Journal of Computational Physics 245 (2013) 235–258

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Analysis and parallel implementation of a forced N-body problem

C.E. Torres [c,d], H. Parishani [b], O. Ayala [e], L.F. Rossi [a,*], L.-P. Wang [b]

[a] Department of Mathematical Sciences, University of Delaware, Newark, DE 19716-3140, USA
[b] Department of Mechanical Engineering, University of Delaware, Newark, DE 19716-3140, USA
[c] Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, USA
[d] Departamento de Informática, Universidad Técnica Federico Santa María, Valparaíso, Chile
[e] Department of Engineering Technology, Old Dominion University, Norfolk, VA 23529, USA

ARTICLE INFO

Article history:
Received 23 April 2012
Received in revised form 28 February 2013
Accepted 2 March 2013
Available online 19 March 2013

Keywords:
GMRes
Preconditioner
Domain decomposition
Hydrodynamic interaction
Cloud droplets
Direct numerical simulation
MPI implementation
Contour integral

ABSTRACT

The understanding of particle dynamics in N-body problems is of importance to many applications in astrophysics, molecular dynamics and cloud/plasma physics where the theoretical representation results in a coupled system of equations for a large number of entities. This paper concerns algorithms for solving a specific N-body problem, namely, a system of disturbance velocities for hydrodynamically interacting particles in a particle-laden turbulent flow. The system is derived from the improved superposition method of [1]. Targeting for scalable computations on petascale computers, we have carried out a thorough study of a parallel implementation of GMRes with different features, such as preconditioners, matrix-free and parallel sparse representation of the matrix through 1D and 2D spatial domain decompositions. Gauss–Seidel method is also studied as a reference iterative algorithm. The range of conditions for efficiency and failure of each method is discussed in detail.

Through perturbation analysis, we have conducted a series of experiments to understand the effect of particle sizes, interaction symmetry, inter-particle distances and interaction truncation on the eigenvalues and normality of the linear system. For situations where the system is ill-conditioned, we introduce a restricted Schwarz type preconditioner. We verified the parallel efficiency of the preconditioner using 1D domain decomposition on a parallel machine. A benchmark problem of particle laden turbulence at 512^3 resolution with 2×10^6 particles is studied to understand the scalability of the proposed methods on parallel machines. We have developed a stable and highly scalable parallel solver with an affordable computational cost even for ill-conditioned systems through preconditioning. On 64 cores, using GMRes in 2D domain decomposition, we achieved a speed-up of ~ 5.6x (relative to 1D domain decomposition on the same number of processors). Our complexity analysis showed that for large N-body problems, the proposed GMRes scheme scales well for moderate to large number of processors in current tera to petascale computers.

1. Introduction

The simulation of the evolution of a system of N interacting bodies is known as the classical N-body problem. A force (or kernel) is exerted on each body, which arises due to its interaction with all the other bodies in the system (e.g. based on Newton's law of gravitation or Coulomb's law of electrostatic interaction). Solving the equations of motion analytically

* Corresponding author. Tel.: +1 302 831 1880.
E-mail address: rossi@math.udel.edu (L.F. Rossi).

0021-9991/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.03.008

for a large number of particles is not possible, thus numerical methods must be introduced. At each discrete time interval, the force on each particle is computed and this information is used to update the position and velocity of each particle.

There are numerous applications of this problem in areas such as astrophysics, molecular dynamics, solid-state physics, and plasma physics. Apart from these, some problems in numerical complex analysis and elliptic partial differential equations can also be solved using this approach. Applications of the problem are also found in the radiosity method, which attempts to create images by computing the equilibrium distribution of light for complex scene geometries [58].

Recently, in the context of cloud physics, [1] solved an N-body problem for the motion of water droplets in a turbulent cloud. They developed a hybrid direct numerical simulation (HDNS) based on an improved superposition method to study droplet–droplet hydrodynamic interactions. The improved formulation relates the drag force to a representation of Stokes disturbance flow around each droplet [2]. The carrier-fluid turbulence affects the motion of the droplets through the interfacial hydrodynamic forces (mainly the viscous drag); the motion of each droplet can be affected by the presence of other droplets in the system, either through the strong local near-field binary hydrodynamic interaction or by the cumulative many-body, long-range interactions [3]; and the background air turbulence can also affect the hydrodynamic interactions as the turbulence defines both the far-field conditions and the local environment for hydrodynamic interactions.

As a result of the coupled hydrodynamic interactions among the droplets, a large linear system has to be solved to obtain the disturbance flow seen by each particle. This system is of dimension $3N_p$, where N_p is the number of droplets in the simulation. The disturbance flow around each droplet decays with distance; the hydrodynamic interactions were truncated at cut-off radius. This leads to a sparse linear system.

Initially [1] implemented an OpenMP parallelization of the HDNS method. In their implementation, hydrodynamic interaction took approximately 80% of the total runtime for a problem at 128^3 grid resolution with approximately 500 K droplets using a Gauss–Seidel linear solver on OpenMP threads. Thus, hydrodynamic interactions are the bottleneck in the computation for larger problem sizes. In this paper we discuss our efforts toward an efficient parallel implementation of droplet–droplet hydrodynamic interactions for large problem sizes when the domain is decomposed into many subdomains for parallel computation.

In general, an efficient parallel implementation of a linear solver requires that we understand the properties of the linear system such as sparsity pattern, physical constraints, distribution of eigenvalues, normality of the matrix [4,5]. We will show that GMRes [6,7,4,8] is able to approximate the solution very efficiently, making it the right tool for the problem. However, we also discuss some extreme conditions when GMRes might not behave well. To overcome the ill-conditioning we proposed a preconditioner. An ill-conditioned linear system has a large condition number $\kappa = \max_i|\lambda_i|/\min_j|\lambda_j|$, where λ_i represent the eigenvalues of the matrix of hydrodynamic interactions. For comparison purposes, we implemented a Block Gauss–Seidel linear solver in parallel as an extension of the implementation of [1]. When comparing different computational approaches, we considered Conjugate Gradient (CG) or Bi-Conjugate Gradient (BiCG) methods [7,4] but found that GMRes was the best option in this case. Our system is not necessarily positive definite so CG would not be suitable, and BiCG requires an additional matrix–vector multiplication and a conjugation.

Our linear solvers have been integrated into a domain decomposition approach to enable parallel computation. The domain decomposition has been chosen because the particles are randomly distributed in the domain and the kernel for the interactions has compact support. The parallelization of the block Gauss–Seidel is reduced to the local iterations and global sharing of data. Notice however that due to the truncation of interaction distance of the kernel [1] the data sharing is only needed with the neighboring subdomains. Parallel implementations of GMRes have been studied by several researchers [9–15]. While most of them focused on parallelizing the whole algorithm, in this work, since the most time consuming portion of the GMRes algorithm is matrix–vector multiplication, we focus our efforts in parallelizing it efficiently in the context of a domain decomposition strategy. In addition, some GMRes parallel implementations rely on commercial libraries, which impacts existing code portability.

Our results and findings are valuable to any N-body problem where the kernels acting on all the bodies are coupled and the kernel is approximated by a cut-off radius of influence. Examples of such systems include calculation of extinction and Raman intensities for small metal particles [16]; electrodynamics of noble metal nanoparticles [17]; particle interactions in electrophoresis [18,19]; interactions for magnetic particles [20,21]; to name a few.

This particular N-body problem in a turbulent background flow is connected to other problems with interactions that are screened by physical effects. The N-body problem for the motion of water droplets in a turbulent cloud discussed in this paper involves long-range Stokes flow interactions among the droplets since they decay as $(r/a)^{-1}$ (a is the particle radius). It is well known that the variance of particle velocity (in an otherwise stagnant flow) diverges with the size of the system [22] due to the long-range effect. [23] has shown that to keep the variance finite a form of hydrodynamic screening is necessary, similar to the Debye-like screening of electrical potential associated with a fixed ion in an ionic solution. This was later demonstrated by [24] when studying the transient sedimentation of a dilute suspension. In our problem, we are dealing with a suspension in a turbulent flow and the particles are smaller than the Kolmogorov length scale. The turbulence also brings a screening effect because the relative motion of droplets due to turbulence damps the long-range Stokes aerodynamic interaction. Furthermore, it has been shown that the effects of particle inertia can reduce the long-range hydrodynamic interactions [25]. [1], when considering the collision efficiency among droplets in a turbulent flow, showed that the cut-off radius of influence was smaller than in the case of a stagnant flow. This implies that the long-range interactions were swapped by turbulence and droplet inertia effects. In comparison with other N-body problems, the truncation of the kernel considered

here could make our problem somehow different. However, by considering turbulence or some other forcing agents that may create a screening effect, the problems become similar.

The paper is organized as follows. Section 2 explains the hybrid DNS method. In Section 3 we study the properties of the governing linear system of equations. The domain decomposition strategy to parallelize the HDNS method is discussed in Section 4. Section 5 includes a perturbation analysis of the matrix and convergence analysis for GMRes method to study the appropriateness of the iterative method to the N-body problem in hand. In Section 6, we present the timing and scalability of the proposed methods, followed by a complexity analysis in Section 7 to study the subroutine performance for very large numbers of processors. Section 8 contains a summary of our work and presents our conclusions. For completeness, we include discussion of other useful approaches for this problem and the methods in appendices.

2. Hybrid DNS

The basic ideas and algorithms for the HDNS approach have been presented in [1,26]. Only a brief description of the approach is given here to prepare for our discussion of the aerodynamic interaction of droplets. We consider a dilute suspension of droplets in a background turbulent air flow $\mathbf{U}(\mathbf{x},t)$ solved by 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 kth droplet of radius $a^{(k)}$, denoted by $\mathbf{U}(\mathbf{Y}^{(k)}(t),t)$, is 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 kth droplet will be denoted by $\mathbf{V}^{(k)}(t)$.

As a first approximation, we assume the disturbance flows resulted from the presence of the droplets in the background turbulent flow are localized in space, on the assumption that the droplet size (10–60 μm in radius) is much smaller than the flow Kolmogorov scale (∼1 mm). If Stokes disturbance velocities of all N_p droplets in the system are superposed appropriately [27], the disturbance velocity $\mathbf{u}^{(k)}$ felt by a kth droplet is:

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

where $\mathbf{u}_S$ is the Stokes disturbance flow defined as:

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

Note that (1) is derived from the requirement that the composite flow field (background flow plus the disturbance flow fields caused by droplets) should satisfy, on average, the no-slip boundary condition on the surface of each droplet [27].

Given $a^{(k)}$, $\mathbf{V}^{(k)}$, $\mathbf{Y}^{(k)}$ and $\mathbf{U}(\mathbf{Y}^{(k)},t)$ at a specified time t, (1) is a system of equations of $3N_p$ unknowns for the three components of the disturbance velocities ($\mathbf{u}$) of the N_p droplets. The disturbance velocity at the location of each droplet is coupled with the disturbance velocities of all other droplets and the three spatial components of the disturbance velocity cannot be separated to form three independent smaller linear systems. Therefore, system (1) must be solved as a whole to yield the disturbance velocities $\mathbf{u}^{(k)}$.

The most computationally demanding step in HDNS approach is to solve (1) for $\mathbf{u}^{(k)}$. Since for every droplet, the summation on the right hand side (RHS) of (1) should be carried out over all other droplets in the domain, the computational cost of the summation in (1) grows as $O(N_p^2)$ after computing the coefficients.

As far as collision statistics are concerned, [1] showed that one could truncate the summation and restrict the radius of influence of the disturbance flow caused by a kth droplet to a distance $H_{trunc}\times a^{(k)}$. Their experiments showed that the computed collision efficiency is insensitive to H_{trunc} if $H_{trunc} > 35$. In this paper, we will use dimensionless truncation radius of $H_{trunc} = 50$ which is more conservative than $H_{trunc} = 35$. This will significantly reduce the number of neighboring droplets in the governing equations as we will see in §3. Inside the truncation sphere, in order to efficiently locate the droplets and their immediate neighbors, we make use of the cell-index method and linked lists [28]. We note that in a periodic domain without truncation, the Stokes' kernel (2) may diverge since it only decays as $1/r$.

Once the perturbation velocities $\mathbf{u}^{(k)}$ are solved (see Appendix A) and undisturbed fluid velocity $\mathbf{U}(\mathbf{Y}^{(k)}(t),t)$ are computed, droplets are advanced by solving their equation of motion which for the kth droplet becomes:

$$\frac{d\mathbf{V}^{(k)}(t)}{dt} = -\frac{\mathbf{V}^{(k)}(t) - \left(\mathbf{U}(\mathbf{Y}^{(k)}(t),t) + \mathbf{u}^{(k)}\right)}{\tau_p^{(k)}} + \mathbf{g} \tag{3}$$

$$\frac{d\mathbf{Y}^{(\mathbf{k})}(\mathbf{t})}{dt} = \mathbf{V}^{(k)}(t) \tag{4}$$

where $\tau_p^{(k)} = \frac{2\rho_p(a^{(k)})^2}{9\mu}$ is the Stokes inertial response time of the k-th droplet and $\mathbf{g}$ is the gravitational acceleration.

3. Governing system of equations

In this section, and in the following ones, we study properties of the linear system of Eqs. (5) to later take advantage of them. We first introduce a set of notations for the linear system of equations in order to develop a complete description (given in Appendix A) of the proposed solution methods. First, let us define (1) in a more familiar form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{3N_p \times 3N_p}$, $\mathbf{x}$ is the vector of the components of the perturbation velocities $\mathbf{u}^{(k)} = \langle u_k, v_k, w_k \rangle$ for $k = 1, \ldots, N_p$ arranged for simplicity as follows $\mathbf{x} = \langle u_1, v_1, w_1, \ldots, u_{N_p}, v_{N_p}, w_{N_p} \rangle^T$ and $\mathbf{b}$ is the right hand side vector of dimension $3N_p$.

To express the linear system as a matrix $\mathbf{A}$ we first rewrite (1) as follows:

$$\mathbf{u}^{(k)} + \underbrace{\sum_{m=1}^{N_p}}_{m \neq k} \mathbf{u}_S\left(\mathbf{Y}^{(k)}(t) - \mathbf{Y}^{(m)}(t); a^{(m)}, \mathbf{u}^{(m)}\right) = \underbrace{\sum_{m=1}^{N_p}}_{m \neq k} \mathbf{u}_S\left(\mathbf{Y}^{(k)}(t) - \mathbf{Y}^{(m)}(t); a^{(m)}, \mathbf{V}^{(m)} - \mathbf{U}(\mathbf{Y}^{(m)}, t)\right), \quad k = 1, 2, \ldots, N_p \tag{5}$$

Now, we rewrite $\mathbf{u}_S\left(\mathbf{Y}^{(k)}(t) - \mathbf{Y}^{(m)}(t); a^{(m)}, \mathbf{u}^{(m)}\right)$ in a more explicit form to obtain the linear dependence on $\mathbf{u}^{(m)} = \langle u_m, v_m, w_m \rangle$. Omitting time dependence for clarity, we obtain:

$$\begin{aligned}
\mathbf{u}_S\left(\mathbf{r}^{(k,m)}; a^{(m)}, \mathbf{u}^{(m)}\right) = \Bigg\langle & \underbrace{\left(\Lambda^{(k,m)}\left(r_1^{(k,m)}\right)^2 + \beta^{(k,m)}\right)}_{\alpha_{1,1}^{(k,m)}} u_m + \underbrace{\left(\Lambda^{(k,m)} r_1^{(k,m)} r_2^{(k,m)}\right)}_{\alpha_{1,2}^{(k,m)}} v_m + \underbrace{\left(\Lambda^{(k,m)} r_1^{(k,m)} r_3^{(k,m)}\right)}_{\alpha_{1,3}^{(k,m)}} w_m, \underbrace{\left(\Lambda^{(k,m)} r_1^{(k,m)} r_2^{(k,m)}\right)}_{\alpha_{2,1}^{(k,m)}} u_m \\
& + \underbrace{\left(\Lambda^{(k,m)}\left(r_2^{(k,m)}\right)^2 + \beta^{(k,m)}\right)}_{\alpha_{2,2}^{(k,m)}} v_m + \underbrace{\left(\Lambda^{(k,m)} r_2^{(k,m)} r_3^{(k,m)}\right)}_{\alpha_{2,3}^{(k,m)}} w_m, \underbrace{\left(\Lambda^{(k,m)} r_1^{(k,m)} r_3^{(k,m)}\right)}_{\alpha_{3,1}^{(k,m)}} u_m + \underbrace{\left(\Lambda^{(k,m)} r_2^{(k,m)} r_3^{(k,m)}\right)}_{\alpha_{3,2}^{(k,m)}} v_m \\
& + \underbrace{\left(\Lambda^{(k,m)}\left(r_3^{(k,m)}\right)^2 + \beta^{(k,m)}\right)}_{\alpha_{3,3}^{(k,m)}} w_m \Bigg\rangle
\end{aligned} \tag{6}$$

where $\mathbf{r}^{(k,m)} = \mathbf{Y}^{(k)} - \mathbf{Y}^{(m)} = \langle r_1^{(k,m)}, r_2^{(k,m)}, r_3^{(k,m)} \rangle$, $r^{(k,m)} = \|\mathbf{r}^{(k,m)}\|$, $\Lambda^{(k,m)} = \left[\frac{3}{4}\frac{a^{(m)}}{r^{(k,m)}} - \frac{3}{4}\left(\frac{a^{(m)}}{r^{(k,m)}}\right)^3\right]\frac{1}{(r^{(k,m)})^2}$ and $\beta^{(k,m)} = \left[\frac{3}{4}\frac{a^{(m)}}{r^{(k,m)}} + \frac{1}{4}\left(\frac{a^{(m)}}{r^{(k,m)}}\right)^3\right]$.

Now, with $\alpha_{i,j}^{(k,m)}$'s, Eq. (5) can be rewritten as follows:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_k \\ v_k \\ w_k \end{pmatrix} + \underbrace{\sum_{m=1}^{N_p}}_{m \neq k} \underbrace{\begin{pmatrix} \alpha_{1,1}^{(k,m)} & \alpha_{1,2}^{(k,m)} & \alpha_{1,3}^{(k,m)} \\ \alpha_{2,1}^{(k,m)} & \alpha_{2,2}^{(k,m)} & \alpha_{2,3}^{(k,m)} \\ \alpha_{3,1}^{(k,m)} & \alpha_{3,2}^{(k,m)} & \alpha_{3,3}^{(k,m)} \end{pmatrix}}_{\alpha^{(k,m)}} \begin{pmatrix} u_m \\ v_m \\ w_m \end{pmatrix} = \underbrace{\sum_{m=1}^{N_p}}_{m \neq k} \alpha^{(k,m)} \begin{pmatrix} V_1^{(m)} - U_1^{(m)} \\ V_2^{(m)} - U_2^{(m)} \\ V_3^{(m)} - U_3^{(m)} \end{pmatrix} \tag{7}$$

where $\alpha^{(k,m)}$ is a 3×3 matrix defined on the left hand side of Eq. (7), $\mathbf{V}^{(m)} = \langle V_1^{(m)}, V_2^{(m)}, V_3^{(m)} \rangle$ and $\mathbf{U}^{(m)} = \langle U_1^{(m)}, U_2^{(m)}, U_3^{(m)} \rangle$. It is important to point out that k will not be equal to m in the definition of $\alpha^{(k,m)}$. We need to define $\alpha^{(k,k)} = I_3$, where I_3 is the 3×3 identity matrix, leading us to the following definition of $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} \alpha^{(1,1)} & \cdots & \alpha^{(1,m)} & \cdots & \alpha^{(1,N_p)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^{(k,1)} & \cdots & \alpha^{(k,m)} & \cdots & \alpha^{(k,N_p)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^{(N_p,1)} & \cdots & \alpha^{(N_p,m)} & \cdots & \alpha^{(N_p,N_p)} \end{pmatrix} \tag{8}$$

The $\alpha_{i,j}^{(k,m)}$'s are symmetric with respect to their subindices (i,j) but not with respect to their superindices (k,m), because particles do not necessarily have the same radii. For instance in (8) $\mathbf{A}_{3k-2,3m-2} \neq \mathbf{A}_{3m-2,3k-2}$ if particle $a^{(k)} \neq a^{(m)}$. Another reason is related to the truncation radius, i.e. particle m with radius $a^{(m)}$ could include particle k with radius $a^{(k)}$ inside its $H_{trunc}a^{(m)}$ interaction radius but particle k might not necessarily interact with particles m because it could have smaller radius. This implies that $\alpha^{(k,m)} \neq \mathbf{0}_3$ and $\alpha^{(m,k)} = \mathbf{0}_3$, where $\mathbf{0}_3$ is the 3×3 zero matrix. Notice that using interactions based on $H_{trunc}a^{(m)}$ is only one option. We could also use a fixed interaction radius for all the particles, for instance $H_{trunc}\max_m a^{(m)}$. This produces another matrix which is still consistent with the model. We discuss this in more detail in Sections 5 and 6.

Notice that $\mathbf{A}$ is a very sparse matrix, so the precomputation of the coefficients $\alpha_{i,j}^{(k,m)}$ for later use saves time compared to using a matrix-free approach where the coefficients $\alpha_{i,j}^{(k,m)}$ are computed every time a matrix–vector multiplication is needed. This is true as long as the number of interactions remains small.

The matrix $\mathbf{A}$ can be rewritten as $\mathbf{B}+\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Thus the right-hand-side $\mathbf{b}$ can be obtained as $\mathbf{By}$, where $\mathbf{y}=\langle \mathbf{V}_1^{(1)}-\mathbf{U}_1^{(1)}, \mathbf{V}_2^{(1)}-\mathbf{U}_2^{(1)}, \mathbf{V}_3^{(1)}-\mathbf{U}_3^{(1)}, \ldots, \mathbf{V}_1^{(N_p)}-\mathbf{U}_1^{(N_p)}, \mathbf{V}_2^{(N_p)}-\mathbf{U}_2^{(N_p)}, \mathbf{V}_3^{(N_p)}-\mathbf{U}_3^{(N_p)}\rangle^T$.

This family of linear system of equations can be summarized as a sequence of linear system of equations $\mathbf{A}_{(n)}\mathbf{x}_{(n)}=\mathbf{b}_{(n)}$, where n represents its temporal dependence. The matrix $\mathbf{A}_{(n)}$ and $\mathbf{b}_{(n)}$ represent the interaction matrix and right hand side, respectively, for the particles at nth time step.

A more elaborated approach could be used here to take advantage of the sequence of linear systems, however we found out that it is not computationally worthwhile to take that path, see Appendix B. However, we did take advantage of the solution $\mathbf{x}_{(n)}$ of $\mathbf{A}_{(n)}\mathbf{x}_{(n)}=\mathbf{b}_{(n)}$ as the initial guess for the next linear system $\mathbf{A}_{(n+1)}\mathbf{x}_{(n+1)}=\mathbf{b}_{(n+1)}$.

We would like to point out that Eq. (7) may be interpreted as a divergence-free interpolation method, as it is divergence-free by construction.

We avoided the use of direct methods, such as Gauss elimination or QR decomposition ([7]) or sparse aware direct methods ([29]), because of their high computational cost and/or difficult parallelization.

4. Domain decomposition and parallel implementation

[1] implemented a loop-level parallelization using OpenMP to speed up the solution procedure. Namely, they divided the RHS of (1) into threads. In order to simulate a larger domain size with a larger number of droplets, a higher level of parallelization is inevitable. Therefore we parallelize the computation of (1) using the Message Passing Interface (MPI) to numerically solve fluid flow, droplet hydrodynamic interactions and droplet equation of motion. Preliminary work in this direction has been performed by [30] who considered MPI implementations of various tasks in the hybrid DNS. We note that their MPI implementation did not include droplet hydrodynamic interactions.

Our MPI implementation is based on domain decomposition previously adopted for efficient MPI implementation of fast Fourier transform (FFT) in the pseudo-spectral simulation of fluid turbulence [31]. Fig. 1 shows the computational domain for both 1D and 2D decomposition [1].[1] The domain is decomposed in the direction perpendicular to gravity in order to minimize the number of droplets crossing the subdomains. This will consequently reduce the data communication leading to higher speed-up.

The solution of (1) could be achieved by truncating the hydrodynamic disturbance flow of a droplet (with radius a) to a distance of $H_{trunc}\times a$ (in this study, we chose $H_{trunc}=50$). To proceed with parallel implementation, we assume that the truncation sphere of any individual droplet can at most cover a space belonging to immediate neighboring subdomains. This assumption amounts to an upper-bound on the number of subdomains in each decomposed direction, as:

$$N_{subd1}\leqslant\left[\frac{L_{box}}{H_{trunc}\times a_{max}}\right] \tag{9}$$

where N_{subd1} is the number of subdomains in a single decomposed direction, a_{max} is the maximum radius of the droplets in the system, L_{box} is the domain size in decomposed direction and $[]$ is the integer part function. Since, the droplet radius is in the range of 10 to 60 μm, this assumption does not pose a severe restriction on number of cores and has the benefit of limiting the data communication to nearest neighbor subdomains. In this manner, data communication will be required with 2 subdomains in the 1D domain decomposition and 8 subdomains in the 2D decomposition.

As shown in Fig. 2, we have made data communication effective by creating a halo layer around each subdomain. The thickness of this halo region is preset based on the maximum truncation radius. In the solution of (1), for a given time step, particles are frozen to their host subdomains. We note that the communication in each method arises from different needs. For instance in the Block Gauss–Seidel method, all data transfers are near-field communications which are restricted to the host subdomain and halo region in the immediate neighbors. On the other hand, the GMRes method not only needs near-field communication but also requires global reduction operations in order to be able to compute norms and dot products which are the essential steps of GMRes algorithm, see Algorithm 1 in Appendix A.2 for details regarding communication step.

5. Perturbation analysis of the linear system

Bounds for the convergence of GMRes are well understood for certain type of matrices, such as positive definite matrices. Our system of equations does not necessarily fall into one of these categories. For our special type of matrix, a numerical study is carried out to extract the important features that drive the behavior of GMRes. Before proceeding with the bounds for the behavior of GMRes, we need to discuss how the linear system behaves and what will be the correct quantities to explain the behavior of GMRes method.

Our first step was to try out GMRes with several test problems to see its behavior. Initial test problems utilized 256^3 and 512^3 grid resolution with particle sizes of 10–15 μm, 10–60 μm and 55–60 μm and number of particles set to 500 K, 1 M to 2 M. We found that GMRes performed consistently well, it converged with a relative error of 10^{-15} in less than 35 iterations over long periods of simulation. The accuracy of 10^{-15} was chosen to show that we can make the relative residual converge to machine precision, however for practical experiments we usually choose 10^{-6}.

[1] Since each subdomain is assigned to a single processor, we may use the terms "process", "processor" or "subdomain" interchangeably.

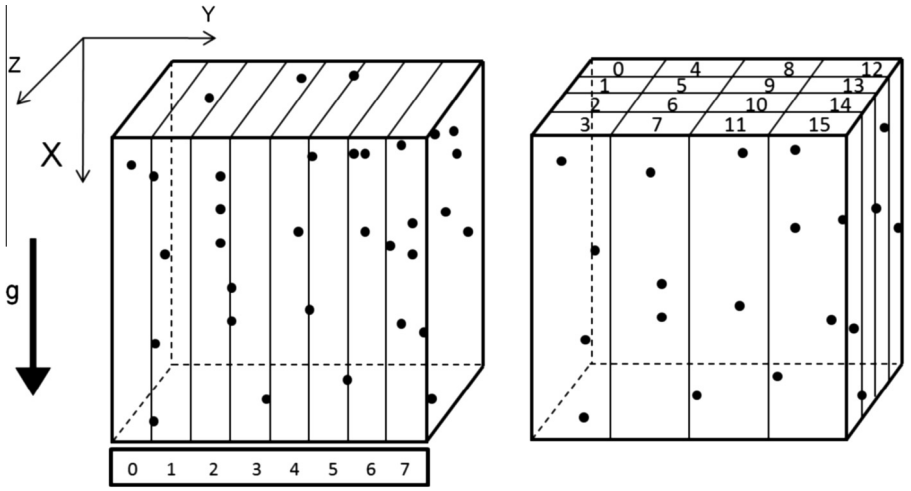


Fig. 1. Two spatial domain decomposition. Left: 8 subdomains in 1D decomposition, Right: 16 subdomains in 2D decomposition. Fluid nodes and droplets in each subdomain are assigned to an individual processor. In this Fig. 4 and 8 subdomains are used for demonstration purpose, though usually higher number of divisions are used.

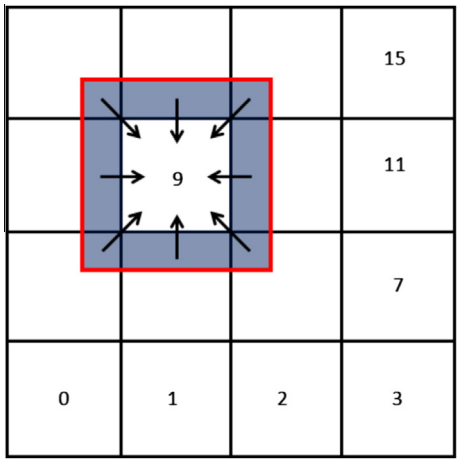


Fig. 2. Dark region around subdomain 9 shows the "halo region" in 2D domain decomposition. Particles' data in this region is made available for subdomain 9 via communications from 8 neighboring subdomains. Arrows indicate the direction of data communication.

We decided then to target a more specific behavior of the particles to push the linear system to behave *poorly*. To obtain this, we looked at this problem as it were a radial basis function (rbf) interpolation of a vector field. This is because from rbf's theory [32,33] we know that the related matrices obtained becomes ill-conditioned as the particles cluster, assuming the *size* of the particle is kept fixed. Thus, we expect to obtain a poor behavior of the matrix **A** as particles become clustered. This was indeed observed when the particles are clustered.

We have identified and studied several features of the problem that affect the convergence of GMRes. These features are the followings:

- Truncation radius
- Different size of particles
- Symmetry of truncation
- Normality of induced matrices
- Proximity of particles of equal and different sizes

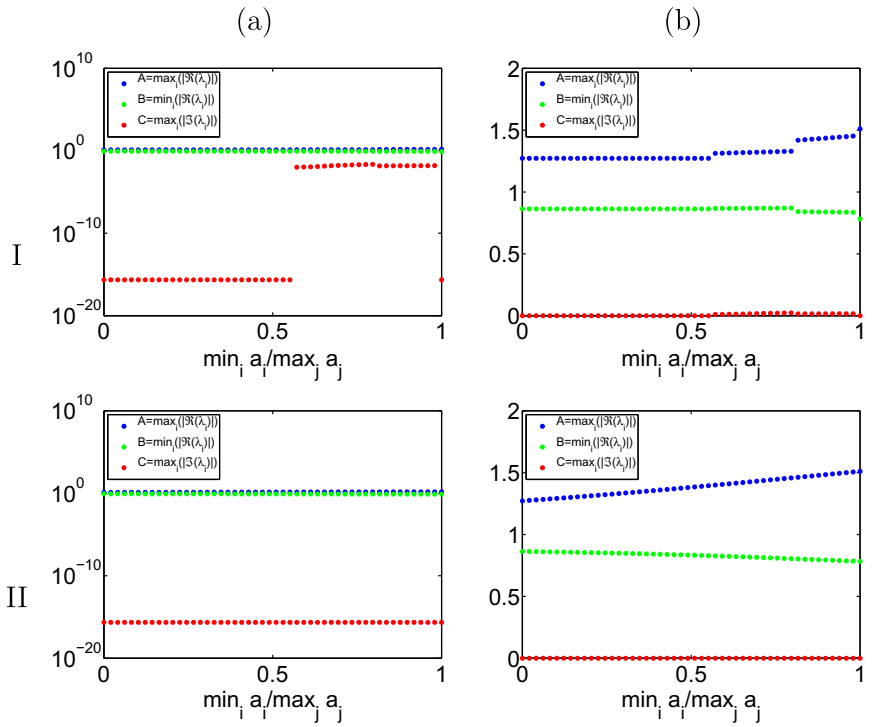
Although we were able to identify the features, it might not be possible to uncouple them explicitly to show their effect independently. We did not use periodic boundary conditions for experiments of this section. We considered the experiments as if domain was a small piece of a larger domain.

The first experiment we performed was to study how asymmetric and symmetric interactions of the droplets affected the interaction matrix. An asymmetric interaction is when the truncation radius is based of the particle radii. This means that the support of influence of particle i is $50a_i$, which implies that particle i will interact with particle j if $r_{ij} \leqslant 50a_i$, where $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$. A symmetric interaction is when the truncation radii is the same for all the particles. This means that the support of particle i is $50\max_k a_k$, which implies that all the particles have the same support. Thus, if particle i interact with particle j, particles j also interacts with particle i. This does not necessarily produces a symmetric matrix. It will produce a symmetric matrix if all the particles have the same radii.

To understand the properties of the linear system, we will study the distribution of the eigenvalues. In Table 1 row I, we observe the bounds $\max_i(|\Re(\lambda_i)|), \min_i(|\Re(\lambda_i)|)$ and $\max_i(|\Im(\lambda_i)|)$ for the eigenvalues using asymmetric interactions. In this experiment, we have 5^3 particles of two different size on a regular cubic mesh without periodic boundary conditions. We set the radii of the largest particles in the domain such that they have, on average, 20 neighbors in its support of influence. This means that particle i will have on average 20 particles that are in a distance smaller than $50\max_k a_k$ from particle i. The number of neighboring particles, in this study, is determined by the upper limit of droplet concentration in dilute suspension of warm-rain clouds. The largest radii is kept fixed and the ratio of the radii is indicated on the x-axis.

Table 1 row I shows a discontinuity in the behavior of the imaginary part of the eigenvalues. There are two discontinuities, the first is located about $\min_i a_i/\max_j a_j = 0.6$ and the second one is located about $\min_i a_i/\max_j a_j = 1$. The first discontinuity happens when the small droplets start to interact with the large droplets. For values of $\min_i a_i/\max_j a_j$ less than 0.6 the large droplets were interacting among them and with the small droplets. However the small droplets were not interacting with anybody. The second discontinuity appears when the small droplets reach the size of the large droplets, i.e. $\min_i a_i/\max_j a_j = 1$. This actually implies that the matrix becomes a symmetric matrix, so the eigenvalues must be real. There-

Table 1
The effect of asymmetric and symmetric interactions on the eigenvalues. The particles have two different radii and the locations are fixed. Row I shows the experiment with asymmetric interactions and row II shows the experiment with symmetric interactions. Column (a) uses a log scale and column (b) uses a linear scale. The maximum radii is set such that on average the largest particles interact with 20 neighboring particles. The ratio between the smallest and largest radii is indicated by the x-axis. The y-axis plots the bounds for the eigenvalues. "A" is the maximum absolute value of the real part of the eigenvalues, "B" is the minimum absolute value of the real part of the eigenvalues and "C" is the maximum absolute value of the imaginary part of the eigenvalues. Particle i affects particles inside its $50a_i$ radius of influence. Notice that particle i could affect particle j but particle j does not necessarily affect particle i, so it is not a symmetric interaction. Both experiments contain 5^3 particles in a regular mesh.



fore, asymmetric interactions explain the discontinuous behavior of the imaginary part of the eigenvalues. For numerical experiments particles keep their radii fix, such that $\min_i a_i / \max_j a_j$ is constant.

In Table 1 row II, we observe the bounds $\max_i(|\Re(\lambda_i)|)$, $\min_i(|\Re(\lambda_i)|)$ and $\max_i(|\Im(\lambda_i)|)$ for the eigenvalues using symmetric interactions. In this case the imaginary part of the eigenvalues is not discontinuous, however we observe a slight increase on the range of the real part of the eigenvalues with respect to the asymmetric interactions.

To better understand what it means to have asymmetric and symmetric interactions, we built two small examples. In rows I and II of Table 2 we have the figures of the first example with 5^3 particles distributed on a cubic mesh. In rows III and IV of Table 2 we have the figures of the first example with 10^3 particles distributed on cubic mesh. Column (a) of Table 2 shows the sparsity pattern of the respective matrices. Column (b) of Table 2 shows the behavior of the eigenvalues. Here we can confirm that for symmetric interactions the range of the real part of the eigenvalues is larger than for asymmetric interactions. Finally, column (c) of Table 2 shows the behavior of the relative residual of GMRes. The right hand sides were generated as $\mathbf{b} = (\mathbf{A} - \mathbf{I})\mathbf{b}_r$, where $\mathbf{b}_r$ is a random vector, see (7). It requires two more iterations to reach convergence in the symmetric case when compared to the asymmetric case.

Table 2
Comparison of several experiments. Column (a) indicates nonzero elements (nz) of **A**, column (b) shows the distribution of eigenvalues of **A** and column (c) shows the relative residual produced by GMRes. The particles have on average 20 neighbors. 50% of the particles have radii a_1 and the rest have radii $a_2 = 0.7a_1$. Rows I and III indicate experiment with asymmetric interaction and rows II and IV indicate symmetric interactions. Rows I and II indicate experiments with 5^3 particles and rows III and IV indicate experiments with 10^3 particles. The particles are located on a uniform mesh. Notice that the vertical scale differs from the horizontal scale in column (b).

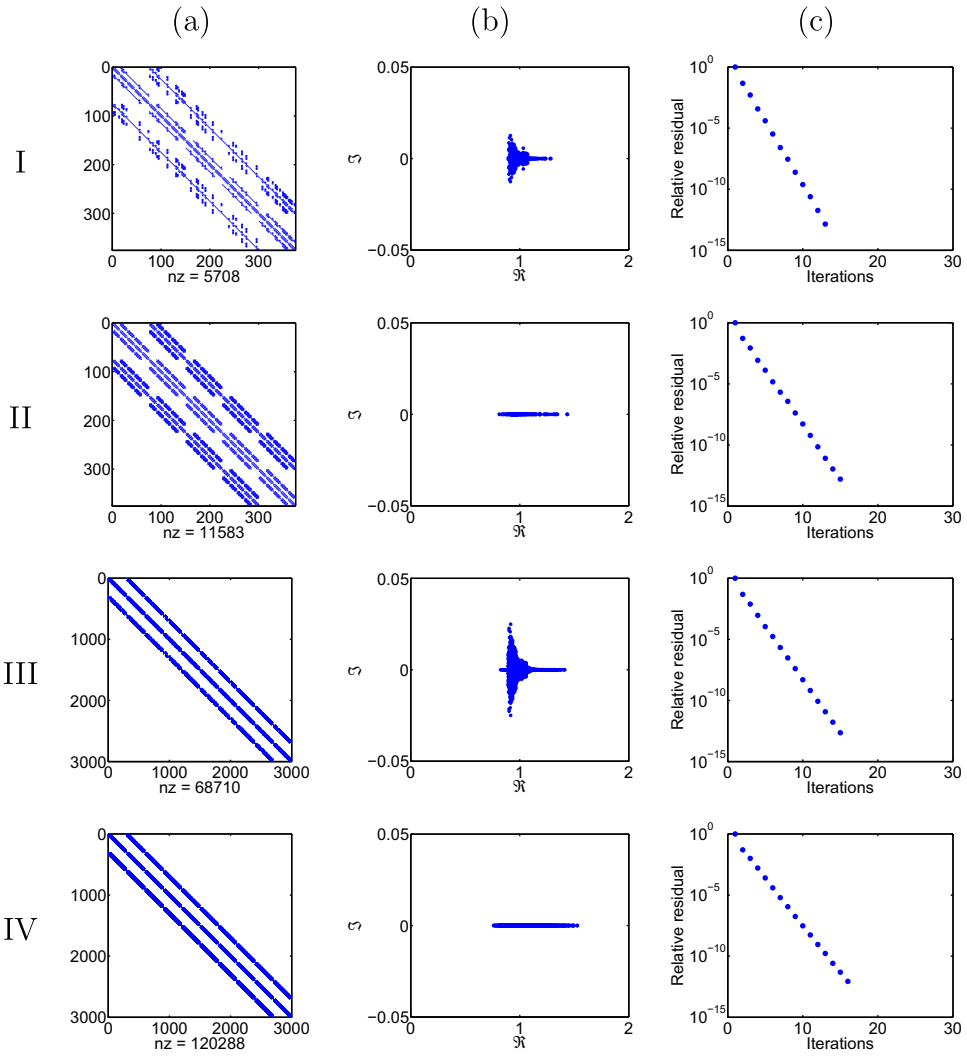


Table 3
Quantitative analysis of experiments shown in Table 2. Refer to Eq. (10), Fig. 3 and the upper bound $\|\mathbf{b}-\mathbf{A}\mathbf{x}_n\|/\|\mathbf{b}\| \leqslant \mathcal{C}\kappa_2(V)\rho^n$. The parameters λ_m and λ_M have been set based on the eigenvalues. The parameter δ has been obtained after an ellipsis has been circumscribed on the eigenvalues and it is dependent on ϵ. The parameter ϵ has been introduced in the analysis as a perturbation to the left bound λ_m to show its effect on ρ.

	ρ	$\kappa_2(V)$	λ_M	λ_m	δ
I	$0.103-0.034\epsilon+\mathcal{O}(\epsilon^2)$	284.416	1.285	$0.8949-\epsilon$	$0.029-0.689\epsilon+\mathcal{O}(\epsilon^2)$
II	0.139	1.251	1.417	0.820	0
III	$0.149+0.2494\epsilon+\mathcal{O}(\epsilon^2)$	2037.577	1.415	$0.821-\epsilon$	$0.0325-0.126\epsilon+\mathcal{O}(\epsilon^2)$
IV	0.173	9.4292	1.530	0.758	0

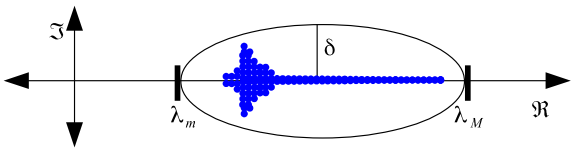


Fig. 3. Ellipse enclosing all eigenvalues of matrix **A**. The eigenvalues are represented by the blue dots. λ_M is an upper bound for the largest real part of the eigenvalues. λ_m is a greater than zero lower bound for the smallest real part of the eigenvalues. δ is an upper bound for the largest imaginary part of the eigenvalues. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

To be more precise, we would like to provide the actual estimation for ρ and $\kappa_2(V)$ for the test cases shown in Table 2 and understand better column (c) of the same table. In Table 3 we observe a direct comparison of ρ and $\kappa_2(V)$ for the symmetric and asymmetric interactions. This shows that for symmetric interaction we obtain a better $\kappa_2(V)$ but with the cost of a larger ρ compared with asymmetric interactions.

Up to this point, we have observed by numerical experiments that the eigenvalues of the matrix **A** tend to have a positive real part and may have a small imaginary part, depending if asymmetric or symmetric interactions are used. We have observed good convergence behavior of the relative residual for GMRes for both asymmetric and symmetric case.

The particular behavior of the eigenvalues shown in Table 2 column (b) allow us to follow [34,35,4] to bound the behavior the relative residual produced by GMRes. This is carried out by enclosing the eigenvalues by an ellipse, see Fig. 3. This allows us to find an explicit bound for the behavior of the relative residual produced by GMRes, this is $\|\mathbf{b}-\mathbf{A}\mathbf{x}_n\|/\|\mathbf{b}\| \leqslant \mathcal{C}\kappa_2(V)\rho^n$. $\rho \leqslant 1$ is a constant, n is the iteration number, $\mathcal{C}$ is a constant independent of n and $\kappa_2(V) = \|V\|\|V^{-1}\|$. V is the matrix with the eigenvectors of **A**, such that $\mathbf{A} = V\Lambda V^{-1}$ is the eigenvalue decomposition of **A**, Λ is the diagonal matrix with the eigenvalues of **A**. ρ measures the ratio between a characteristic radius of the ellipsis over the distance of the center of the ellipse to the origin [35]. Thus, we must avoid the origin when we enclose the eigenvalues otherwise ρ is useless. This leads us to the following definition of ρ,

$$\rho = \frac{\lambda_M - \lambda_m + 2\delta}{\lambda_M + \lambda_m + 2\sqrt{\lambda_M\lambda_m + \delta^2}} \tag{10}$$

where λ_M is an upper bound for the largest real part of the eigenvalues, λ_m is a greater than zero lower bound for the smallest real part of the eigenvalues and δ is an upper bound for the largest imaginary part of the eigenvalues. We choose λ_M, λ_m and δ such that they minimize ρ and all the eigenvalues are enclosed by the ellipse. This result reduces to the well-known result for positive definite matrices ([35]) by taking the Taylor expansion of ρ as a function of δ and letting $\delta \to 0$,

$$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} + \frac{2\delta}{(\sqrt{\lambda_M}+\sqrt{\lambda_m})^2} + \mathcal{O}(\delta^2) \tag{11}$$

where $\kappa = \lambda_M/\lambda_m$ is the condition number of **A**. Notice that when the matrix is positive definite it is enough to use $\lambda_M = \max_i(|\Re(\lambda_i)|)$, $\lambda_m = \min_i(|\Re(\lambda_i)|)$ and $\delta = 0$.

For problems where the eigenvalues show different patterns, new estimates must be computed [35–37].

Since ρ is small, GMRes will perform well for our test problems. However, we still have to describe the behavior of $\kappa_2(V)$ to have a tight bound for GMRes, i.e. we need $\kappa_2(V) \approx 1$. In Fig. 4, we show the behavior of $\kappa_2(V)$ for asymmetric and symmetric interactions. We observe that for the case of asymmetric interactions $\kappa_2(V)$ increases as we increase the number of particles. This implies that the matrix induced by the asymmetric interactions behaves as a nonnormal matrix as we increase the number of particles. On the other hand, the matrix induced by the symmetric interactions shows a bounded behavior of $\kappa_2(V)$, which suggests that it behaves as a normal matrix. This apparent bad behavior [5] might not be that harmful after all, remember that from column (c) of Table 2 we know that GMRes converged slightly faster for the asymmetric case than the symmetric case. This faster convergence rate is explained by ρ, i.e. for the asymmetric case ρ is slightly smaller than the symmetric case. So, even though $\kappa_2(V)$ increases with the number of droplets, this is overcome by ρ^n. Another important advantage for the asymmetric case over the symmetric case is its computational cost. Even though that both of them are sparse, the

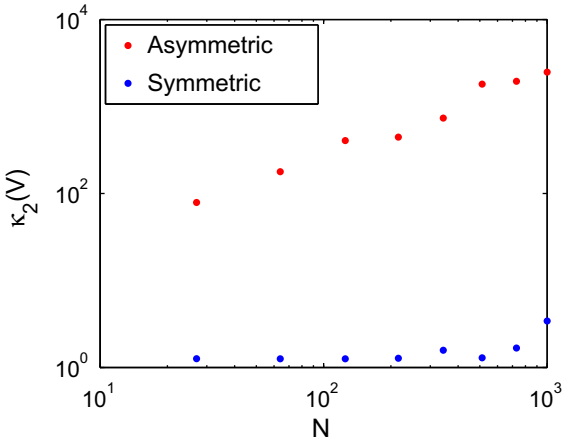
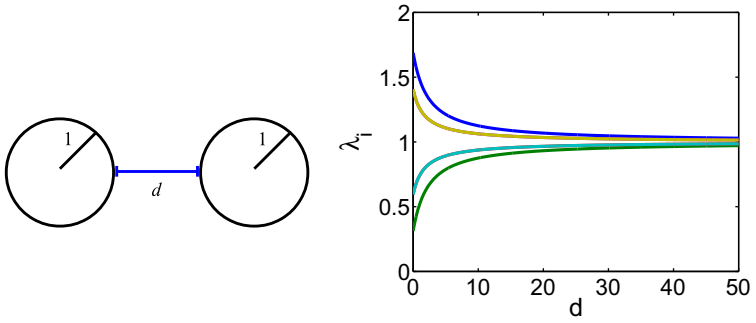


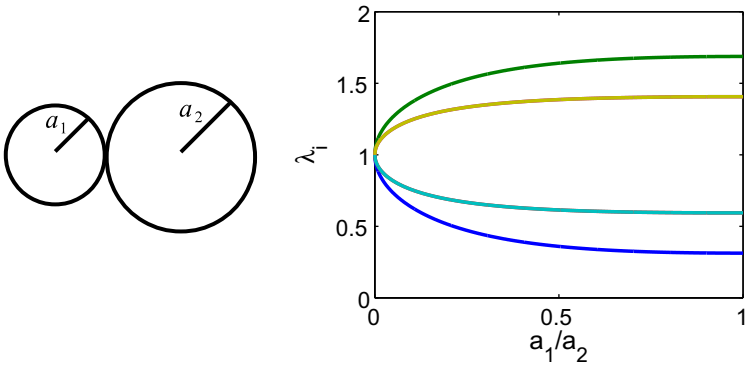
Fig. 4. This experiment compares the behavior of $\kappa_2(V)$ for the asymmetric and symmetric interaction of the particles. $\kappa_2(V) = \|V\|_2\|V^{-1}\|_2$, where V is the matrix with the eigenvectors of $\mathbf{A}$, such that $\mathbf{A} = V\Lambda V^{-1}$, where Λ is the diagonal matrix with the eigenvalues of $\mathbf{A}$.



(a) Two particles approaching each other

(b) Distribution of eigenvalues

Fig. 5. Experiment showing the behavior of the eigenvalues for two particles with normalized radii approaching each other. The distance between the particles is $2 + d$, so, since the particles have normalized radii, $d = 0$ implies they are touching each other.



(a) Two particles touching each other

(b) Distribution of eigenvalues

Fig. 6. Experiment showing the behavior of the eigenvalues when two particles of different sizes are touching each other. a_1 is the radius of the smaller particle and a_2 is the radius of larger particle.

asymmetric case is sparser so less computation is required when solving the linear system. We will show in Section 6 that the use of asymmetric interactions is after all the best.

Another part of the problem we would like to study in more detail is the behavior of the eigenvalues as two particles approach each other and when the particles are touching each other. This could be seen as an extreme case for interacting particles. These two experiments are performed in an unbounded domain, i.e. no periodic boundary conditions. In Fig. 5 (a) we show an sketch of the first experiment of two particles of the same radii approaching each other. in Fig. 6 (a) we show a sketch of the second experiment of two particles with different radii touching each other.

For simplicity the distance between the particles has been normalized by their radii. In Eq. (12) we write the explicit definition of the interaction matrix. For simplicity, the locations of the particles have been aligned on the x-axis.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \widehat{\Lambda} & 0 & 0 \\ 0 & 1 & 0 & 0 & \widehat{\beta} & 0 \\ 0 & 0 & 1 & 0 & 0 & \widehat{\beta} \\ \widehat{\Lambda} & 0 & 0 & 1 & 0 & 0 \\ 0 & \widehat{\beta} & 0 & 0 & 1 & 0 \\ 0 & 0 & \widehat{\beta} & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

where $\widehat{\Lambda} = \frac{1}{4(2+d)^3} + \frac{3}{4(2+d)} + \frac{3}{4}\left(\frac{1}{2+d} - \frac{1}{(2+d)^3}\right)$, $\widehat{\beta} = \frac{1}{4(2+d)^3} + \frac{3}{4(2+d)}$ and the distance between the particles is $2+d$. The eigenvalues have been found to be,

$$\lambda_1 = \frac{(3+d)^2(3+2d)}{2(2+d)^3} \tag{13a}$$

$$\lambda_2 = \frac{(1+d)^2(5+2d)}{2(2+d)^3} \tag{13b}$$

$$\lambda_3 = \lambda_4 = \frac{(1+d)(19+d(17+4d))}{4(2+d)^3} \tag{13c}$$

$$\lambda_5 = \lambda_6 = \frac{1}{4}\left(4 + \frac{1}{(2+d)^3} + \frac{3}{2+d}\right) \tag{13d}$$

In Fig. 5 we show the behavior of the eigenvalues as a function of d. We observe that the eigenvalues spread out as $d \to 0$. This suggests that having two particles close to each other may slow down the convergence behavior of GMRes. This is seen directly from the definition of ρ in Eq. (10). Another important theoretical feature that can be obtained from the eigenvalues of (12) in Eq. (13), is that when $d = -1$ the matrix is singular, i.e. $\lambda_2 = \lambda_3 = \lambda_4 = 0$. Notice that when $d = -1$ the particles overlap but they are not on top of each other. Thus, when $d = -1$ the distance between the centers is 1. This is very interesting behavior from the theoretical point of view but an unreachable state from the physical point of view, since particles are not allowed to overlap.

The second experiment explores further the behavior of the eigenvalues when two particles are touching each other. In Fig. 6 we observe the behavior of the eigenvalues for a complete range of different radii. The experiment was design such that one particle keeps its radius fixed and the other one changes its radius. Notice that even one particle is changing its size, they are always touching. The conclusion from this experiment is clear, the eigenvalues will spread out most when two particles of the same size are touching each other.

The conclusion from these *two* experiments with two particles is clear. The eigenvalues of the matrix will spread out most when two particles of the same size are touching each other. For any other configuration, such as when the particles are not touching or have different size, the effect on the eigenvalues will be less severe.

Finally, we performed an experiment with 8 particles having the same radii arranged on the corners of a cube. The purpose of this experiment is to explore how a cluster affects the behavior of the eigenvalues. The radii of the particles is controlled by the nondimensional parameter ξ. The nondimensional parameter ξ defines the radii of the particles such that when $\xi = 0$ the particles have radii equal to 0 and when $\xi = 1$ the closest particles in the cubic mesh are touching each other. We found these two configurations lead to a very interesting behavior of the eigenvalues. For the two cases we use symmetric interactions. In Table 4 we summarize the outputs found for the two configurations.

The main purpose of all the experiments in this section was to understand how the configuration of the particles affects the convergence of GMRes. We have shown that the convergence of GMRes is controlled by two components. These are the normality of the matrix and distribution of its eigenvalues. We have shown how to overcome the non-normality of the inter-

Table 4
Showing ill-conditioning and negative eigenvalues of the matrix for our linear system.

ξ	$\max_i(\Re(\lambda_i))$		ρ
0.946851675535042	3.71	$\min_i(\lvert\Re(\lambda_i)\rvert) = 1.2033 \times 10^{-8}$	0.9998
0.97	3.78	$\min_i(\Re(\lambda_i)) = -0.0116$	Does not apply

action matrix by using symmetric interaction. However, as mention before, we decide to use asymmetric interactions. We discuss this point further in Section 6.

On the other hand, we have not discussed how to overcome a bad distribution of eigenvalues. As we showed before, when the particles cluster they could induce a poor configuration of the eigenvalues. Although in cloud physics this might not be common, mathematically the matrix could reach a configuration such that GMRes might behave poorly. Therefore, we must prepare to handle this issue in case it happens. To handle a bad distribution of eigenvalues, we consider using a preconditioner. This preconditioner was already described in Appendix A.3, however at this point we have only shown why we need to have a preconditioner. We shall discuss in Section 6 the mathematical effectiveness and computational efficiency of this type of preconditioner.

6. Scalability

To solve system (1) on a 1D decomposed domain, we implemented the following methods:

1. Block Gauss–Seidel explained in Appendix A.1.
2. GMRes explained in Appendix A.2.
   (a) GMRes with a matrix-free approach
   (b) GMRes with a parallel sparse matrix implementation, discussed in Section 3.
3. Preconditioned GMRes explained in Appendix A.3.
   (a) Preconditioner with no overlap
   (b) Preconditioner with overlapping

For 2D domain decomposition, we only implemented GMRes with a matrix-free approach and with the parallel sparse representation of the matrix.

In HDNS simulations, output quantities of interest are as follows: radial relative velocity which is the relative velocity of droplet pairs projected on their center line, radial distribution function is a measure of local clustering of droplets, and collision kernel is the normalized rate of collisions between droplets per unit time per unit volume. The results (radial relative velocity, radial distribution function and collision kernel) from all methods discussed in this paper are strictly validated against each other.

Symmetric and asymmetric interaction of the droplets are compared in Fig. 7. With symmetric interaction, particles are allowed to interact within a truncation sphere of $H_{trunc} \times a_{max}$ in radius while asymmetric interaction is restricted to a sphere of radius $H_{trunc} \times a_i$. As expected, symmetric interaction takes a few (~ 2) more iterations to converge in comparison with asymmetric interaction. For symmetric interactions, each particle on average has more neighboring particles and the computational cost per iteration is slightly higher than the asymmetric interaction as shown in Table 5.

For the rest of this section, we will only consider asymmetric interactions due to its lower computational cost.

The scaled performance of proposed methods are shown in Fig. 8. The timing measurements are carried out for a benchmark problem of 2×10^6 droplets of radii 20 and 40 microns at 512^3 grid resolution. For the overlapped preconditioner, the

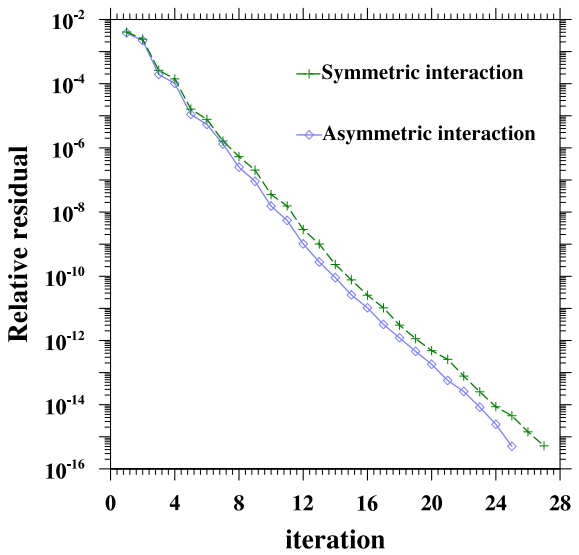


Fig. 7. Convergence rate for symmetric and asymmetric interaction (2D decomposition, 512^3 simulation on $N_{proc} = 256$ cores, $N_p = 2 \times 10^6$, $a_1 = 10\ \mu m$ and $a_2 = 60\ \mu m$).

Table 5
Effect of symmetric and asymmetric interaction on approx. number of iterations to converge and timing of GMRes with sparse representation of **A** (2D decomposition, 512^3 simulation on $N_{proc} = 256$ cores, $N_p = 2 \times 10^6$, $a_1 = 10\,\mu$ and $a_2 = 60\,\mu$, $\|\mathbf{Ax} - \mathbf{b}\|/\|\mathbf{b}\| \leqslant \varepsilon$).

Interaction	Fix cost	Variable cost	$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-13}$	
			# Iter.	Total time (s)	# Iter.	Total time (s)
Asymmetric	0.0093	0.0058	7	0.0499	20	0.1253
Symmetric	0.0117	0.0081	7	0.0684	21	0.1818

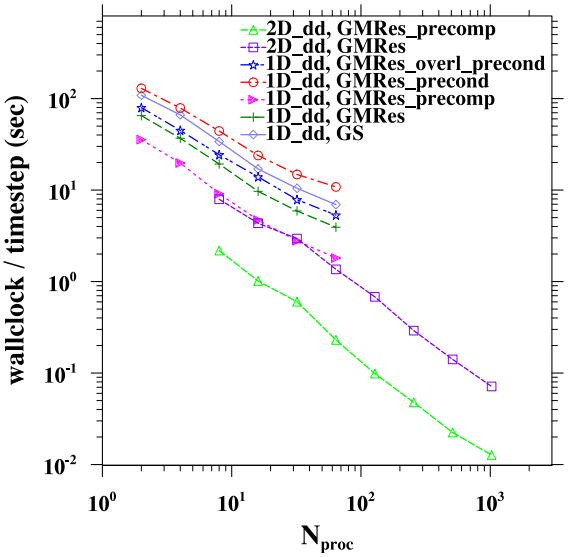


Fig. 8. Scalability of different proposed methods.

overlapping thickness in all 3 directions is set to $H_{trunc} \cdot a_{max}$ with $H_{trunc} = 50$. All measurements are carried out on NCAR's Bluefire machine with identical compiler and optimization settings. The convergence criterion is set to $\varepsilon = 10^{-6}$ and measurements are averaged over 1000 time-steps of simulation. This value of ε is the acceptable measure we use in our simulations. We note that ε could be set as small as machine ε, but the average statistics we are interested in are not sensitive to ε when $\varepsilon \leqslant 10^{-5}$. Furthermore $\varepsilon = 10^{-6}$ allows us to simulate large problems in a manageable run time.

We observed following behavior for 1D decomposition: All implemented methods have an almost ideal scaling with the number of cores but start to saturate at about $N_{proc} = 64$. This suggests that 1D decomposition will not be the method of choice for larger problem sizes. Our proposed preconditioning methods have a high computation overhead so that it renders both methods to be less efficient than GMRes without preconditioning for dilute suspensions. This suggests that one should use the preconditioner when stagnation of GMRes is observed. Although we have not seen this yet, we know from the perturbation study in Section 5 that the matrix could become ill-conditioned, which would make GMRes stagnate.

Also in Fig. 8, we show the timing results for GMRes and Block Gauss–Seidel on a 1D and 2D decomposition implementation The 2D decomposition does not show any symptom of saturation up to 1024 cores suggesting that 2D implementation of GMRes and GMRes with sparse representation are highly scalable. Please also note that the sparse representation of **A** leads to the most efficient method both in 1D and 2D decomposition. As we discussed earlier, with precomputation of coefficients, we avoid a heavy computational burden in each iteration which highly increases the efficiency of the method. Namely, in 2D decomposition, precomputation yields a speed-up of ~ 5.6x compared to matrix-free GMRes on 1024 cores. Further, in Table 6, we have summarized the number of iterations of each method to converge to a preset ε along with wallclock time per iteration for each method. This table is obtained for our benchmark problem on 64 cores. Even though overlapped preconditioned GMRes takes fewer iterations, each iteration is slower compared to (for instance) precomputed GMRes. We note that it is possible to implement a sparse representation of Block Gauss–Seidel. However, even if we assume the cost per iteration becomes as low as sparse representation of GMRes, since the total number of iterations will remain the same (Table 6: 18 for BGS versus 11 for GMRes to achieve $\varepsilon = 10^{-6}$), the sparse representation of Block Gauss–Seidel will still be slower. Also we should add that Block Gauss–Seidel has a stricter convergence behavior. For instance Block Gauss–Seidel becomes extremely slow or it will diverge for large density of particles.

Table 6
Average number of iterations to converge and timing of proposed methods (1D decomposition, 512^3 simulation on $N_{proc} = 64$ cores, $N_p = 2 \times 10^6$, $a_1 = 20$ μ and $a_2 = 40$ μ, $\|\mathbf{Ax} - \mathbf{b}\|/\|\mathbf{b}\| \leqslant \varepsilon$).

Solution method	Fix cost	Variable cost	$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-13}$	
			# Iter.	Total time (s)	# Iter.	Total time (s)
Block Gauss–Seidel	0.1254	0.3785	18	6.9390	92	34.9474
Matrix-free GMRes	0.1163	0.3463	11	3.9259	31	10.8516
Non-Overlap. Precond. GMRes	0.1402	1.1873	9	10.8261	26	31.01
Overlap. Precond. GMRes	0.2138	2.5564	2	5.3266	4	10.4394
GMRes with Sparse Rep. of **A**	0.3591	0.1317	11	1.8084	31	4.4418

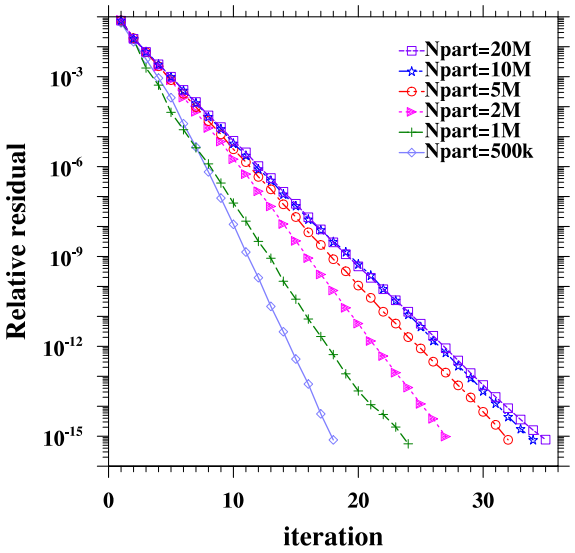


Fig. 9. Scaling of GMRes with sparse representation of **A** versus number of particles (2D decomposition, 512^3 simulation on $N_{proc} = 256$ cores, $a_1 = 20\ \mu m$ and $a_2 = 40\ \mu m$).

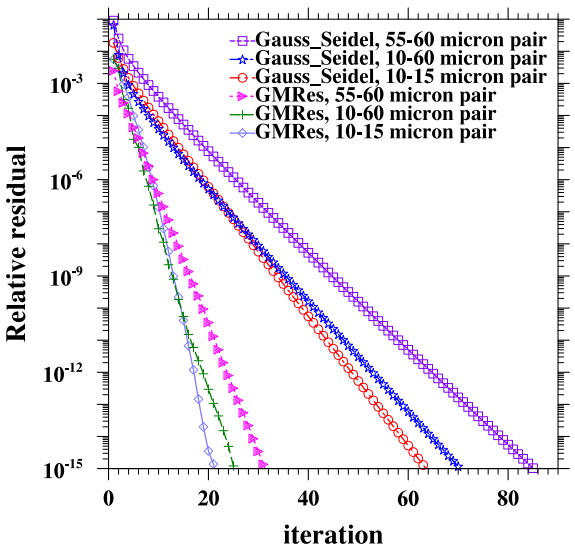


Fig. 10. Effect of particle size on the convergence rate of GMRes with sparse representation of **A** (2D decomposition, 512^3 simulation on $N_{proc} = 256$ cores, $N_p = 2 \times 10^6$).

In Fig. 8, the 2D domain decomposition codes could not be run on small number of cores due to memory requirements. On the other hand, Eq. (9) restricts the option of large number of cores for 1D domain decomposition.

Fig. 9 demonstrates the scaling of GMRes with sparse representation of **A** versus total number of particles in the domain. We observed that number of iterations has a relatively weak dependence on the number of particles. Namely, the number of iterations almost doubles for a 40-fold increase of number of particles (from 500 K to 20 M). The number of particles has an even weaker effect for larger number of droplets. For droplets in the range of application in cloud physics, the effect of particle size on the convergence rate of sparse representation is shown in Fig. 10. Figs. 9 and 10 are obtained by averaging the residuals over 50 time steps.

Next, we discuss the main drawback of GMRes with sparse representation of **A**. As discussed earlier, at least 6 coefficients need to be stored for each neighbor pair of particles in the simulation. Therefore, in total, $O(6n_{avg}N_p)$ values must be stored in total where n_{avg} is the average number of neighbors in the truncation sphere per droplet. The required memory might grow rapidly if N_p or n_{avg} become large for a specific simulation which might cause a drastic slow-down of the computation depending on the machine architecture and memory limitations. In our simulations, we do not reach these extreme limits, since n_{avg} is small (of $O(10)$) due to the nature of the physical problem we explore. Namely due to the dilute suspension of particles in clouds, the number of neighboring particles for a single droplet is bounded because the liquid water content is quite small for typical cloud physics applications.

7. Complexity analysis

We analyzed the complexity of the proposed subroutine with the 2D domain decomposition scheme to theoretically estimate the wall-clock execution time. This allows us to analyze how different variables affect the code performance. The expression is built by using the elemental times t_a, t_c, t_w, and t_s where t_a is the computation time per floating point operation, t_c 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 before actual communication occur. The reader can find in [38] a discussion on how to obtain these times for the case of a parallel 3D FFT. In this paper, we will use same elemental times, and they are shown in Table 7. These elemental times belong to the supercomputer Bluefire at NCAR, an IBM Power 575 cluster.

Our fastest implementation builds the sparse representation of **A** before the iterative process. Eq. (14) gives the wall-clock time of the precomputation step together with the initialization part (Line 1 and 2 of the GMRes algorithm in Appendix A.2).

$$t_{\text{Precomp}} = 124t_a + 21t_c + \left[16 + 3.32\log(N_{\text{proc}})\right]t_s \tag{14a}$$

$$+ (28t_c + 6t_w)\left(\frac{50a_{\max}}{L\sqrt{N_{\text{proc}}}} + 2500\left(\frac{a_{\max}}{L}\right)^2\right)N_p \tag{14b}$$

$$+ \left[\left(62 + 2\left(3 + \frac{100a_{\max}}{dx}\right)^3\right)t_a + \left(15 + 3\left(3 + \frac{100a_{\max}}{dx}\right)^3\right)t_c\right]\frac{N_p}{N_{\text{proc}}} \tag{14c}$$

$$+ (18t_a + 4t_c)\left(3 + \frac{100a_{\max}}{dx}\right)^3\left(\frac{dx}{L}\right)^3\frac{N_p^2}{N_{\text{proc}}} \tag{14d}$$

$$+ (48t_a + 6t_c)\left(\frac{2.5\times 10^5\pi}{3}\right)\left[\left(\frac{a_1}{L}\right)^3 + \left(\frac{a_2}{L}\right)^3\right]\frac{N_p^2}{N_{\text{proc}}}, \tag{14e}$$

where a_{max} is the largest droplet radii ($= \max(a_1, a_2)$), L is the domain box size, and dx is the cell unit size of the particle search grid used for the cell-index method and linked lists [28].

In this equation, there are two communication times; (1) the third term in line (14a) mainly represents the global communication needed at initialization (for this, MPI_All_Reduce command is utilized), and (2) the line (14b) gives the time it takes to transfer the data related to particles located in the halo layer around each subdomain (see Fig. 2). This term depends on the volume of the layer composed by four neighboring slabs and four corners. Note that for very large number of processors, the second communication time decreases until it saturates with a halo layer containing only corners while the first increases slowly.

The wall-clock time for the iterative process is:

Table 7
Estimated elemental times on Bluefire for the complexity analysis.

t_a	0.37	ns/FLOPS
t_c	6.37	ns/word
t_w	2.19	ns/word
t_s	7.00	μs

$$t_{\text{lte}} = n\left(\frac{851}{6} + \frac{11}{2}n + \frac{5}{3}n^2\right)t_a + n\left(\frac{33}{2} + \frac{3}{2}n\right)t_c \tag{15a}$$

$$+ n\left[16 + 1.66\left(\frac{3}{2} + \frac{n}{2}\right)\log(N_{\text{proc}})\right]t_s \tag{15b}$$

$$+ n(28t_c + 6t_w)\left(\frac{50a_{\max}}{L\sqrt{N_{\text{proc}}}} + 2500\left(\frac{a_{\max}}{L}\right)^2\right)N_p \tag{15c}$$

$$+ n[(20 + 9n)t_a + 18t_c]\frac{N_p}{N_{\text{proc}}} \tag{15d}$$

$$+ n(19t_a + 4t_c)\left(\frac{2.5 \times 10^5\pi}{3}\right)\left[\left(\frac{a_1}{L}\right)^3 + \left(\frac{a_2}{L}\right)^3\right]\frac{N_p^2}{N_{\text{proc}}}, \tag{15e}$$

where n is the number of iterations. Similar to Eq. (14), there are two types of communication times and they take place in Line 7 and 9 of the GMRes algorithm shown Appendix A.2. Note that some terms have the number of iterations (n) to the second or third power due to the double nested loops in the GMRes algorithm.

Both equations assumed a truncation radius of influence of $H_{trunc} = 50$ and a uniform spatial distribution of the droplets in the domain of size L. To simplify, we set dx to the same grid size used for flow simulations and the number of iterations n to 10. Setting n equal to 10 may be seen as being optimistic but this is indeed a good approximation because it gives us a lower bound for the total cost of the complexity analysis. dx should be any small enough value to reduce the N_p^2 dependence but not smaller than $2(a_1 + a_2)$. The number of iterations depends on the problem setting, however we have shown before that it depends weakly on N_p and droplet sizes (a_1, a_2). 10 iterations is an average value to reach the 10^{-6} convergence criteria (see Figs. 9 and 10). In the context on cloud physics, for a fixed liquid water content (typically about 1 g/m^3), the number of droplets N_p is determined by the box size L and droplet sizes (a_1, a_2). With these premises, in reality the total wall-clock time depends only on the problem size (L), droplet sizes (a_1, a_2), and the number of processors (N_{proc}).

Before examining these dependencies, we show in Fig. 11 the relative importance of each of the terms in the complexity analysis using the case discussed in Fig. 8. We note that the total wall-clock time of the subroutine is dominated by the precomputation and initialization, Eq. (14), (mainly for small to moderate number of processors) and by the global communication (MPI_All_Reduce command) during the iteration process, line (15b) of Eq. (15), (mainly for large number of processors). The latter effect is due to the $\log(N_{proc})$ dependency of the communications, which leads to a poor scalability of the subroutine.

We note that the time for communication of data of particles located in the halo layer (line (15c) of Eq. (15)) is negligible in comparison to the leading terms (similar behavior was found for the same term in Eq. (14)). The theoretical total wall-clock time matches the data of the case 2D_dd GMRes with precomputation from Fig. 8, which validates our complexity analysis.

In Fig. 12 we show the dependency of theoretical wall-clock execution time on the problem size (L), droplet sizes (a_1, a_2), and the number of processors (N_{proc}). For similar numbers of droplets, the wall-clock execution time is similar, thus the subroutine depends strongly on the number of particles. From the figure we can conclude that our parallel GMRes implementation is suitable for a moderate number of processors in the tera to petascale computers, depending on the size of the

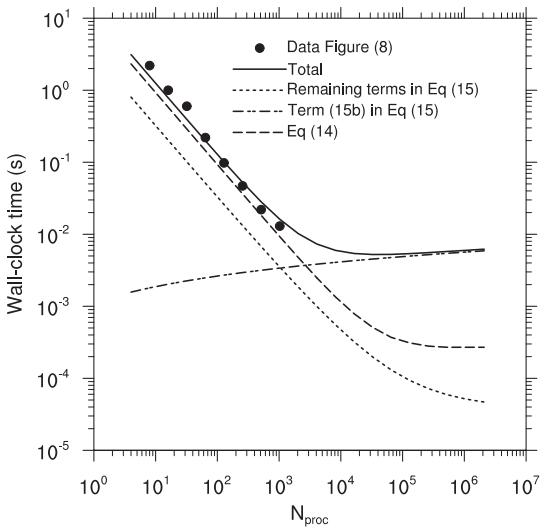


Fig. 11. Complexity analysis of each term of the theoretical wall-clock execution time, using the case discussed in Fig. 8.

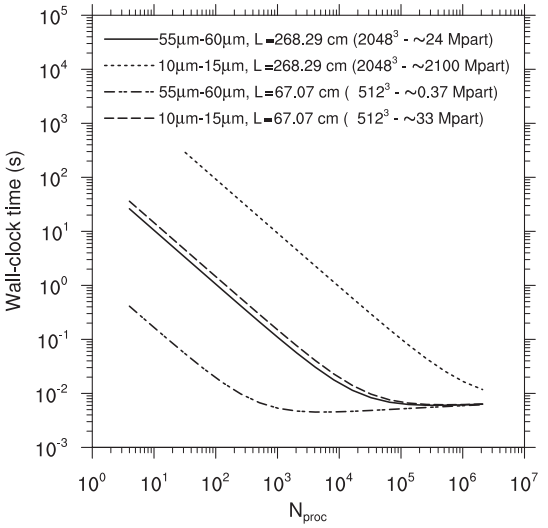


Fig. 12. Complexity analysis of the theoretical wall-clock execution time for different cloud physic cases.

problem. Further research is needed toward reducing the global communications, as it becomes the bottleneck for the subroutine for very large number of processors. There are few options we could explore. Although Block Gauss Seidel is more computationally expensive, it does not require global communications. We showed that the use of preconditioners in GMRes increase the computation time but greatly reduces the number of iterations. This could also be used to our advantage for a large number of processors because the computation effort is negligible in comparison to the global communication which is reduced as the number of iterations are reduced. We point out that our aim is to simulate large problems with problem sizes of 2048^3 or larger. For those cases, our subroutine scales well for moderate to large numbers of processor.

We also performed a weak scalability analysis (not shown) and we found that the scalability deteriorates with number of processors because, while the computational load per processor is constant, global communication dominates the subroutine's performance.

8. Summary and conclusions

In this paper, we have studied the mathematical and computational properties of the linear system obtained from the hydrodynamic interaction of cloud droplets in a turbulent flow, in order to accelerate the convergence rate of the relevant iterative solver. The hydrodynamic interactions have been modeled by the Stokes disturbance flow approximation, see [2]. We compared the original Block Gauss–Seidel solver (explained in Appendix A.1), with a GMRes solver.

We provided a complete mathematical description of the linear system. Specific properties of the system matrix **A** such as eigenvalue distribution and normality of the matrix were analyzed to understand the convergence of the linear solver. We studied the performance of GMRes with and without a preconditioner, using a matrix-free and sparse representation of the matrix and on a 1D and 2D domain decompositions. We provided theoretical understanding of the convergence rate for GMRes method. The results were compared to the Block Gauss–Seidel method originally used by [1].

This study shows that the linear system could behave very poorly under certain conditions, such as strong clustering of particles. Although, in cloud physics this extreme clustering is unlikely, we should be aware that the Stokes approximation produces an ill-conditioned linear system, i.e. the matrix of hydrodynamic interaction may have a large condition number $\kappa = \max_i|\lambda_i|/\min_j|\lambda_j|$, where λ_i represent the eigenvalues of the matrix. By analyzing the eigenvalues, we also showed that the closeness of similar-size droplets affects the conditioning of the matrix as opposed to different-size droplets. Namely, the worst case of ill-conditioning will be same size particles in close proximity. For these situations, we developed a restricted Schwarz type preconditioner for the linear system which can handle the ill-conditioning, see A.3. This preconditioner was tested in a 1D domain decomposition implementation. However, the scalability results in Table 6 show that the preconditioning overhead dominates the method for our well-behaved system.

We also studied how the normality of the matrix can be obtained by having symmetric interactions instead of asymmetric interactions. Even when the matrix behaves as a nonnormal matrix, it did not yield a stagnation behavior in GMRes for our problem. However, for the rest of our experiments, we kept the nonnormal matrix for performance reasons. It is important to point out that by making the matrix a normal matrix we obtained a different behavior of the eigenvalues. However, the same bound derived in Section 5 holds.

We did a complete convergence analysis of the Block Gauss–Seidel method in AppendixA.1. However, we did not pursue further because GMRes was more computationally efficient and stable.

We demonstrated, quantitatively, that convergence rate of GMRes method saturates for high density of particles, see Fig. 9. Particle size has a slight effect on the convergence rate. We showed that 55–60 μm droplet pair requires ∼10 more iterations to converge to machine roundoff error when compared to 10–15 μm pair, see Fig. 10. Thus, for droplets in the range of application in cloud physics, we observe a systematic convergence of GMRes to machine roundoff error in <35 iterations.

We studied a problem of particle laden turbulence at 512^3 resolution with 2×10^6 particles and obtained scalability data of the proposed methods on parallel machines. Using GMRes with a parallel sparse representation of the matrix, we have been able to speed up the linear solver by a factor of ~ 5.6x (2D domain decomposition compared to 1D domain decomposition on 64 cores). Using the combination of techniques described in this paper, we will be able to scale HDNS computations to petascale computers as they become available. Our results along with the proposed solver can be of value to other research areas where an interacting N-body problem is encountered (e.g. particle interactions in electrophoresis of [19] and magnetic particle interactions by [20]). Furthermore, knowing the nature of the governing kernel (namely the decay of involved forces with distance), the eigenvalue analysis can be extended to such problems.

Finally, our complexity analysis shows that for large problems (large numbers of particles), the proposed GMRes scheme scales well for moderate to large numbers of processors in current tera to petascale computers. Further research is needed toward reducing global communications, as it becomes the bottleneck for very large number of processors. Block Gauss Seidel and GMRes with preconditioner could be re-explored due to lower global communication requirements, respectively.

Acknowledgments

This work was made possible by support from the National Science Foundation through grants OCI-0904534 and ATM-0730766 and by the National Center for Atmospheric Research (NCAR). NCAR is sponsored by the National Science Foundation. Computing resources are provided by National Center for Atmospheric Research through CISL-35751010, CISL-35751014 and CISL-35751015.

Appendix A. Solution methods

In this appendix, we give a complete mathematical description of the methods we use to solve (7). For each method, we provide a detailed description of our implementation, explain the convergence criteria used, and provide a simplified convergence study. A complete convergence study has been developed in the perturbation analysis, Section 5.

A.1. Block Gauss–Seidel method

The first method uses a non-overlapping Block Gauss–Seidel implementation. This is a domain decomposition where in each local domain we applied one Gauss–Seidel iteration and then share the local approximation. We iterate until convergence is achieved. To describe the method, consider the domain decompositions described in Fig. 1 and sub-domain l. Notice that this description is valid for the 1D domain decomposition and the 2D domain decomposition described in Fig. 1. In Eq. (A.1) we provide a sketch of the decomposition in matrix form,

$$\underbrace{\begin{pmatrix} \vdots & \vdots & \vdots \\ \mathbf{A}^{\text{out}}_{l,a} & \mathbf{A}^{\text{in}}_{l} & \mathbf{A}^{\text{out}}_{l,b} \\ \vdots & \vdots & \vdots \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \vdots \\ u_{1,l} \\ v_{1,l} \\ w_{1,l} \\ \vdots \\ u_{m_l,l} \\ v_{m_l,l} \\ w_{m_l,l} \\ \vdots \end{pmatrix}}_{\mathbf{x}} \tag{A.1}$$

where m_l is the number of particles inside sub-domain l, $\mathbf{A}$ and $\mathbf{x}$ are described in Section 3, $\mathbf{A}_l^{\text{in}}$ represents the **square** matrix of the interaction among the particles inside subdomain l, $\mathbf{A}_{l,a}^{\text{out}}$ and $\mathbf{A}_{l,b}^{\text{out}}$ represents the interactions of particles inside of subdomain l with the particles outside subdomain l. Notice that $\mathbf{A}_{l,a}^{\text{out}}$ and $\mathbf{A}_{l,b}^{\text{out}}$ do not consider the interactions among particles outside subdomain l.

If we decompose $\mathbf{A}_l^{\text{in}}$ into $L_l^{\text{in}}, D_l^{\text{in}}$ and U_l^{in}. Where L_l^{in} is the lower triangular part of $\mathbf{A}_l^{\text{in}}$ below the diagonal, D_l^{in} be the diagonal of $\mathbf{A}_l^{\text{in}}$ and U_l^{in} is the upper triangular part of $\mathbf{A}_l^{\text{in}}$ above the diagonal. Such that $\mathbf{A}_l^{\text{in}} = L_l^{\text{in}} + D_l^{\text{in}} + U_l^{\text{in}}$. Thus, the iteration equation for $\mathbf{x}_l = \langle u_{1,l}, v_{1,l}, w_{1,l}, \ldots, u_{m_l,l}, v_{m_l,l}, w_{m_l,l}\rangle^T$ is as follows,

$$\mathbf{x}_l^{(n+1)} = \left(D_l^{\text{in}} + U_l^{\text{in}}\right)^{-1}\left(\widetilde{\mathbf{b}}_l - \left[\mathbf{A}_{l,a}^{\text{out}}\middle|L_l^{\text{in}}\middle|\mathbf{A}_{l,b}^{\text{out}}\right]\mathbf{x}^{(n)}\right) \tag{A.2}$$

where $\widetilde{\mathbf{b}}_l$ is the right hand side related to the subdomain l. The global approximation of the solution $\mathbf{x}^{(n+1)}$ for the $n+1$ iteration is as follows,

$$\mathbf{x}^{(n+1)} = \begin{pmatrix} \mathbf{x}_1^{(n+1)} \\ \vdots \\ \mathbf{x}_l^{(n+1)} \\ \vdots \\ \mathbf{x}_{\mathcal{L}}^{(n+1)} \end{pmatrix} \tag{A.3}$$

where $\mathcal{L}$ is the number of subdomains and (n) represents the iteration counter. Notice that the computation of $\left(D_l^{\text{in}} + U_l^{\text{in}}\right)^{-1}$ is not expensive since it is a sparse upper triangular linear system of equation where backward substitution is applied. It is important to point out that (A.2) is highly parallel and it only requires to communicate $\mathbf{x}^{(n)}$ between iterations. Actually, it only requires a local communication between neighboring subdomains, see Fig. 1, because of the truncation radius.

The convergence criteria used for this method was originally proposed in [1] and is included here for completeness. This is defined as follows

$$\text{normalized error} = \frac{\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\|_\infty}{u_{charac}} \leqslant \varepsilon \tag{A.4}$$

where $\mathbf{x}^{(n)} = \langle u_1^{(n)}, v_1^{(n)}, w_1^{(n)}, \ldots, u_{N_p}^{(n)}, v_{N_p}^{(n)}, w_{N_p}^{(n)}\rangle^T$ and u_{charac} is a characteristic velocity. For a bidisperse system with two different radii of droplets, the characteristic velocity u_{charac} can be defined to be the differential terminal velocity ($u_{charac} = |v_{p_2} - v_{p_1}|$).

To study the global convergence properties of this method, we rewrite it as a global (non local) iterative procedure, see [39], leading us to the following,

$$\mathbf{x}^{(n+1)} = \sum_{l=1}^{\mathcal{L}} R_l^T\left(D_l^{\text{in}} + U_l^{\text{in}}\right)^{-1}\left(R_l\widetilde{\mathbf{b}} - \left[\mathbf{A}_{l,a}^{\text{out}}\middle|L_l^{\text{in}}\middle|\mathbf{A}_{l,b}^{\text{out}}\right]\mathbf{x}^{(n)}\right) \tag{A.5}$$

where R_l is a restriction from $\mathbb{R}^{3N_p}$ to the lower dimensional vector which entries are related to the subdomain l. Thus, if the error is defined as $e_n = \mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}$, we conclude that it has the following behavior,

$$\left(\mathbf{x}^{(n+2)} - \mathbf{x}^{(n+1)}\right) = \left(-\sum_{l=1}^{\mathcal{L}} R_l^T\left(D_l^{\text{in}} + U_l^{\text{in}}\right)^{-1}\left[\mathbf{A}_{l,a}^{\text{out}}\middle|L_l^{\text{in}}\middle|\mathbf{A}_{l,b}^{\text{out}}\right]\right)\left(\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\right) \tag{A.6}$$

Therefore, the convergence properties of the Block Gauss–Seidel method are reduced to studying the eigenvalues of the matrix,

$$\mathbf{E} = \sum_{l=1}^{\mathcal{L}} R_l^T\left(D_l^{\text{in}} + U_l^{\text{in}}\right)^{-1}\left[\mathbf{A}_{l,a}^{\text{out}}\middle|L_l^{\text{in}}\middle|\mathbf{A}_{l,b}^{\text{out}}\right] \tag{A.7}$$

Finally, by the fixed point theorem, the method will converge as long as the magnitude of all the eigenvalues of $\mathbf{E} \in \mathbb{R}^{3N_p \times 3N_p}$ are less than one, i.e. $|\lambda(E)| < 1$, where $\lambda(E)$ is the spectrum of E.

A.2. GMRes

We also implemented the Generalized Minimal Residual method (GMRes), see [6,7,4,8]. This method which can handle asymmetric and indefinite matrices and so is well suited for our linear system. This method has been studied and modified by several researchers [40,41,11,42]. Such modifications include the use of static/dynamic preconditioners, recycling, fix number of basis used in the Arnoldi iteration; see [4]. However not all such of modifications are suitable for our linear system. For instance, we tried the recycling of the Krylov subspace basis for our sequence of linear system and did not provide any advantage, see Appendix B.1.

GMRes uses a projection into the Krylov subspace of the matrix $\mathbf{A}$ as a basis for the solution $\mathbf{x}$ of the linear system $\mathbf{Ax} = \mathbf{b}$. The Krylov subspace $K_n(\mathbf{A}, \mathbf{b}) = \langle \mathbf{b}, \mathbf{Ab}, \mathbf{A}^2\mathbf{b}, \ldots, \mathbf{A}^{n-1}\mathbf{b} \rangle$ is orthonormalized by the Arnoldi algorithm and then the problem is reduced to a least square problem to minimize the residual $\|\mathbf{Ax}_n - \mathbf{b}\|$. GMRes usually provides a very good approximation with only a small number of basis/iterations of the Krylov subspace for well-conditioned linear systems, but extreme cases can also be found [36].

For the GMRes implementation, in Algorithm 1, we do not require a specific form of the matrix $\mathbf{A}$, see [6,7,4,40]. We only require an efficient matrix–vector multiplication between $\mathbf{A}$ and a given vector. This will be denoted as $\mathbf{Av}$, where $\mathbf{v}$ is going be specified on each iteration of GMRes. For this particular implementation of GMRes, we do not include restarts because, as shown in Sections 5 and 6, GMRes converges in few iterations. However, precautions must be taken into account because we found conditions when GMRes may stagnate for our linear system, see Section 5.

Algorithm 1. GMRes

1: $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ ← $\mathbf{Ax}_0$ **req. Near-Field comm.**
2: $\beta_0 = \|\mathbf{b}\|, \beta_1 = \|\mathbf{r}_0\|, \mathbf{q}_1 = \mathbf{r}_0/\beta_1$ ← β_0 **and** β_1 **req. Global comm.**
3: **for** $n = 1, 2, 3, \ldots$ **do**
4: $\quad \mathbf{w} = \mathbf{Aq}_n$ ← $\mathbf{Aq}_n$ **req. Near-Field comm.**
5: $\quad$ **for** $j = 1 \to n$ **do**
6: $\quad\quad h_{jn} = \mathbf{q}_j^* \mathbf{w}$
7: $\quad\quad \mathbf{w} = \mathbf{w} - h_{jn}\mathbf{q}_j$ ← h_{jn} **req. Global comm.**
8: $\quad$ **end for**
9: $\quad h_{n+1,n} = \|\mathbf{w}\|$ ← $\|\mathbf{w}\|$ **req. Global comm.**
10: $\quad \mathbf{q}_{n+1} = \mathbf{w}/h_{n+1,n}$
11: $\quad$ Find y_n to minimize $\|\widetilde{H}_n y_n - \beta_1 e_1\| (= \|\mathbf{r}_n\|)$
12: $\quad$ **if** $\|\mathbf{r}_n\|/\beta_0 < \varepsilon$ **then**
13: $\quad\quad \mathbf{x} = \mathbf{x}_0 + Q_n y_n$ ← $Q_n y_n$ **req. Near-Field Comm.**
14: $\quad\quad$ **BREAK** main loop
15: $\quad$ **end if**
16: **end for**

Now we proceed to the explanation of Algorithm 1 line by line. For steps which require data communication, please see Section 4.

Lines 1–2 Initialization with initial guess $\mathbf{x}_0$ and definition of β_0, β_1 and $\mathbf{q}_1$ (first orthonormal basis for K_n).
Lines 3–16 Main loop to build and orthonormalize the Krylov subspace K_n.
Lines 4 Matrix–vector multiplication to compute next Krylov basis.
Lines 5–8 Arnoldi iteration to orthogonalize vector $\mathbf{w}$ with respect to $\mathbf{q}_j$, $j = 1, \ldots, n$. Notice that the Arnoldi method is just a modified Gram–Schmidt orthonormalization, and it can be modified for stability purposes, see [4].
Lines 9–10 Normalization and storage of $\mathbf{w}$ as $\mathbf{q}_{n+1}$.
Line 11 Least square problem to find y_n, where e_1 is a vector of dimension $(n+1)$ defined as $e_1 = \langle 1, 0, \ldots, \rangle^T$; and $\widetilde{H}_n$ is an upper Hessenberg matrix of size $(n+1) \times (n)$. This least square problem can be solved by using a QR decomposition of $\widetilde{H}_n$ with Givens rotations.
Line 12 Checking if relative residual is smaller than threshold ε. Notice here we are dividing by β_0 not β_1, the reason for this is that we want a small residual for the original problem ($\mathbf{Ax} = \mathbf{b}$) we are solving not the modified one ($\mathbf{Az} = \mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, with $\mathbf{x} = \mathbf{z} + \mathbf{x}_0$) modified by the initial guess $\mathbf{x}_0$.
Lines 13–14 Final construction of the *approximated solution* $\mathbf{x}$ from the initial guess and the Krylov subspace $Q_n = [\mathbf{q}_1 | \ldots | \mathbf{q}_n]$.

The convergence criteria for GMRes is the relative residual,

$$\frac{\|\mathbf{Ax}_n - \mathbf{b}\|}{\|\mathbf{b}\|} \leqslant \varepsilon. \tag{A.8}$$

thus, we stop GMRes as soon as the relative residual is below ε, as was already mentioned in Algorithm 1, line 12. Notice that the convergence criterion (A.8) is different from (A.4). However (A.8) has been used for comparison purposes.

The convergence analysis for GMRes has been studied by several authors, see [34,37,35,36,4,5]. A complete analysis of the convergence of GMRes for this type of matrix has been performed in Section 5. Convergence is mainly affected by the proximity of the particles, i.e. if the particles cluster, GMRes start to behave poorly.

A.3. GMRes with preconditioner and (optional) storage of preconditioned basis

There are some situations, which have been described in Section 5, where the linear system (1) may show a slow convergence. Up to this point we have not provided any tool to manage this bad behavior. So, it is of high importance to be able to provide such a tool. Thus, the next step will be to provide a preconditioner for (1) to be used with GMRes.

The preconditioner we finally decided to use is a restricted Schwarz type [43], which is adapted to our problem. A similar approach can be found in [44] where it was used as an iterative solver, and in [45]. Notice that we cannot follow the same approach developed in [44] because their method, unfortunately, diverges for our problem. See more about preconditioners in [46].

The preconditioner takes advantage of the domain decomposition. However a modification must be made. This is because the preconditioner will include an overlap region, see Fig. 13. The overlap must be chosen such that it handles the ill-conditioning of the linear system and also minimizes the total computation time. Thus, following the same notation used in Appendix A.1, we obtain the following representation for the preconditioner,

$$M^{-1}\widetilde{\mathbf{x}} = \sum_l \widehat{R}_l^T P_l \widehat{\mathbf{A}}_l^{-1} \widehat{R}_l \widetilde{\mathbf{x}} \tag{A.9}$$

where $\widehat{\mathbf{A}}_l$ is the matrix of interaction among all the particles inside subdomain l together with the particles in the overlap region, $\widehat{R}_l$ is the restriction from $\mathbb{R}^{3N_p}$ to the lower dimensional vector which entries are related to the subdomain l and its overlap, P_l is the projection from the entries related to the subdomain l and its overlap to *only* the entries related to the subdomain l and $\widehat{R}_l^T$ is the transpose of $\widehat{R}_l$. A graphical description of the overlapped subdomain related to $\widehat{\mathbf{A}}_l$ can be seen in Fig. 13. Notice that we could also have a preconditioner with no overlap. The overlap, of course, makes the preconditioner more computationally expensive. However, we considerably improve the quality of the preconditioner with the overlap. Both preconditioners are considered and compared. For future reference, we define as the *overlapped preconditioner* as the preconditioner considering the overlap region and the *non-overlapped preconditioner* the preconditioner with no overlap.

It is important to point out that this is an additive type preconditioner [39], meaning that the terms on the sum on the right hand side of (A.9) are independent. This implies that it is a highly parallel preconditioner. More details on the parallel implementation can be found in Section 4.

GMRes must be modified to include the preconditioner, see Algorithm 2. However, we will only explain the lines that differ from Algorithm 1.

Algorithm 2. GMRes with preconditioner and (optional) storage of preconditioned basis

1: $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0, \beta_0 = \|\mathbf{b}\|, \beta_1 = \|\mathbf{r}_0\|$
2: $\mathbf{q}_1 = \mathbf{r}_0/\beta_1$
3: **for** $n = 1, 2, 3, \ldots,$ **do**
4: $\quad \mathbf{z}_n = \mathbf{M}^{-1}\mathbf{q}_n$ ← **NEW LINE**
5: $\quad \mathbf{w} = \mathbf{A}\mathbf{z}_n$ ← **NEW LINE**
6: $\quad$ **for** $j = 1 \to n$ **do**
7: $\quad\quad h_{jn} = \mathbf{q}_j^* \mathbf{w}$
8: $\quad\quad \mathbf{w} = \mathbf{w} - h_{jn}\mathbf{q}_j$
9: $\quad$ **end for**
10: $\quad h_{n+1,n} = \|\mathbf{w}\|$
11: $\quad \mathbf{q}_{n+1} = \mathbf{w}/h_{n+1,n}$
12: $\quad$ Find y_n to minimize $\|\widetilde{H}_n y_n - \beta_1 e_1\| (= \|\mathbf{r}_n\|)$
13: $\quad$ **if** $\|\mathbf{r}_n\|/\beta_0 < \varepsilon$ **then**
14: $\quad\quad \mathbf{x} = \mathbf{x}_0 + Z_n y_n$ ← **NEW LINE**
15: $\quad\quad$ **BREAK** main loop
16: $\quad$ **end if**
17: **end for**

Lines 4–5 Use of right-preconditioner and storage of right preconditioned vectors (line 4); and matrix–vector multiplication to compute next Krylov basis (line 5). Line 4 can be modified to have an iteration dependent preconditioner as $\mathbf{z}_n = \mathbf{M}_n^{-1}\mathbf{q}_n$, see more details in [40,4].

Lines 14 Final construction of *approximated solution* $\mathbf{x}$ from initial guess and the Krylov subspace Z_n, where $Z_n = [\mathbf{z}_1 | \ldots | \mathbf{z}_n]$. Line 14 must be modified if the vectors $\mathbf{z}_n$ are not stored and the right preconditioner is in use, this should change to $\mathbf{x} = \mathbf{x}_0 + \mathbf{M}^{-1} Q_n y_n$, where $Q_n = [\mathbf{q}_1 | \ldots | \mathbf{q}_n]$

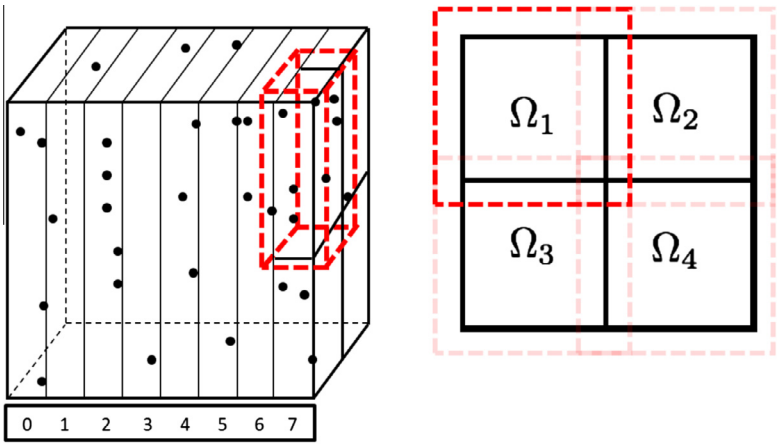


Fig. 13. Restricted Schwarz domain decomposition preconditioner in 1D domain decomposition. Left: Dashed box is the preconditioning volume for Ω_1. Right: Side view of the computational domain for 4 preconditioning volumes.

Appendix B. Other attempts to solve the governing system

We investigated several other solution methods, and we discuss two of them in this appendix. For our problem these methods turned out to be computationally expensive, but we mention them because they are suitable for this problem and they are mathematical attractive. The first approach is making use of recycling (Appendix B.1) and the second method is a black-box type preconditioner by means of a contour integral (Appendix B.2).

B.1. Sequence of linear systems awareness

This approach considers, in its design, that the solver is aware that it is solving a sequence of linear systems [41,47–49] and takes advantage of it. Namely, if we have $\mathbf{A}_{(n)}\mathbf{x}_{(n)} = \mathbf{b}_{(n)}$ and assuming that the change from $\mathbf{A}_{(n)}$ and $\mathbf{b}_{(n)}$ to $\mathbf{A}_{(n-1)}$ and $\mathbf{b}_{(n-1)}$ is small, it is then expected that $\mathbf{x}_{(n)}$ is close to $\mathbf{x}_{(n-1)}$ and the procedure used in solving $\mathbf{A}_{(n-1)}\mathbf{x}_{(n-1)} = \mathbf{b}_{(n-1)}$ can provide certain advantage compared to just solving $\mathbf{A}_{(n)}\mathbf{x}_{(n)} = \mathbf{b}_{(n)}$ from scratch.

For our problem we found that the overhead added in this *awareness*/*recycling* of Krylov subspace [41] is quite high to the point that renders the method to be computationally slower than just solving each linear system independently. We tried this method using the computer code provided by one of the authors of [41] on his web page. Our conclusion is that our linear system was too well-conditioned to be able to take advantage of the sequence of linear systems. For the problem discussed in [41] the matrix is more ill-conditioned than our matrix. We believe that recycling is a good idea to handle certain sequence of linear systems, however, we are left with the following open question: How do we identify a sequence of linear system that would be a good candidate for this kind of awareness?.

B.2. The contour preconditioner

We experimented with reusing the already implemented matrix–vector multiplication **Ax**. The first preconditioner we explored was a shifted version of the original matrix $\mathbf{M} = \alpha\mathbf{I} + \mathbf{A}$. This approach has already been considered in literature but from a different point of view [50,4] in Section 10.8.2. Our point of view was to use it directly as a nested GMRes method. This means use GMRes and also use GMRes for the preconditioner. The reason for doing this can be seen directly from Eq. (10). In other words, when we add the shift we are basically reducing the factor ρ_M for **M** respect to the original factor ρ for the matrix **A**. This implies that GMRes should converge faster for the shifted matrix **M**. The factor ρ_M is the following,

$$\rho_M(\alpha) = \frac{\lambda_M - \lambda_m + 2\delta}{\lambda_M + \lambda_m + 2\alpha + 2\sqrt{(\lambda_M + \alpha)(\lambda_m + \alpha) + \delta^2}} \tag{B.1}$$

ρ_M is definitely smaller than ρ in (10), i.e. $\rho_M(\alpha) \leqslant \rho_M(0) = \rho$ for $\alpha \geqslant 0$. Notice that this implies a better performance of GMRes for **M**. Unfortunately, this approach does not speed up the global computation. This makes it worse since each use of the preconditioner requires several matrix–vector calls. Notice however that it does reduce the global iteration number.

One would think that having a large α would help the convergence of the preconditioner, which is true. But, on the other hand, having a large α implies that **M** is not a very good preconditioner. A way to see this is that $\mathbf{M}^{-1}$ should be as close as possible to $\mathbf{A}^{-1}$. But having a large α goes against this principle and having a too small α is similar to coming back to the original problem $\mathbf{Ax} = \mathbf{b}$. Thus, we need a preconditioner as close as possible to $\mathbf{A}^{-1}$ and having an α as large as possible.

To do this, we considered that $\mathbf{M}^{-1}$ is not other than just applying the function $f(x) = 1/x$ to $\mathbf{M}$, which is equivalent to say $f(\mathbf{M})$. We could obtain a new function by approximating $1/x$ with a truncated Laurent series. For a positive definite matrix, we get the following,

$$\begin{aligned}\frac{1}{x} &= \frac{1}{x+\alpha-\alpha}\\ &= \frac{1}{z-\alpha}(\text{Changing variable} z = x+\alpha)\\ &= \frac{1}{z}\frac{1}{1-\alpha/z} = \frac{1}{z}\sum_{k=0}^{\infty}\left(\frac{\alpha}{z}\right)^k(\text{Considering}\left|\frac{\alpha}{z}\right| < 1)\\ &\approx \frac{1}{z}\sum_{k=0}^{N_f}\left(\frac{\alpha}{z}\right)^k(\text{Truncating after} N_f+1 \text{ terms})\\ &= \frac{1-\left(\frac{\alpha}{z}\right)^{N_f+1}}{z-\alpha}\end{aligned} \tag{B.2}$$

Notice that for this derivation it is very important to have $\left|\frac{\alpha}{z}\right| < 1$, otherwise this derivation does not make sense. For our case, it does make sense because all the eigenvalues seems to have a positive real part. Thus, the proposed function is,

$$f(z) = \frac{1-\left(\frac{\alpha}{z}\right)^{N_f+1}}{z-\alpha}. \tag{B.3}$$

Notice that N_f should be defined such that $1/x$ is well approximated by $f(z)$ for all the eigenvalues of $\mathbf{M}$. When f is an analytic function, $f(\mathbf{M})$ can be computed by a contour integral [51–55],

$$f(\mathbf{M}) = \frac{1}{2\pi i}\int_{\Gamma} f(z)(z\mathbf{I}-\mathbf{M})^{-1}dz \tag{B.4}$$

where $i = \sqrt{-1}$ and Γ is a contour that encloses all the spectrum of $\mathbf{M}$. Notice that we actually do not need to compute the matrix $f(\mathbf{M})$, we only need to have $f(\mathbf{M})\mathbf{v}$, where $\mathbf{v}$ is a given vector, thus,

$$f(\mathbf{M})\mathbf{v} = \frac{1}{2\pi i}\int_{\Gamma} f(z)(z\mathbf{I}-\mathbf{M})^{-1}\mathbf{v}dz \tag{B.5}$$

where the contour integral can be approximated by the trapezoidal rule, see [55,54] for a complete discussion. Thus, the approximation is as follows,

$$f(\mathbf{M})\mathbf{v} \approx \frac{1}{2\pi i}\sum_{k}^{N_t} f(z_k)(z_k\mathbf{I}-\mathbf{M})^{-1}\mathbf{v}\Delta_k \tag{B.6}$$

Therefore, the problem is reduced to solving $(z_k\mathbf{I}-\mathbf{M})\mathbf{h} = \mathbf{v}$, where $\mathbf{h}$ is a given vector defined by $\mathbf{h} = (z_k\mathbf{I}-\mathbf{M})^{-1}\mathbf{v}$ and $k = 1,\ldots,N_t$. We propose to handle these linear systems with GMRes since we are using the shifted matrix $\mathbf{M} = \alpha\mathbf{I}+\mathbf{A}$, which has a shifted eigenvalue distribution compared to $\mathbf{A}$. The linear systems $(z_k\mathbf{I}-\mathbf{M})\mathbf{h} = \mathbf{v}$ can be handle in parallel by only shifting the reduced Hessenberg matrix of $\mathbf{M}$. It will only require a highly parallel solution of N_t small least square problems [49,56,57].

From our experiments, we concluded that the overhead added is high. However, we found that the preconditioner reduces the number of iterations significantly. Another way to see this preconditioner is that it is a *black-box* type preconditioner. It could be applied to any positive definite matrix and reuses the matrix–vector multiplication. It could also be used with matrices having all eigenvalues with positive real part and small imaginary part [54]. For matrices that do not satisfy these constraints a new $f(z)$ and/or a different contour must be defined. Another option would be to use the normal equations, which ensures a positive definite matrix for a non-singular matrix. This might be a double edged sword since the normal equations square the condition number.

References

[1] O. Ayala, W. Grabowski, L.-P. Wang, A hybrid approach for simulating turbulent collisions of hydrodynamically-interacting particles, J. Comp. Phys. 225 (2007) 51–73.
[2] L.-P. Wang, O. Ayala, S. Kasprzak, W. Grabowski, Theoretical formulation of collision rate and collision efficiency of hydrodynamically-interacting cloud droplets in turbulent atmosphere, J. Atmos. Sci. 62 (2005) 2433–2450.
[3] L.-P. Wang, O. Ayala, W.W. Grabowski, Effects of aerodynamic interactions on the motion of heavy particles in a bidisperse suspension, J. Turbul. 8 (2007).
[4] Y. Saad, Iterative methods for sparse linear systems, second ed., SIAM, 2003.
[5] V. Simoncini, D. Szyld, On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods, SIAM Review 47 (2005) 247–272.
[6] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856.
[7] L.N. Trefethen, D. Bau III, Numerical Linear Algebra, SIAM, 1997.
[8] H.A. van der Vost, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, 2003.

[9] G. Pashos, M.E. Kavousanakis, A.N. Spyropoulos, J.a. Palyvos, A.G. Boudouvis, Simultaneous solution of large-scale linear systems and eigenvalue problems with a parallel GMRES method, J. Comput. Appl. Math. 227 (2009) 196–205.
[10] H. He, G. Bergere, S. Petiton, A hybrid GMRES/LS-Arnoldi method to accelerate the parallel solution of linear systems, Comput. Math. Appl. 51 (2006) 1647–1662.
[11] J. Erhel, A parallel GMRES version for general sparse matrices, Electronic Trans. Numer. Anal. 3 (1995) 160–176.
[12] I.S. Duff, H.A. van Der Vorst, Developments and trends in the parallel solution of linear systems, Parallel Comput. 25 (1999) 1931–1970.
[13] K. Dekker, Parallel GMRES and domain decomposition, Technical Report, 2000.
[14] D.C. Allison, M. Sosonkina, L.T. Watson, Scalability analysis of parallel GMRES implementations, Parallel Alg. Appl. 17 (2002) 263–284.
[15] E. de Sturler, H. van Der Vorst, Reducing the effect of global communication in GMRES() and CG on parallel distributed memory computers, Appl. Numer. Math. 18 (1995) 441–459.
[16] W.-H. Yang, G. Schatz, R. Van Duyne, Discrete dipole approximation for calculating extinction and Raman intensities for small particles with arbitrary shapes, J. Chem. Phys. 103 (1995) 869–875.
[17] T. Jensen, L. Kelly, A. Lazarides, G. Schatz, Electrodynamics of noble metal nanoparticles and nanoparticles clusters, J. Cluster Sci. 10 (1999) 295–317.
[18] H. Keh, J. Chen, Particle interactions in electrophoresis, J. Colloid Interf. Sci. 158 (1993) 199–222.
[19] A. Filippov, Phoretic motion of arbitrary cluster of N spheres, J. Colloid Interf. Sci. 241 (2001) 479–491.
[20] B. Yellen, G. Friedman, A. Feinerman, Analysis of interactions for magnetic particles assembling on magnetic templates, J. Appl. Phys. 91 (2002) 8552–8554.
[21] J. Alward, W. Imaino, Magnetic forces on monocomponent toner, IEEE Trans. Magnet. 22 (1986) 128–134.
[22] R. Caflish, J. Luke, Variance in the sedimentation speed of a suspension, Phys. Fluids 28 (1985) 759–760.
[23] D. Koch, E. Shaqfeh, Screening in sedimenting suspensions, J. Fluid Mech. 224 (1991) 275–303.
[24] B. Felderhof, Hydrodynamic screening and axisymmetric turbulence in the transient sedimentation of a dilute suspension at small Reynolds number, Physica A 387 (2008) 5999–6012.
[25] X. Yin, D.L. Koch, Velocity fluctuations and hydrodynamic diffusion in finite-Reynolds-number sedimenting suspensions, Phys. Fluids 20 (2008) 043305.
[26] L.-P. Wang, B. Rosa, H. Gao, H. Guowei, G. Jin, Turbulent collision of inertial particles: Point-particle based hybrid simulations and beyond, Int. J. Multiphase Flow 35 (2009) 854–867.
[27] L.-P. Wang, O. Ayala, W. Grabowski, On improved formulations of the superposition method, J. Atmosp. Sci. 62 (2005) 1255–1266.
[28] M. Allen, D. Tildesley, Computer Simulation of Liquids, Oxford University Press, 1987.
[29] T.A. Davis, Direct Methods for Sparse Linear Systems, SIAM, 2006.
[30] B. Rosa, L.-P. Wang, Parallel implementation of particle tracking and collision in a turbulent flow, Lect. Notes Comput. Sci. 6068 (2009) 388–397.
[31] P. Dmitruk, L.-P. Wang, W. Matthaeus, R. Zhang, D. Seckel, Scalable parallel FFT for spectral simulations on a Beowulf cluster, Parallel Comput. 27 (2001) 1921–1936.
[32] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences, 6, World Scientific Publishers, 2007.
[33] H. Wendland, Scattered Data Approximation, Cambridge University Press, 2005.
[34] B. Fischer, Polynomial Based Iteration Methods of Symmetric Linear Systems, John Wiley and Sons, Inc, 1996.
[35] T.A. Driscoll, K.-C. Toh, L.N. Trefethen, From potential theory to matrix iterations in six steps, SIAM Rev. 40 (1998) 547.
[36] A. Greenbaum, V. Ptak, Z. Strakos, Any nonincreasing convergence curve is possible for GMRES, SIAM J. Matrix Anal. Appl. 17 (1996) 465–469.
[37] N.M. Nachtigal, S.C. Reddy, L.N. Trefethen, How fast are nonsymmetric matrix iterations?, SIAM J Matrix Anal. Appl. 13 (1992) 778.
[38] O. Ayala, L.-P. Wang, Parallel implementation and scalability analysis of 3D fast Fourier transform using 2D domain decomposition, Parallel Comput. 39 (2013) 58–77.
[39] B.F. Smith, P.E. Bjørstad, W.D. Gropp, Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004.
[40] Y. Saad, A flexible inner-outer preconditioned GMRes algorithm, SIAM J. Sci. Comput. 14 (1993) 461–469.
[41] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J. Sci. Comput. 28 (2006) 1651.
[42] Z. Bai, D. Hu, L. Reichel, A Newton basis GMRES implementation, IMA J. Numer. (1994) 563–581.
[43] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (1999) 792.
[44] C.E. Torres, L. Barba, Fast radial basis function interpolation with Gaussians by localization and iteration, J. Comput. Phys. 228 (2009) 4976–4999.
[45] R. Yokota, L. Barba, M.G. Knepley, PetRBF – A parallel O(N) algorithm for radial basis function interpolation with Gaussians, Comput. Methods Appl. Mech. Eng. 199 (2010) 1793–1804.
[46] K. Chen, Matrix Preconditioning Techniques and Applications, Cambridge University Press, 2005.
[47] J. Tebbens, M. Tuma, Efficient preconditioning of sequences of nonsymmetric linear systems, SIAM J. Sci. Comput 29 (2007) 1918–1941.
[48] L. Giraud, S. Gratton, E. Martin, Incremental spectral preconditioners for sequences of linear systems, Appl. Numer. Math. 57 (2007) 1164–1180.
[49] D. Darnell, R. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, Linear Algebra Appl. 429 (2008) 2415–2434.
[50] T.a. Manteuffel, An incomplete factorization technique for positive definite linear systems, Math. Comput. 34 (1980) 473.
[51] T. Schmelzer, L. N. Trefethen, Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals, ETNA, vol. 29, 2007, Technical Report, 1969.
[52] P. Davies, N. Higham, Computing f(A) b for matrix functions *f*, in: A. Bori∼i, A. Frommer, B. Joó, A. Kennedy, B. Pendleton (Eds.), QCD and Numerical Analysis III, vol. 47, Springer Berlin Heidelberg, pp. 15–24.
[53] L.N. Trefethen, J.a.C. Weideman, T. Schmelzer, Talbot quadratures and rational approximations, BIT Numer. Math. 46 (2006) 653–670.
[54] N. Hale, N. Higham, L.N. Trefethen, Computing Aα, log (A), and related matrix functions by contour integrals, SIAM J. Numer. Anal. 46 (2008) 2505–2523.
[55] N.J. Higham, Function of Matrices, Theory and Computation, SIAM, 2008.
[56] A. Frommer, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput. 19 (1998) 15–26.
[57] S. Olver, Shifted GMRES for oscillatory integrals, Numer. Math. 114 (2009) 607–628.
[58] S. Aluru, J. Gustafson, G.M. Prabhu, F.E. Sevilgen, Distribution-independent hierarchical algorithms for the N-body problem, J. Supercomput. 12 (1998) 303–323.