Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 450980, 12 pages
http://dx.doi.org/10.1155/2015/450980

Research Article

Parallel N-Body Simulation Based on the PM and P3M Methods Using Multigrid Schemes in conjunction with Generic Approximate Sparse Inverses

P. E. Kyziropoulos, C. K. Filelis-Papadopoulos, and G. A. Gravvanis

Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, Kimmeria, 67100 Xanthi, Greece

Correspondence should be addressed to P. E. Kyziropoulos; pkyzirop@ee.duth.gr

Received 27 November 2014; Revised 15 March 2015; Accepted 18 March 2015

Academic Editor: Yuming Qin

During the last decades, Multigrid methods have been extensively used for solving large sparse linear systems. Considering their efficiency and the convergence behavior, Multigrid methods are used in many scientific fields as solvers or preconditioners. Herewith, we propose two hybrid parallel algorithms for N-Body simulations using the Particle Mesh method and the Particle Particle Particle Mesh method, respectively, based on the V-Cycle Multigrid method in conjunction with Generic Approximate Sparse Inverses. The N-Body problem resides in a three-dimensional torus space, and the bodies are subject only to gravitational forces. In each time step of the above methods, a large sparse linear system is solved to compute the gravity potential at each nodal point in order to interpolate the solution to each body. Then the Velocity Verlet method is used to compute the new position and velocity from the acceleration of each respective body. Moreover, a parallel Multigrid algorithm, with a truncated approach in the levels computed in parallel, is proposed for solving large linear systems. Furthermore, parallel results are provided indicating the efficiency of the proposed Multigrid N-Body scheme. Theoretical estimates for the complexity of the proposed simulation schemes are provided.

1. Introduction

The simulation of an N-Body system is referred to as a dynamical system of bodies influenced by mainly the gravitational force. The N-Body simulation has become a fundamental tool in the study of complex physical systems; compare [1]. The physical interactions of bodies (e.g., gravitational, Coulomb) and the evolution of the dynamical system present the phase-space density distribution of the system. These systems rely on a substantial number of bodies, thus making the complexity an important factor of the computational procedure. The computation of the gravitational influence of the bodies in a confined space can be made with different approaches. The direct method of solving an N-Body problem, called Particle Particle, is based on the computation of the gravitational force in each pair of bodies within the system (Newtonian gravitation). This direct approach scales the problem with $O(N^2)$ complexity, where N denotes the number of bodies, rendering the scheme restrictive for large number of bodies. A different approach to the problem, called Particle Mesh (PM) method, neglects the close interactions between bodies and takes into account only the dynamics of superparticles, each consisting of a great number of bodies. The particles contribute to their masses to the mass density of the nodal points on the confined space. A more complicated hybrid method, Particle Particle Particle Mesh method (P3M), which is composed of the above two methods, divides the force into two parts: the short range and the long range force. The long range force is computed with the Particle Mesh method, and the short range force is computed via the Particle Particle method leading to more accurate results; compare [1–4].

Additionally, there are other approaches based on tree codes, such as Tree code methods (cf. [3]) and Fast Multipole

methods (cf. [5]). The Tree code methods have $O(N\log(N))$ complexity, but Fast Multipole methods can reach up to $O(N)$ complexity, though "the exact scaling seems implementation dependent and has been debated in the literature" (cf. [6]). Approximate inverses have been used as preconditioners for various iterative schemes combined with fast approximate matrix-vector products such as FMM (cf. [7, 8]). Furthermore, various implementations have been proposed based on Mesh, Tree, and Multipole methods and on hybrid models (cf. [9]).

The N-Body simulations cover a wide area of cosmological tests. The proposed schemes can be used, as an alternative method, for solving the N-Body problem on mainly spherical or disk galaxies simulations; compare [10]. These simulations can be solved using Tree code techniques, with computational complexity of $O(N\log N)$, where N is the number of bodies. Mesh type methods in conjunction with the FFT algorithm for solving the potential have periodical boundary conditions. These boundary conditions when applied to a galaxy replicate the solution on the boundaries of the domain, introducing significant error; compare [10]. The proposed scheme takes into consideration the physical space where Dirichlet boundary conditions can be used, to eliminate such errors. Furthermore, the Multigrid method has computational complexity of $O(n)$, while the FFT algorithm has computational complexity of $O(n\log n)$, where n is the number of grid points.

The gravity potential required by the field force, on both PM and P3M methods, is computed by solving a linear system derived from a Poisson PDE; compare [3]. The potential is commonly computed with a Fast Fourier approach with an $O(n\log n)$ complexity, where n denotes the number of grid points. Some examples with this approach can be found in [11–13]. Due to the nature of the problem given by the Poisson equation, a different approach for solving the linear system is applied using Multigrid methods which possess near optimal $O(n)$ complexity. During the last decades, Multigrid methods have been used in a variety of scientific fields for solving Partial Differential Equations in Computational Fluid Dynamics and Computational Physics due to their near optimal computational complexity and convergence behavior; compare [14–23]. In order to accelerate the convergence of the Multigrid method, the Dynamic Over/Under Relaxation (DOUR) scheme is used (cf. [24]) in conjunction with the Generic Approximate Sparse Inverse (GenAspI) matrix (cf. [25]).

Approximate inverses have been used as preconditioners for various iterative schemes due to their inherent parallelism and convergence behavior; compare [15, 26–30]. Furthermore, approximate inverses have been used effectively in conjunction with the Multigrid method for a variety of problems; compare [16, 17, 31, 32].

Herewith, we propose a new hybrid parallel algorithm for computing the gravity potential using the Particle Mesh and the P3M methods and the Multigrid V-Cycle method in conjunction with the GenAspI (GenAspI-MGV) method to accelerate the solution of the linear system. The Particle Mesh method computes the mass density in parallel by adding each mass influence according to weights computed from the distance of each node to the grid points surrounding each respective body. The interpolation of the respective weights in each nodal point is performed using the Cloud in Cell (CIC) method; compare [1]. Then the gravity potential at the cell centers is computed by solving the linear system using the parallel GenAspI-MGV method with a new approach where the lower order levels are not being parallelized in order to avoid overhead in levels with low computational cost. In the P3M method the short range forces are computed with Newton's gravity formula in parallel for each short range force pair. Additionally, by integrating the equations of motion through the Velocity Verlet integrator (cf. [33]), the new position of each body is computed. The low computational complexity of the Multigrid method utilized in each time step for the solution of the linear system renders the algorithm efficient, especially for increasing numbers of bodies.

The parallelization of the proposed algorithms is utilized on a hybrid parallel system with the parallel tools OpenMP and MPI. The hybrid parallel system is preferred due to the nature of the problem that requires a lot of data to be sent. The implementation sends the N-Body data to every cluster node and then the data is further parallelized with OpenMP on each computational node. The linear system is solved on the head node, due to the fact that the order of the linear system is much smaller than the number of bodies and requires less computational effort.

OpenMP is a collection of directives available for a variety of languages including C/C++/Fortran used to parallelize programs for Symmetric Multiprocessing Units. On the other hand, MPI (Message Passing Interface) is a message-passing system used to parallelize the algorithm on a cluster level.

Finally, numerical results on the performance and convergence behavior of the proposed N-Body GenAspI-MGV algorithms are presented for solving three-dimensional N-Body simulation problems on a hybrid system and on Symmetric Multiprocessing Units for both PM and P3M methods.

2. Parallel Multigrid Method in conjunction with the Generic Approximate Sparse Inverse

Multigrid is a computational method for solving linear systems of equations which is used extensively in Computational Physics due to its near-optimal complexity and its convergence behavior; compare [14–16, 18, 21–23, 31, 32, 34]. Stationary iterative methods are not effective on damping the low-frequency components of the error. On the other hand, they are highly efficient on damping the high frequency components of the error within the first few iterations; compare [14, 15, 22]. Multigrid methods exploit the way stationary iterative schemes handle the error by creating a hierarchy of (finer and coarser) grids, as presented in Figure 1, transferring the linear system to various levels with different mesh size.

The Multigrid methods can be analyzed in the distinct components: smoother, transfer operators, and cycle strategy.

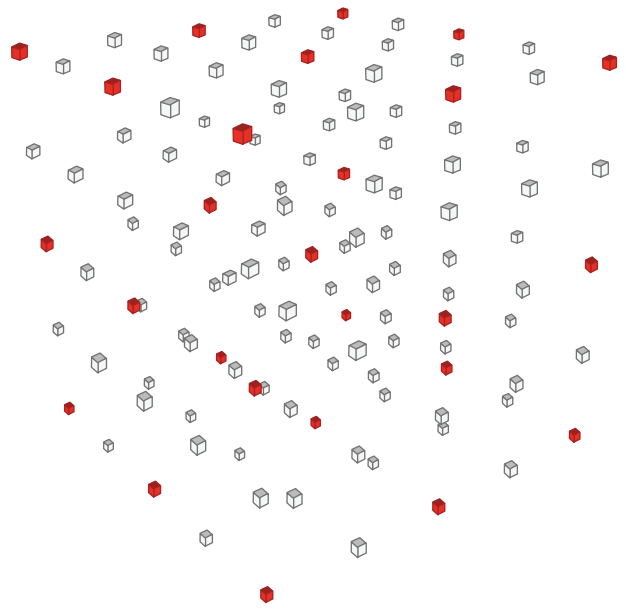


Figure 1: Multigrid level hierarchy with mesh size $h = 1/2$ (red points) and $h = 1/4$ (red and white points).

The smoothers are used to obtain the respective corrections for the solution of the linear system on each level and can be described by the following relation (cf. [15, 16, 22, 34]):

$$x_\ell^{(k+1)} = x_\ell^{(k)} + \omega M_\ell \left(b_\ell - A_\ell x_\ell^{(k)}\right), \quad k = 0, 1, 2, 3, \ldots, \tag{1}$$

where ℓ denotes the level in which the smoother is applied, M_ℓ is the preconditioner to Richardson's iterative method, and ω is the damping parameter ($0 < \omega < 2$). The Generic Approximate Sparse Inverse is used as a preconditioner to the Richardson method. By increasing the $l$fill values the approximate inverse tends to the exact inverse of the matrix A. The algorithm for the computation of an approximate inverse sparsity pattern is given in [25, 35, 36]. The GenAspI matrix $M_{\text{drptol}}^{l\text{fill}}$ is then computed, according to this sparsity pattern, by solving recursively the following system (cf. [25, 36]):

$$UM_{\text{drptol}}^{l\text{fill}} = I, \qquad M_{\text{drptol}}^{l\text{fill}} L = 0, \tag{2}$$

where L and U are the lower and upper triangular factors computed by the $ILU(0)$ factorization. The computation of the elements of the GenAspI matrix is given by

$$\mu_{ij} = \begin{cases} \dfrac{\left(1 - \sum_{k=i+1}^{n} u_{ik}\mu_{ki}\right)}{u_{ii}}, & i = j, \\ -\displaystyle\sum_{k=j+1}^{n} \mu_{ik} l_{kj}, & i < j, \\ \dfrac{\left(-\sum_{k=i+1}^{n} u_{ik}\mu_{kj}\right)}{u_{ii}}, & i > j, \end{cases} \tag{3}$$

and the GenAspI algorithm has been presented in [25].

The computational complexity of the GenAspI algorithm (cf. [25]) in terms of its nonzero elements is as follows:

Multiplications

$$\approx \left(\frac{3}{4}\right)\left(\frac{nnz\,(M)^2}{n}\right) - \left(\frac{3}{2}\right)(nnz\,(M)) + \left(\frac{5}{8}\right)n,$$

Additions

$$\approx \left(\frac{3}{4}\right)\left(\frac{nnz\,(M)^2}{n}\right) - \left(\frac{3}{2}\right)(nnz\,(M)) + \left(\frac{3}{4}\right)n. \tag{4}$$

In order for a smoother to be effective the smoothing property must be satisfied (cf. [15, 18, 20–22, 34]). The smoothing property has been proven in [31] for the Optimized Banded Generalized Approximate Inverse (OBGAIM) matrix (cf. [31, 37]):

$$\eta(\nu) = \left[(1-\omega)\,\rho\left(\left(M_\ell\right)_r^{\delta l} A_\ell\right)\right]^\nu, \quad \text{with } \lim_{\nu\to\infty} \eta(\nu) \longrightarrow 0, \tag{5}$$

where

$$\omega > 1 - \frac{1}{\rho\left(\left(M_\ell\right)_r^{\delta l} A_\ell\right)}. \tag{6}$$

Moreover, sharp estimates for the convergence of Multigrid algorithms, for generalized smoothing, have been proven by Bank and Douglas in [38].

Moreover, Hackbusch has found the optimal values for the damping parameter for various iterative schemes and various PDEs; compare [18, 20, 24]. The proposed scheme requires a damping parameter ω. The damping parameter cannot be defined optimally for the problem, so an algorithm for the dynamic computation of its value must be used. The value of the damping parameter can be computed by the Dynamic Over/Under Relaxation (DOUR) algorithm; compare [24]. The DOUR algorithm predicts dynamically the value of the damping parameter (predictor-corrector) to ensure the convergence of the smoothing scheme.

Given an initial damping parameter ω, we use (1) to predict a value for $x_\ell^{(k+1)}$:

$$\widetilde{x}_\ell^{(k+1)} = x_\ell^{(k)} + \omega\left(M_{\text{drptol}}^{l\text{fill}}\right)_\ell \left(b_\ell - A x_\ell^{(k)}\right). \tag{7}$$

The corrected value is given by

$$x_\ell^{(k+1)} = \widetilde{x}_\ell^{(k)} + \kappa \Delta x_\ell^{(k)}, \tag{8}$$

where

$$\Delta x_\ell^{(k)} = \widetilde{x}_\ell^{(k)} - x_\ell^{(k)},$$

$$\kappa = \frac{\left(\Delta x_\ell^{(k)}\right)^T \left(b_\ell - A_\ell \widetilde{x}_\ell^{(k)}\right)}{\left(\Delta x_\ell^{(k)}\right)^T A_\ell \Delta x_\ell^{(k)}}. \tag{9}$$

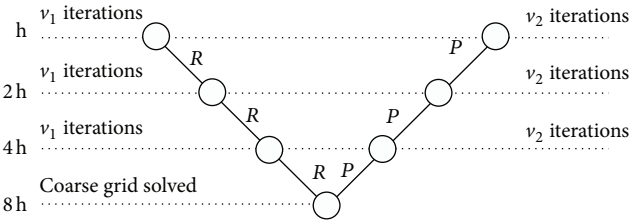


FIGURE 2: The Multigrid V-Cycle for four levels with ν_1 presmoothing iterations and ν_2 postsmoothing iterations.

From (7), (8), and (9) we obtain

$$x_\ell^{(k+1)} = x_\ell^{(k)} + \omega_e \left(M_{\text{drptol}}^{l\text{fill}}\right)_\ell \left(b_\ell - A x_\ell^{(k)}\right), \tag{10}$$

where $\omega_e = \omega(1 + \kappa)$ is the effective relaxation parameter and (10) is the proposed two-stage nonstationary smoothing scheme. Further information about the DOUR algorithm and its convergence analysis is given in detail in [24].

The vectors needed by the iterative scheme for the solution process of the linear system must be transferred across different grid levels. Transfer operators are special operators that are used to transfer these vectors from coarser to finer grids and finer to coarser grids. The operators are divided into two categories: the restriction operator and the prolongation operator. The restriction operator R is used to transfer vectors from finer to coarser grids. An effective choice for the restriction operator is the full-weighting, which in three dimensions can be expressed as presented in [22, 39].

The prolongation operator P is an interpolation procedure used to transfer vectors from coarser to finer grids. An effective choice for the prolongation operator is the trilinear interpolation. The trilinear interpolation can be expressed as presented in [22, 39].

These operators $A_{\ell+1}$ and A_ℓ are related through the Galerkin condition, $A_{\ell+1} = RA_\ell P$ so the mapping on the data is simplified; compare [14, 30]. Transferring vectors in the Multigrid method requires about 25% of the total computational work (cf. [15]) so the selection of higher order interpolation schemes may lead to excessive computational cost with no extra gain in the convergence behavior (cf. [15, 22]). A sparse matrix representation of these operators is used and thus the transfer operation between the levels is limited to matrix "times" vector multiplication, and their parallelization is covered by the parallelization of the matrix-vector multiplication. Further information about the transfer operators is presented in [14, 15, 22, 34].

The cycle strategy refers to the sequence in which the grids are visited and the respective corrections are obtained (cf. [15, 22, 34]). The most commonly used cycle strategy is the V-Cycle, where the method descends to coarser levels executing ν_1 smoother iterations in each level and then the method ascends executing ν_2 iterations in each level. The V-Cycle strategy is depicted in Figure 2. More information about the cycle strategies and the V-Cycle is presented in [29, 32, 40, 41].

The parallelization of the V-Cycle algorithm is performed only in the higher order levels to ensure the efficiency of the scheme; compare [39]. Lower order levels possess computational effort comparable to the parallelization overhead produced during the procedure of creating and detaching threads present in the parallel computation. Therefore, only levels containing a substantial number of unknowns are computed in parallel. The other levels are computed sequentially. As the number of levels increases, the sequential part requires less computational effort compared to higher order levels, thus increasing the speedup and efficiency of the proposed scheme.

The parallel truncated V-Cycle Multigrid algorithm with GenAspI parallel smoothing (cf. [15, 22, 25, 34, 39]) has been given in [39].

3. Parallel Particle Mesh Type Algorithms

The Particle Mesh method (PM), introduced by Hockney in [1], avoids the computation of every force pair in the system via the Newtonian gravitation and reduces the computational cost of the N-Body simulation; compare [1–4]. This method introduces the so-called "superparticles" term, which defines a large formation of bodies that react together through their gravitational force as a whole. Hence, PM is more efficient than the direct method due to the fact that the nodal points are less than the number of bodies in the system and so the force pairs that have to be computed are much smaller. The PM method is not as accurate as the Particle Particle method. The method computes rapidly the long range (field force) force but is not taking into account the gravity influence between the short range bodies. This approach is more suitable on large scale systems, where low complexity and large scale results are more important; compare [1–4].

The gravity potential is computed by the Partial Differential Equation (cf. [1, 40]),

$$\nabla\Phi = -4\pi G\rho, \qquad (x, y, z) \in \Omega, \tag{11}$$

subject to Dirichlet boundary conditions,

$$\Phi = 0, \qquad (x, y, z) \in \partial\Omega, \tag{12}$$

where Ω denotes the region, $\partial\Omega$ is the boundary of the region, G is the gravitational constant, and ρ is the mass density in each nodal point computed by the mass of the neighboring bodies.

Applying the Finite Difference method with the seven-point stencil on the PDE (cf. (11)-(12)) results in solving

$$A\phi = f, \tag{13}$$

where A is a large sparse diagonally dominant symmetric matrix, ϕ is the gravity potential at each nodal point, and f is the right hand side vector consisting of the forcing term and respective boundary conditions.

Each body contributes to the density of the concentrated mass along the grid points, resulting in the mass density vector in that PDE; compare (11). The simplest interpolation method used for computing the mass density vector is the Nearest Grid Point (NGP). The NGP method suggests that the body contributes to the nearest nodal point; thus bodies

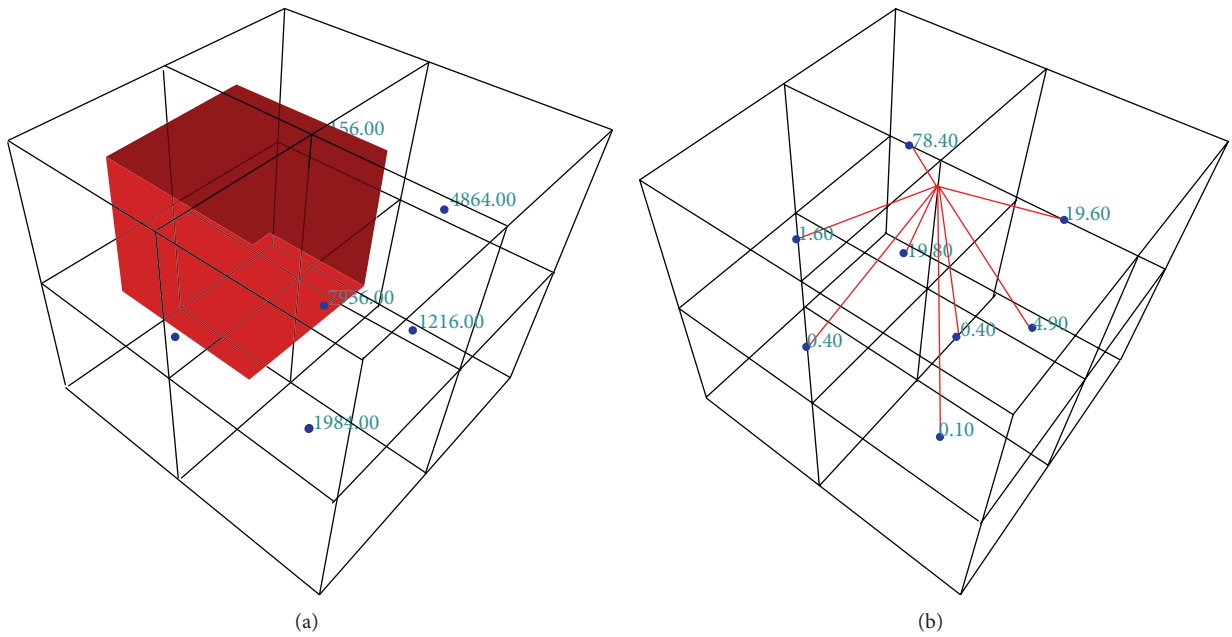


FIGURE 3: On the left is the Cloud in Cell method in three dimensions. The cube denotes the "cloud" and the values are the mass densities. On the right is the inverse Cloud in Cell method in three dimensions. The lines denote the weights.

within one cell are not interactive with other nodal points increasing the error. In the proposed scheme, the multilinear interpolation is used, known as Cloud in Cell (CIC); compare [1]. The Cloud in Cell method interpolates the contribution of mass to every nearby grid point via the so-called "cloud" increasing the accuracy of the computations. The cloud can be a square or a cube with its center at the location of the body. The method is based on the area (two dimensions) or the volume (three dimensions) of every part of the "cloud cell" with respect to the other mesh cells, to create the interpolation weights. In three dimensions, using the Cloud in Cell method on a body with mass m_p located at (x_p, y_p, z_p) results in affecting the mass density of the eight surrounding grid points (cube). The mass density (cf. [1, 4]) in the nearest mesh cell can be computed by

$$\rho_{i,j,k} = \frac{m_p}{h^6}\left(h - \Delta_{pi}\right)\left(h - \Delta_{pj}\right)\left(h - \Delta_{pk}\right), \tag{14}$$

where h is the mesh size and $\Delta_{pi}, \Delta_{pj}, \Delta_{pk}$ are the distances between the body and the center of the mesh cell. The remaining nodal point densities are computed analogously.

The bodies in the Particle Mesh method are divided among the computational nodes for the parallelization of the algorithm. So every node of the hybrid system computes the mass density of their own bodies in parallel on the processors through the parallel Cloud in Cell method. The computed vectors are sent from every other node to the head node through the MPI parallel environment. The PDE subject to Dirichlet boundary conditions (cf. (11)-(12)) discretized with the Finite Differences method is solved with the parallel GenAspI-MGV method on the head node, and the gravitational potential is the solution of the linear system. The acceleration g in the center of the cells (cf. [1–4]) can be computed by the following equation:

$$g = -\nabla\Phi = -\left(\frac{\partial\Phi}{\partial x_i}, \frac{\partial\Phi}{\partial y_j}, \frac{\partial\Phi}{\partial z_k}\right). \tag{15}$$

Central differences are used to approximate the first order partial derivative; that is,

$$\frac{\partial\Phi}{\partial x} \approx \frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h} + O\left(h^2\right). \tag{16}$$

The method is computed in parallel using the OpenMP at all nodes. The acceleration on the center of the cells must be interpolated back to the bodies. Applying the parallel CIC method for the inverse operation on a body in the three dimensions results in the following formula for the acceleration (cf. [1, 4]):

$$\begin{aligned} g_p = {} & k_1 g_{i,j,k} + k_2 g_{i,j,k+1} + k_3 g_{i,j+1,k} \\ & + k_4 g_{i,j+1,k+1} + k_5 g_{i+1,j,k} \\ & + k_6 g_{i+1,j+1,k} + k_7 g_{i+1,j,k+1} + k_8 g_{i+1,j+1,k+1}, \end{aligned} \tag{17}$$

where index p denotes the particle and k_i are computed weights through the CIC method. For the nearest mesh cell the k_l is computed by

$$k_1 = \left(h - \Delta_{pi}\right)\left(h - \Delta_{pj}\right)\left(h - \Delta_{pk}\right). \tag{18}$$

The remaining weights are computed in an analogous way. The acceleration vector is sent back to the other nodes from the head node through the MPI interface. The Cloud in Cell method and the inverse procedure are depicted in Figure 3.

```
On every node
    On the node threads
        (i) Compute the mass density. /∗equation (14)∗/
        (ii) Send the mass density to the head node.
    end
end
Only the head node
    On the node threads
        (i) Compute gravity potential with the GenAspI-MGV method /∗cf. [4]∗/
        (ii) Send the solution vector to the other nodes
    end
end
On every node
    On the node threads
        (i) compute acceleration in the center of the cells. /∗equation (15)∗/
        (ii) inverse interpolation of the acceleration to the bodies. /∗equation (17)∗/
        (iii) compute new position through Velocity Verlet Integration. /∗equations (19)-(20)∗/
    end
end
```

Algorithm 1: Parallel PM algorithm.

Finally, knowing the acceleration for each body the equations of motion have to be integrated. On the proposed algorithms the Velocity Verlet integrator is utilized. The Velocity Verlet integrator is used widely on N-Body simulations due to its low computational cost and $O(h^2)$ accuracy; compare [33]. The Verlet integration of the equations of motion leads to the following equations:

$$\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\vec{\alpha}(t)\,\Delta t^2, \tag{19}$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\vec{\alpha}(t) + \vec{\alpha}(t+\Delta t)}{2}\Delta t, \tag{20}$$

where $\vec{x}$, $\vec{v}$, $\vec{\alpha}$ are the three-dimensional vectors of the position, the velocity, and the acceleration, respectively. The new positions are computed via the Velocity Verlet method, which is performed in parallel on each computational node using the OpenMP environment. Further information concerning the Particle Mesh method can be found in [1–4].

The new hybrid parallel PM algorithm, based on MPI and OpenMP, is given by the compact scheme in Algorithm 1.

The computational complexity of the Particle Mesh algorithm is

$$\begin{aligned} \text{Multiplications} &\approx \frac{319N}{\text{threads} * n_{\text{nodes}}} + \frac{4ng^3}{\text{threads}}, \\ \text{Additions} &\approx \frac{184N}{\text{threads} * n_{\text{nodes}}} + \frac{ng^3}{\text{threads}}, \end{aligned} \tag{21}$$

where n_{nodes} denotes the number of computational nodes and ng is the number of mesh grid points per dimension and has been proven in [42].

The Particle Particle Particle Mesh is a hybrid method, which consists of the Particle Mesh and the Particle Particle (direct) method; compare [1–4]. The P3M method divides the force into two parts: the short range force and the long range force. The long range force is computed via the PM method by solving the resulting sparse linear system with the Multigrid method in conjunction with the GenAspI algorithm. The short range forces are computed with the direct approach using the Newtonian gravitation. The computation of the short range force requires the definition of the short range distance between bodies.

Let us denote by r_{near} the maximum distance between two bodies that can be considered as neighbors. The distance is three or four times the mesh size. In the proposed algorithm the distance is considered as $r_{\text{near}} = 3h_1$, where h_1 is the mesh size. The distance of every pair of bodies in the system must be smaller than the r_{near} for the short range force to act. If this condition is applied on every pair of bodies, the computational cost will increase. Therefore, a new mesh, called chaining mesh, is created, with $h_2 = 4h_1$, to group the short range pairs; compare [1, 2, 4].

Each body compares its distance from the bodies that are on the nine (in two dimensions) nearby mesh cells of the chaining mesh, with the r_{near} distance. The neighbor chaining mesh cells are shown in Figure 4. If the distance of the bodies is less than r_{near}, the short range force between the two bodies is computed. This assumption of the nine nearby cells of the chaining mesh is based on the fact that bodies within the r_{near} distance cannot be outside these cells as depicted in Figure 4.

In three dimensions, these are twenty-seven cells, and the region of short range interactions around a body can be described as a sphere.

The bodies of the P3M method are also divided into the computational nodes for the parallelization of the algorithm. The computation of the long range force is parallelized the same way as the Particle Mesh method. For the computation of the short range force, every node assembles a vector containing the distinctive numbers of the associated bodies in the chaining mesh and sends it to the other nodes.

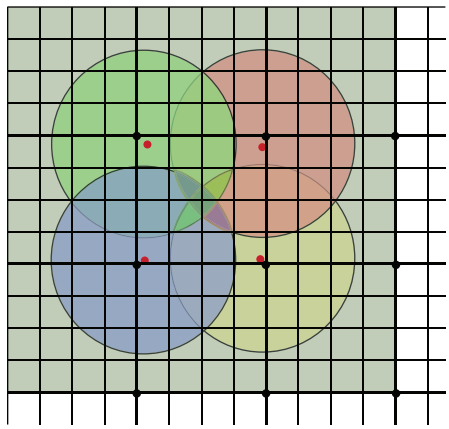


Figure 4: The nine neighbor chaining mesh cells. The bold mesh grid denotes the chaining mesh. The dots represent the position of a body in the center mesh cell (chaining mesh). The circles denote the r_{near} of each body.

After the determination of the close distance bodies, the short distance acceleration can be computed by the following equation which is derived by Fourier analysis of the overall potential (cf. [1, 2, 4]):

$$g_{\text{near}}(r) = Gm_i \frac{\vec{x}_0 - \vec{x}_i}{\left|\vec{x}_0 - \vec{x}_i\right|^3} \cdot \left[\operatorname{erfc}\left(-\frac{\vec{x}_0 - \vec{x}_i}{2r_{\text{near}}}\right) + \frac{\vec{x}_0 - \vec{x}_i}{r_{\text{near}}\sqrt{\pi}} \exp\left(-\frac{(\vec{x}_0 - \vec{x}_i)^2}{4{r_{\text{near}}}^2}\right)\right], \tag{22}$$

where G is the gravitational constant, $\vec{x}_0$ is the position of the computed body, m_i and $\vec{x}_1$ are the mass and the position of the close range body, and erfc is the complementary error function (cf. [1]):

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-r^2} dr = \frac{2}{\sqrt{\pi}} \int_0^\infty e^{-r^2} dr. \tag{23}$$

The erfc function is used, in order to remove the influence that was computed through the mass density on the long range force. The computation of the short range force for a body requires the knowledge of the position of each neighboring body, so the positions of the neighboring bodies should be transferred to its respective computational node. Further details can be found in [1].

The long range force is computed through the PM method. The PM method is used to compute the acceleration of each body, so it is preferable to compute the short range acceleration instead of the F_{near}. Thus, for each body

$$g_{\text{tot}} = g_{\text{near}} + g_{\text{far}}. \tag{24}$$

Finally, the new positions are computed by integrating the equations of motion with the Velocity Verlet integration in parallel (cf. [33]) as in the parallel PM method.

The P3M method corrects the short range solution in the system, but in large scale clusters with large number of concentrated bodies, the short range part of the algorithm monopolizes the simulation time, and the computational time increases.

The new Hybrid parallel P3M algorithm, based on MPI and OpenMP, is given by the compact scheme in Algorithm 2.

The computational complexity of the P3M algorithm is

$$\text{Multiplications} \approx \frac{653N * \left(N_\phi\right)}{n_{\text{nodes}} * \text{threads}} + \frac{4ng}{\text{threads}},$$

$$\text{Additions} \approx \frac{805N * \left(N_\phi\right)}{n_{\text{nodes}} * \text{threads}} + \frac{ng^3}{\text{threads}} + \frac{ncg^3}{n_{\text{nodes}} * \text{threads}}, \tag{25}$$

where n_{nodes} denotes the number of computational nodes, ng is the number of mesh grid points, and ncg is the number of chaining mesh points per dimension and has been proven in [42].

4. Numerical Results

In this section the applicability as well as performance of the proposed scheme is demonstrated by simulating the three-dimensional torus space with various numbers of bodies.

The numerical tests were performed on a cluster of eight computer nodes. The head node was a dual socket quad core Xeon processor (8 cores) with 2.5 GHz and 8 GB RAM. The other nodes had 2 quad sockets dual core Xeon processors (8 cores) with 2.5 GHz and 8 GB RAM. The cluster network was an Ethernet network with 1 Gbps bandwidth. The domain chosen for the simulations was the unit cube. The termination criterion was set to $\|r_i\| < 1e-10\|r_0\|$, where $r_i = b - Ax_i$. The presmoothing and postsmoothing steps were set to $\nu_1 = 2$ and $\nu_2 = 2$. For computing the GenAspI matrix, the drop tolerance and levels of fill were set to drptol = 0.0 and $l$fill = 1, respectively. For these parameter values the method presented the fastest performance results; compare [39]. The levels below $n = 343$ were executed sequentially because the computational overhead exceeds the computational effort required in order to obtain the coarse grid corrections. It should be noted that the preprocessing cost of the simulation algorithms designed was several orders less than the computational time needed for the model problems, rendering the methods efficient by slightly enlarging the computational cost. Moreover, new computational schemes that will enhance performance and accuracy of the simulations are researched.

4.1. Particle Mesh Numerical Results. The proposed method based on the Particle Mesh method and the GenAspI-MGV, as shown in the results, has an advantage regarding the computational speed.

In Table 1, the overall performance of the Particle Mesh simulation algorithm based on the GenAspI-MGV method for various numbers of bodies, various numbers of cluster nodes, various numbers of threads, and various resolutions

On every node
  On the node threads
    (i) Compute the mass density. /∗equation (14)∗/
    (ii) Compute the chaining mesh place vector.
    (iii) Send the mass density to the head node, and the chaining mesh place vector to every other node.
  end
end
Only the head node
  On the node threads
    (i) Compute gravity potential with the GenAspI-MGV method, /∗cf. [4]∗/
    (ii) Send the solution vector to the other nodes.
    (iii) Send the positions and the mass of the bodies to the other nodes.
  end
end
On every node
  On the node threads
    (i) compute acceleration in the center of the cells. /∗equation (15)∗/
    (ii) reverse interpolate the acceleration to the bodies. /∗equation (17)∗/
    (iii) compute the short range force. /∗equation (22)∗/
    (iv) compute new position through Velocity Verlet Integration. /∗equations (19)-(20)∗/
  end
end

Algorithm 2: Parallel P3M algorithm.

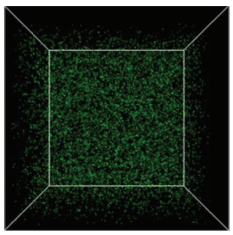
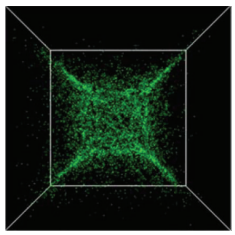
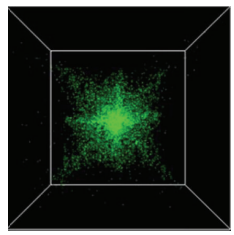
Table 1: The overall performance of the Particle Mesh simulation algorithm based on the GenAspI-MGV method for various numbers of particles, various numbers of cluster nodes (N), various numbers of threads, and various resolutions (results in seconds.hundreds).

		Performance		
N	Threads	$h = 1/16$	$h = 1/32$	$h = 1/64$
Number of bodies = 30.000.000				
1	1	76.65	77.75	88.88
2	2	22.68	24.07	26.77
	4	12.14	12.28	14.39
	8	7.06	6.53	8.19
4	2	11.07	12.10	14.27
	4	6.10	6.14	7.81
	8	3.31	3.40	4.79
8	2	5.70	5.85	8.05
	4	3.06	3.13	4.56
	8	1.72	1.73	3.05
Number of bodies = 40.000.000				
1	1	105.81	104.31	116.48
2	2	32.86	31.23	35.73
	4	16.72	16.48	20.05
	8	8.83	8.95	10.51
4	2	15.81	18.73	18.36
	4	8.24	8.55	9.98
	8	4.38	4.41	6.09
8	2	7.85	7.97	9.88
	4	4.28	4.16	5.64
	8	2.37	2.25	3.60

Table 2: The speedups of the Particle Mesh simulation algorithm based on the GenAspI-MGV method for various numbers of bodies, various numbers of cluster nodes (N), various numbers of threads, and various resolutions.

		Speedup		
N	Threads	$h = 1/16$	$h = 1/32$	$h = 1/64$
Number of bodies = 30.000.000				
2	2	3.38	3.23	3.32
	4	6.32	6.33	6.19
	8	10.87	11.91	10.88
4	2	6.52	6.42	6.24
	4	12.57	12.66	11.39
	8	23.15	22.83	18.58
8	2	13.45	13.29	11.06
	4	25.04	24.85	19.54
	8	44.43	44.75	29.17
Number of bodies = 40.000.000				
2	2	3.22	3.34	3.26
	4	6.13	6.33	5.82
	8	11.61	11.65	11.10
4	2	6.48	6.63	6.35
	4	12.44	12.20	11.68
	8	23.41	23.66	19.14
8	2	13.06	13.08	11.81
	4	23.93	25.06	20.67
	8	43.24	46.30	32.3

Table 3: Visual results of the P3M simulation with different body densities and their sequential performances are presented.

N-bodies = 4.000.000		
Frame 1	Frame 2	Frame 3
		
Performance		
25.3378 s	109.3888 s	221.19557 s

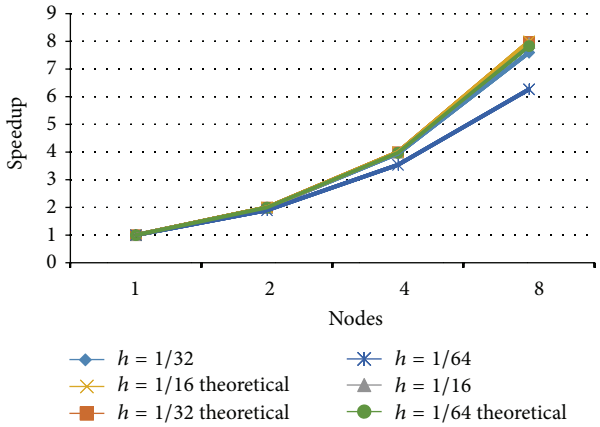


Figure 5: The theoretical estimates compared to the speedups of the parallel Particle Mesh algorithm, for different mesh sizes and different number of nodes for N = 20,000,000 bodies.

is presented. In Table 2, the speedups of the Particle Mesh simulation algorithm, based on the GenAspI-MGV method for various numbers of bodies, cluster nodes, threads, and various resolutions, are presented.

In Figure 5, the theoretical estimates along with the experimental results for the speedup on the parallel Particle Mesh algorithm, for different mesh sizes and different numbers of nodes and for N = 20,000,000 bodies, are presented. In Figure 6, visual results of the Particle Mesh simulation on twelve frames are presented.

4.2. P3M Numerical Results. The proposed P3M method has the same characteristics as the proposed PM method but lacks the computational speed of the PM method due to the direct part of the algorithm, which raises the accuracy of the results, as expected.

The numerical results of the P3M method were derived from a high density system with a large number of concentrated bodies that reveal the flaws of the method regarding high density body clusters. In Table 3, visual results of the P3M simulation with different body densities and their sequential performances are presented.

Table 4: The overall performance of the P3M simulation algorithm based on the GenAspI-MGV method for various numbers of particles, various numbers of cluster nodes (N), various numbers of threads, and various resolutions (results in seconds.hundreds).

		Performance		
N	Threads	h = 1/16	h = 1/32	h = 1/64
Number of bodies = 2.000.000				
1	1	58.40	58.37	60.75
2	2	18.42	18.53	20.05
	4	11.60	11.64	12.88
	8	8.08	8.21	9.44
4	2	11.07	11.29	13.67
	4	8.34	8.46	9.55
	8	6.72	6.84	7.83
8	2	8.42	8.45	9.90
	4	6.75	6.89	7.84
	8	5.78	5.80	7.03
Number of bodies = 4.000.000				
1	1	220.79	221.9	223.92
2	2	69.65	70.00	71.54
	4	43.90	44.01	45.2
	8	30.55	30.90	32.34
4	2	44.48	44.51	45.69
	4	31.31	31.67	32.65
	8	25.03	25.39	26.18
8	2	31.53	31.98	33.11
	4	24.95	25.01	26.24
	8	21.91	22.00	23.13

In Table 4, the overall performance of the P3M algorithm, based on the GenAspI-MGV method for various numbers of bodies, various numbers of cluster nodes, various

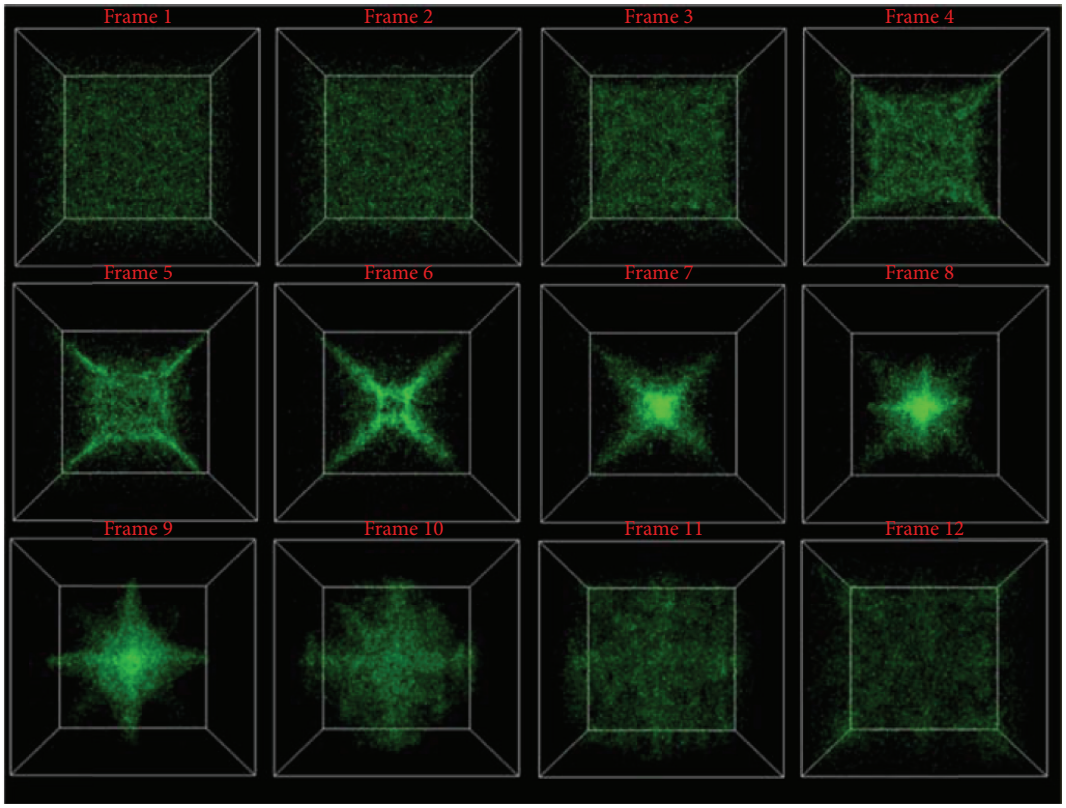


Figure 6: Visual results from the Particle Mesh simulation (twelve frames).

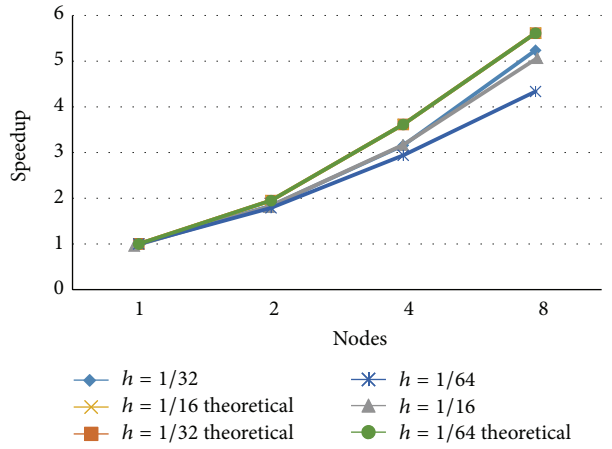


Figure 7: The theoretical estimates compared to the speedups of the parallel P3M algorithm, for different mesh sizes and different numbers of nodes and for $N = 2{,}000{,}000$ bodies.

numbers of threads, and various resolutions, is presented. In Table 5, the speedups of the parallel P3M algorithm based on the GenAspI-MGV method for various numbers of bodies, cluster nodes, threads, and various resolutions are presented. In Figure 7, the theoretical estimates along with the experimental results for the speedup on the parallel P3M algorithm, for different mesh sizes and different numbers of nodes and for $N = 2{,}000{,}000$ bodies, are presented.

5. Conclusion

The proposed schemes based on the GenAspI-MGV method are proven to be highly effective regarding the computational performance; compare [25, 31, 37]. Moreover, the numerical results indicate the effectiveness of the proposed parallelization scheme especially for the Particle Mesh method. The numerical results reveal the differences of the two N-Body

Table 5: The speedups of the P3M simulation algorithm based on the GenAspI-MGV method for various number of bodies, various numbers of cluster nodes (N), various number of threads, and various resolutions.

		Speedup		
N	Threads	$h = 1/16$	$h = 1/32$	$h = 1/64$
Number of bodies = 2.000.000				
2	2	3.15	3.14	3.03
	4	4.99	5.00	4.72
	8	7.17	7.09	6.45
4	2	5.24	5.15	4.45
	4	6.95	6.87	6.37
	8	8.63	8.51	7.78
8	2	6.88	6.89	6.15
	4	8.58	8.45	7.76
	8	10.03	10.03	8.66
Number of bodies = 4.000.000				
2	2	3.17	3.16	3.13
	4	5.03	5.04	4.96
	8	7.23	7.16	6.94
4	2	5.00	4.97	4.91
	4	7.06	6.99	6.87
	8	8.83	8.72	8.57
8	2	7.01	6.92	6.78
	4	8.86	8.85	8.55
	8	10.08	10.06	9.70

simulation methods (Particle Mesh, P3M) regarding the computational performance. The Particle Mesh method is faster than the P3M method. Finally, future work will be concentrated on the performance of P3M method with new approaches utilizing adaptive schemes; compare [43].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, 1981.

[2] S. J. Aarseth, *Gravitational N-Body Simulations: Tools and Algorithms*, Cambridge University Press, Cambridge, UK, 2003.

[3] J. Barnes and P. Hut, "A hierarchical *O(N log N)* force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[4] R. W. Hockney, S. P. Goel, and J. W. Eastwood, "A 10000 particle molecular dynamics model with long range forces," *Chemical Physics Letters*, vol. 21, no. 3, pp. 589–591, 1973.

[5] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, 1987.

[6] W. Dehnen, "A very fast and momentum-conserving tree code," *The Astrophysical Journal Letters*, vol. 536, no. 1, pp. L39–L42, 2000.

[7] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand, "Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations," *SIAM Journal on Scientific Computing*, vol. 27, no. 3, pp. 774–792, 2005.

[8] J. Lee, J. Zhang, and C.-C. Lu, "Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 9, pp. 2277–2287, 2004.

[9] R. Yokota and L. A. Barba, "A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems," *International Journal of High Performance Computing Applications*, vol. 26, no. 4, pp. 337–346, 2012.

[10] J. Binney and S. Tremaine, *Galactic Dynamics*, Princeton University Press, Princeton, NJ, USA, 2nd edition, 2011.

[11] A. W. Appel, "An efficient program for many-body simulation," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 85–103, 1985.

[12] T. Darden, D. York, and L. Pedersen, "Particle mesh Ewald: an $N\cdot\log(N)$ method for Ewald sums in large systems," *The Journal of Chemical Physics*, vol. 98, no. 12, pp. 10089–10092, 1993.

[13] G. Efstathiou, M. Davies, C. S. Frenk, and S. D. M. White, "Numerical techniques for large cosmological N-Body simulations," *The Astrophysical Journal Supplement Series*, vol. 59, pp. 241–260, 1985.

[14] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.

[15] L. W. Briggs, V. E. Henson, and F. S. McCormick, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, 2000.

[16] O. Bröker, M. J. Grote, C. Mayer, and A. Reusken, "Robust parallel smoothing for multigrid via sparse approximate inverses," *SIAM Journal on Scientific Computing*, vol. 23, no. 4, pp. 1396–1417, 2001.

[17] P. Frederickson, "High performance parallel multigrid algorithms for unsctructured grids," in *Proceedings of the 7th Copper Mountain Conference on Multigrid Methods*, CP 3339, pp. 317–326, 1996.

[18] W. Hackbusch, "Convergence of multigrid iterations applied to difference equations," *Mathematics of Computation*, vol. 34, no. 150, pp. 425–440, 1980.

[19] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer, 1985.

[20] W. Hackbusch, "Multigrid convergence theory," in *Multigrid Methods*, vol. 960 of *Lecture Notes in Mathematics*, pp. 177–219, Springer, Berlin, Germany, 1982.

[21] W. Hackbusch, "On the convergence of multigrid iterations," *Numerische Mathematik*, vol. 9, pp. 213–239, 1981.

[22] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.

[23] P. Wesseling, "The rate of convergence of a multiple grid method," in *Numerical Analysis: Proceedings of the 8th Biennial Conference Held at Dundee, Scotland, June 26–29, 1979*, vol. 773 of *Lecture Notes in Mathematics*, pp. 164–184, Springer, Berlin, Germany, 1980.

[24] R. Haelterman, J. Viederndeels, and D. van Heule, "Non-stationary two-stage relaxation based on the principle of aggregation multi-grid," in *Computational Fluid Dynamics 2006*, pp. 243–248, Springer, Berlin, Germany, 2006.

[25] C. K. Filelis-Papadopoulos and G. A. Gravvanis, "Generic approximate sparse inverse matrix techniques," *International Journal of Computational Methods*, vol. 11, no. 6, Article ID 1350084, 16 pages, 2014.

[26] G. A. Gravvanis, "Explicit approximate inverse preconditioning techniques," *Archives of Computational Methods in Engineering*, vol. 9, no. 4, pp. 371–402, 2002.

[27] G. A. Gravvanis, "High performance inverse preconditioning," *Archives of Computational Methods in Engineering*, vol. 16, no. 1, pp. 77–108, 2009.

[28] G. A. Gravvanis, "The rate of convergence of explicit approximate inverse preconditioning," *International Journal of Computer Mathematics*, vol. 60, no. 1-2, pp. 77–89, 1996.

[29] E. A. Lipitakis and D. J. Evans, "Explicit semi-direct methods based on approximate inverse matrix techniques for solving boundary-value problems on parallel processors," *Mathematics and Computers in Simulation*, vol. 29, no. 1, pp. 1–17, 1987.

[30] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[31] C. K. Filelis-Papadopoulos and G. A. Gravvanis, "On the multigrid method based on finite difference approximate inverses," *Computer Modeling in Engineering & Sciences*, vol. 90, no. 3, pp. 233–253, 2013.

[32] G. A. Gravvanis, C. K. Filelis-Papadopoulos, and P. I. Matskanidis, "Algebraic multigrid methods based on generic approximate inverse matrix techniques," *Computer Modeling in Engineering & Sciences*, vol. 100, no. 4, pp. 323–345, 2014.

[33] L. Verlet, "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules," *Physical Review*, vol. 159, no. 1, pp. 98–103, 1967.

[34] P. Wesseling, "Theoretical and practical aspects of a multigrid method," *SIAM Journal on Scientific and Statistical Computing*, vol. 3, no. 4, pp. 387–407, 1982.

[35] E. Chow, "A priori sparsity patterns for parallel sparse approximate inverse preconditioners," *SIAM Journal on Scientific Computing*, vol. 21, no. 5, pp. 1804–1822, 2000.

[36] E. Chow, "Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns," *International Journal of High Performance Computing Applications*, vol. 15, no. 1, pp. 56–74, 2001.

[37] G. A. Gravvanis, C. K. Filelis-Papadopoulos, and E. A. Lipitakis, "On numerical modeling performance of generalized preconditioned methods," in *Proceedings of the 6th Balkan Conference in Informatics (BCI '13)*, C. K. Georgiadis, P. Kefalas, and D. Stamatis, Eds., pp. 23–30, ACM, Thessaloniki, Greece, September 2013.

[38] R. E. Bank and C. C. Douglas, "Sharp estimates for multigrid rates of convergence with general smoothing and acceleration," *SIAM Journal on Numerical Analysis*, vol. 22, no. 4, pp. 617–633, 1985.

[39] P. E. Kyziropoulos, C. K. Filelis-Papadopoulos, and G. A. Gravvanis, "N-body simulation based on the Particle Mesh method using multigrid schemes," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '13)*, pp. 471–478, Kraków, Poland, September 2013.

[40] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1996.

[41] G. Beutler, *Methods of Celestial Mechanics*, Springer, New York, NY, USA, 2005.

[42] P. E. Kyziropoulos, *Parallel simulation of N-bodies [Diploma Thesis]*, Democritus University of Thrace, 2013.

[43] H. M. P. Couchman, "Mesh-refined P3M—a fast adaptive N-body algorithm," *The Astrophysical Journal Letters*, vol. 368, no. 2, pp. L23–L26, 1991.