs)	TWO NODES Processors 00-63 (μs)	FOUR NODES Processors 00-127 (μs)
2^3	6.008	7.391	13.78
4^3	6.1035	7.987	14.114
8^3	7.391	9.0122	19.193
16^3	14.9	18.883	42.22
32^3	91.98	103.62	166.79
64^3	198.0	737.8	967.59
128^3	1288.0	6136	7533.41

Table 4: Communication time between two processors sitting at different nodes on Bluefire for different message sizes

The theoretical equations can be compared with the real time results obtained previously. Figure 8 shows this comparison. For the theoretical results in this plots, the

communication times (t_w, and t_s) are from Table 3 (i.e. two communicating processors in the same node). The theoretical equation given by Dmitruk *et al.*[12] slightly over predicts the real time results. However, the theoretical timing equations for the 2D decomposition predicts reasonably well up to 32 processors. For larger number of processors, the equations under predict the real times. We argue that this is mainly due to the fact that for a number of processors larger than 32, the communication times are larger (see Table 4). Note again that our new 2D decomposition strategy is faster than the one from Pekurovsky[17]. Pekurovsky communicational strategy is based on the MPI_Alltoall command. Dmitruk *et al.*[12] has proved that a combination of MPI_send and MPI_receiv commands for communicational strategy is faster than using simply the MPI_Alltoall command. It also seems that the MPI_Alltoall command is not a suitable command to be used in machine architecture such as Bluefire's. From this figure we can also note that the speed up's slope of the 1D decomposition decreases when the number of processors used is near to the domain size along the decomposed direction.

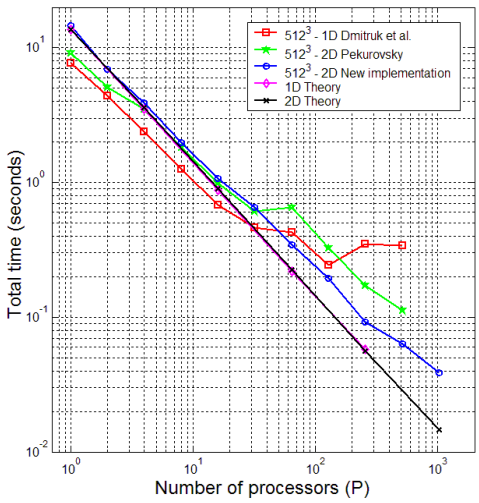

Figure 8: Comparison of the complexity analysis equations with real time data

Plimpton[16] presents another 2D decomposition strategy. Following that strategy, the total time to perform a FFT is given by:

$$T_{COMP} = \frac{5}{2}\frac{N^3 \log_2(N^3)}{P_y P_z} t_c + 2\left[2\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y} + \log_2(P_y)\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y}\right] t_a \quad (13)$$

$$T_{COMM} = 2\left[2\log_2(P_y)\left(\frac{(N+2)N^2}{P_y P_z}\right)\frac{1}{P_y} t_w + 2\log_2(P_y) t_s\right]. \quad (14)$$

The analysis presented can be used to predict the timing behavior for any number of processors. In figure 9, we compare the theoretical total time using the equations just

presented as the problem was solved in Bluefire. It can be noticed that all strategies theoretically behave similarly. However, for larger than 4096 processors, Plimpton's strategy scales better. In order to explain this, in figure 10 the communication time and the computational time is shown separately.

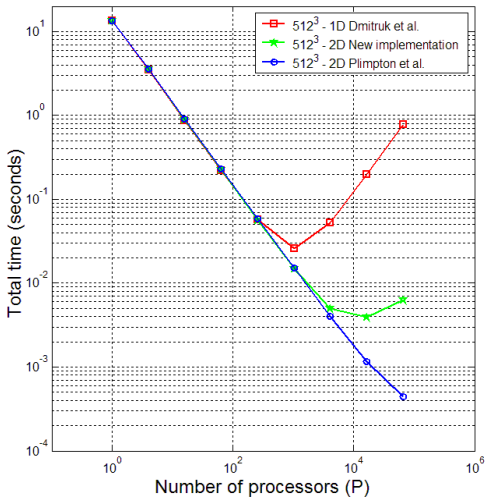

Figure 9: Comparison of the Total Time among the different complexity analysis equations for a 512^3 problem

In that figure (10), we can see that the most time consuming part for a 512^3 problem in Bluefire is the computation time and it is similar to any of the strategies as expected. The reason for this is that Bluefire has very good communication times. It takes 0.6 ns to send and receive a word but 1.5 ns for an operation and 1.3 ns to copy a word. Also, this computation time scales linearly while the communication time is not linear at all. In the later, we note that our 2D decomposition strategy theoretically behaves badly for large number of processors, while Plimpton's strategy behaves well. The reason for this is that for larger number of processors the time to communicate a message in our strategy is $(P_y - 1)$ while for Plimpton's is $\log_2(P_y)$, the first is considerably larger for large number of processors. This triggers a problem with the latency time in our approach, which is negligible for fewer processors.

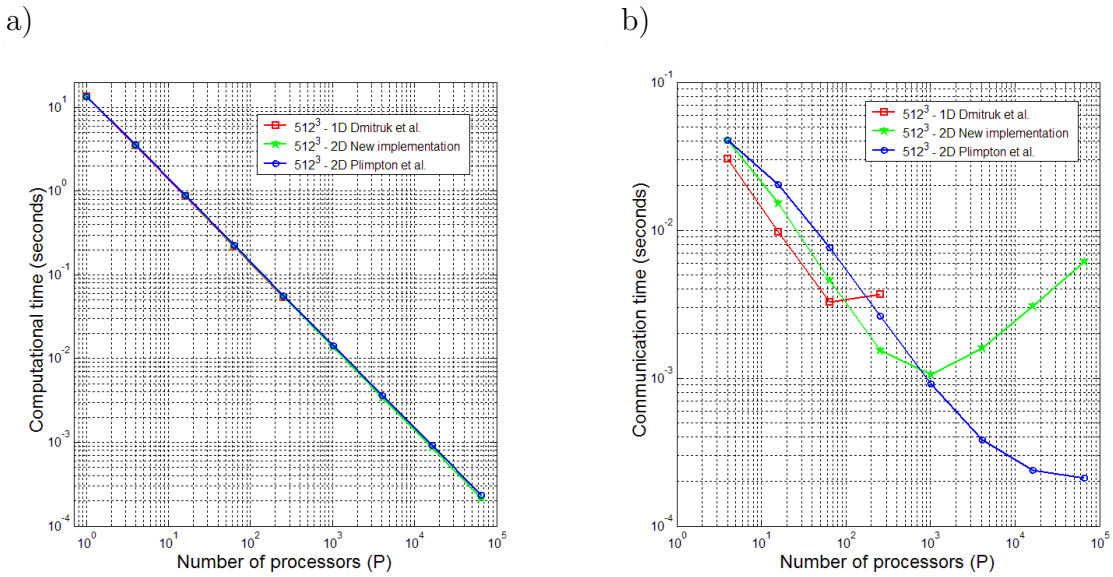

Figure 10: Comparison of the computational time and communication time among the different complexity analysis equations: a) Communication time, b) Computation time

5 CONCLUSIONS

In this paper, we have discussed several implementation issues for the HDNS code. First, a parallel HDNS code has been developed using MPI. The results pertinent to turbulent collisions of cloud droplets we obtained from our newly developed MPI HDNS code are in excellent agreement, with previously published results based on the OpenMP implementation. It shows that the MPI code can now be used to simulate a larger problem size and larger flow Reynolds number.

Second, an alternative approach based on the asymptotic expansion of Maxey[3] has been implemented to treat the motion of droplets with small Stokes numbers. This optimized Lagrangian integration scheme could use a significantly larger time step, yielding a better computational efficiency without much effect on the accuracy of the simulated collision statistics. However, the new approach requires 4 additional FFTs and some extra memory usage. This prepares us for treating a polydisperse size distribution of droplets in our HDNS approach.

Finally, a new MPI implementation of FFT based on 2D domain decomposition has been developed. It shows a promising scalability performance. For tests done at 512^3 grid and up to 1024 processors, our implementation scales better than other known implementations (also utilizing 2D decomposition), namely Plimpton's[16], Pekurovsky's[17] and Takahashi's[18]. However, for larger number of processors, Plimpton's strategy scales better due to more massive data communication.

The above advances are being used to perform HDNS for large domain size and for a polydisperse droplet size distribution. New results on collision rate and kinematics pair

statistics of cloud droplets will be reported at the conference.

Acknowledgements

This work was supported by the U.S. National Science Foundation (NSF) under grants ATM-0527140 and NSF ATM-0730766. Computing resources are provided by National Center for Atmospheric Research (CISL-35751010 and CISL-35751014). We are also grateful to Professor Federico Toschi and Dr. Daisuke Takahashi for the helpful advises regarding parallel FFT implementation.

References

[1] O. Ayala, W.W. Grabowski and L.-P. Wang, A hybrid approach for simulating turbulent collisions of hydrodynamically-interacting particles, *J. Comp. Phys.*, **225**, 51–73 (2007).

[2] L.P. Wang, B. Rosa, H. Gao, H. Guowei and G. Jin, Turbulent collision of inertial particles: Point-particle based, hybrid simulations and beyond, *Int. Journal of Multiphase Flow*, **35**, 854–867 (2009).

[3] M.R. Maxey, The gravitational settling of aerosol-particles in homogeneous turbulence and random flow fields, *J. Fluid Mech.*, **174**, 441–465 (1987).

[4] C.N. Franklin, P.A. Vaillancourt, M.K. Yau, Statistics and parameterizations of the effects of turbulence on the geometric collision kernel of cloud droplets, *J. Atmos. Sci.*, **64**, 938–954, (2007).

[5] O. Ayala, B. Rosa, L.-P. Wang and W.W. Grabowski, Effects of Turbulence on the Geometric Collision Rate of Sedimenting Droplets: Part 1. Results from direct numerical simulation *New J. Physics*, **10**, 075015, 2008

[6] L.P. Wang, O. Ayala, B. Rosa and W.W. Grabowski, Turbulent collision efficiency of heavy particle relevant to cloud droplets, *New Journal of Physics*, **10**, 075015,(2008).

[7] M.P. Allen and D.J. Tildesle, Computer Simulation of Liquids, Oxford University Press, (1989).

[8] L.-P. Wang, O. Ayala, S.E. Kasprzak, and W.W. Grabowski, Theoretical formulation of collision rate and collision efficiency of hydrodynamically-interacting cloud droplets in turbulent atmosphere, *J. Atmos. Sci.*, **62**, 2433–2450, (2005)

[9] L.-P. Wang, A. Wexler, Y. Zhou, Statistical mechanical description and modelling of turbulent collision of inertial particles, *J. Fluid Mech.*, **415**, 117–153 (2000).

[10] L.-P. Wang and B. Rosa, A spurious evolution of turbulence originated from round-off error in pseudo-spectral simulation, *Computers & Fluids*,**38**, 1943–1949 (2009).

[11] Y. Zhou, A. Wexler and L.-P. Wang, Modelling turbulent collision of bidisperse inertial particles, *J. Fluid Mech.*, **433**, 77–104 (2001).

[12] P. Dmitruk, L-P. Wang, W.H. Matthaeus, R. Zhang, D. Seckel, Scalable parallel FFT for spectral simulations on a Beowulf cluster, *Parallel Computing*, **27**, 1921–1936, (2001).

[13] M. Eleftheriou, J.E. Moreira, B.G. Fitch, and R.S. Germain, A volumetric FFT for BlueGene/L, *High Performance Computing - HIPC*, **2913**, 194–203, (2003).

[14] M. Eleftheriou, B.G. Fitch, A. Rayshubskiy, J.C. Ward and R.S. Germain, Scalable Framework fo 3D FFTs on the Bluegen/l supercomputer: Implementation and early performance measurements, *IBM Journal of Research and Development*, **49** 457 (2005).

[15] B. Fang, Y. Deng, G. Martyna, Performance of the 3D FFT on the 6D network torus QCDOC parallel supercomputer, *Computer Physics Communications*, **176**, 531–538, (2007).

[16] J.S. Nelson, S.J. Plimpton and M.P. Sears, Plane-wave electronic-structure calculations on a parallel supercomputer, *Physical Review B*, **47**, 1765–1774, (1993).

[17] D. Pekurovsky, Scaling Three-dimensional Fourier Transform, San Diego Super Computer Summer Institute, http://www.sdsc.edu/us/resources/p3dfft/, San Diego, USA, (2007).

[18] D. Takahashi, An Implementation of Parallel 3-D FFT with 2-D Decomposition on a Massively Parallel Cluster of Multi-Core Processors, In proceedings of the *Eighth International Conference on Parallel Processing and Applied Mathematics*, (PPAM), Wroclaw, Poland, (2009).