Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 471–478

N-body simulation based on the Particle Mesh method using Multigrid schemes

P. E. Kyziropoulos
Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, University Campus, Kimmeria, GR 67100 Xanthi, Greece
panakyzi@ee.duth.gr

C. K. Filelis-Papadopoulos
Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, University Campus, Kimmeria, GR 67100 Xanthi, Greece
chripapa9@ee.duth.gr

G. A. Gravvanis
Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, University Campus, Kimmeria, GR 67100 Xanthi, Greece
ggravvan@ee.duth.gr

***Abstract*—Through the last decades multigrid methods have been used extensively in the solution of large sparse linear systems derived from the discretization of Partial Differential Equations in two or three space variables, subject to a variety of boundary conditions. Due to their efficiency and convergence behavior, multigrid methods are used in many scientific fields as solvers or preconditioners. Herewith, we propose a new algorithm for N-body simulation, based on the V-Cycle multigrid method in conjunction with Generic Approximate SParse Inverses (GenAspI). The N-body problem chosen is in toroidal 3D space and the bodies are subject only to gravitational forces. In each time step, 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 and through the velocity Verlet method compute the new position, velocity and acceleration of each respective body. Moreover, a parallel version of the multigrid algorithm with a truncated approach in the parallel levels is utilized for the fast solution of the linear system. Furthermore parallel results are provided which depict the efficiency and performance for the proposed multigrid N-body scheme.**

I. Introduction

LET US consider the Partial Differential Equation (PDE) describing the gravity potential in a domain Ω, [19]:

$$\Delta\Phi = 4\pi G\rho, (x, y, z) \in \Omega \quad (1)$$

subject to Dirichlet boundary conditions

$$\Phi = 0, (x, y, z) \in \vartheta\Omega \quad (1.a)$$

where Ω denotes the region where the problem resides, $\partial\Omega$ 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, for the PDE (1)-(1.a) results in solving a seven diagonal linear system,

$$A\varphi = f, \quad (2)$$

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

The linear system (2) derived from the discretization of the 3D PDE can be solved by the multigrid method. Multigrid methods have been used extensively, during the last decades, in a variety of scientific fields such as Computational Fluid Dynamics, Computational Economics and Partial Differential Equations, due to their near optimal computational complexity and convergence behavior, [3, 4, 5, 10, 15, 16, 17, 18, 21, 23, 24]. Multigrid methods are based on the observation that the low-frequency components of the error are not effectively damped by a stationary iterative method. However, the high frequency components of the error are quickly reduced towards zero within the first few iterations, [3, 4, 21]. In order to handle the low frequency components of the error, multigrid methods utilize a grid hierarchy consisting of coarser levels with higher mesh size (h) and by projecting the finer problem to the coarser levels the lower frequency components are becoming more oscillatory and can be damped efficiently by a stationary iterative solver. The vectors required in each level are transferred between the respective grid with the use of two operators: the prolongation and the restriction operator, which transfer vectors from coarser to finer grids and vice versa, [3, 4, 21]. The sequence in which the coarser grids are visited and the respective coarse grid corrections to the solution are obtained is referred to as the cycle strategy, [3, 4, 21]. A 3D Geometric multigrid hierarchy is depicted in Figure 1. In order to accelerate the convergence of the multigrid method the Dynamic Over / Under Relaxation (DOUR) scheme is used, [18], in conjunction with the GENeric Approximate SParse Inverse (GenAspI) matrix, as the smoother for the multigrid method, [8].

Approximate inverses have been used as preconditioners for various iterative schemes due to their inherent parallelism and convergence behavior, [5, 11, 12, 13, 20]. Moreover, approximate inverses have been used effectively in conjunction with the multigrid method for a variety of problems, [5, 9, 10, 14]. Recently, classes of Generic Approximate Inverses have been proposed, [8,14], that can handle any sparsity pattern of the coefficient matrix A, based on Incomplete LU factorization with zero fill-in. Unlike their predecessors, [11, 12, 13, 20], these Generic schemes are not limited by the structure of the coefficient matrix or the method used for the discretization and produce approximate inverses with sparsity patterns, based on Powers of Sparsi-

978-1-4673-4471-5/$25.00

fied Patterns (PSM's), [6, 7], using adequate dropping strategies to further sparsify the initial sparsity pattern of the coefficient matrix A, thus leading to sparser more efficient approximate inverses. The GenAspI matrices are then computed using a modified procedure introduced in the GENeric Approximate Banded Inverses class (GenAbI), [14], according to an a priori known sparsity pattern. The GenAspI matrices are used as preconditioners in the damped Richardson scheme, in conjunction with the DOUR scheme, and V-cycle strategy to derive the GenAspI-MGV method which is used to obtain the gravity potential in each time step of the N-body simulation scheme proposed. The parallelization of the GenAspI-MGV method is performed with a new approach where the lower order levels are executed sequentially in order to avoid overhead in levels with low computational cost.

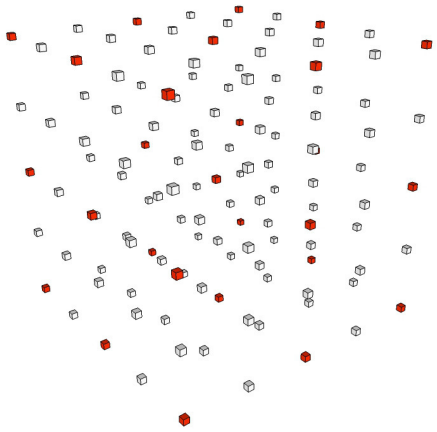

Fig. 1 Grid hierarchy of a three-dimensional PDE discretized with the Finite Differences method for mesh size h=1/2 (red points) and h=1/4 (red and white points).

The N-body simulation is a simulation of a dynamical system of particles, under the influence of, mainly the gravitational force. N-body simulations have become a fundamental tool in the study of complex physical systems, [19]. Starting from a basic physical interaction (e.g., gravitational, Coulomb) one can follow the dynamical evolution of a system of bodies, which represent the phase-space density distribution of the system. The greater the number of the particles used on the simulation results in more accurate and complete results, [19]. There are a lot of methods that can be used to calculate such kind of forces in a confined space. A direct approach to the problem called particle-particle simulation (P-P) assumes that on a simulation containing N bodies, there are N×(N-1) force pairs (Newtonian gravitation), [19]. This direct approach to the problem scales with $O(N^2)$ complexity, where N denotes the number of bodies, rendering the scheme restrictive for large values of particles.

Herewith, we propose a new scheme for computing the gravity potential and thus the forces in the bodies, using the Particle Mesh method and the GenAspI-MGV method to accelerate the solution of the linear system. The Particle Mesh method neglects the close interactions between particles and takes into account only the dynamics of superparticles consisting of a great number of particles. The density of these masses is then added according to weights computed from the distance of each node to the grid points surrounding each respective particle resulting in the rhs vector of the linear system (2). The assignment of the respective weights in each nodal point is performed using the Cloud in Cell (CIC) method, [19]. The linear system is solved using an iterative method and the potential is computed at the cell centers. Finally, by integrating the equations of motion through the Verlet integrator, [22], the new position of each particle is computed. Repeating the aforementioned process leads to the simulation of the system of particles. The Particle Mesh method is efficient due to the fact that the nodal points are, in general, less than the number of particles and thus a large number of pairing forces is not computed. Moreover, the low computational complexity of the mutligrid method utilized in each time step for the solution of the linear system renders the algorithm efficient especially for increasing numbers of particles.

The proposed N-body algorithm is parallelized using OpenMP. OpenMP is a collection of directives available for a variety of languages including C/C++/Fortran used to parallelize programs for Symmetric Multi-Processing Units.

Finally, numerical results on the performance and convergence behavior of the proposed GenAspI-MGV schemes are given for solving three-dimensional N-body simulation problems.

II. Multigrid Method In Conjunction With Generic Approximate Sparse Inverses

The multigrid method, especially in the last decades, is used extensively by the scientific community in the fields of computational physics, computational fluid dynamics and financial engineering due to its near-optimal complexity and its convergence behavior, [3, 4, 5, 9, 14, 15, 16, 17, 21, 23, 24].

Multigrid methods are composed by four distinct components: Smoothers, Prolongation and Restriction operators and Cycle strategy. The smoothers are an essential component of the multigrid algorithm and are used to obtain the respective corrections for the solution on each level. Smoothers are stationary iterative methods that can be described by the following relation, [4,5,21,23],

$$x_l^{(k+1)} = x_l^{(k)} + \omega M_l\left(b_l - A_l x_l^{(k)}\right),\ k = 0,1,2,.... \quad (3)$$

where l denotes the level in which the smoother is applied, M_l is the preconditioner to the Richardson's iterative method and ω is the damping parameter with values between 0 and 2. In case where the $M_l=D_l^{-1}$, the iterative scheme (3) results in the Damped Jacobi method, [4, 21, 23]. Substituting $M_l = \left(M_{lfill}^{drptol}\right)_l$, where $\left(M_{lfill}^{drptol}\right)_l$ is the l-th level Generic Approximate Sparse Inverse with lfill levels of fill and drptol drop tolerance results in the proposed parametric smoothing scheme. The Generic Approximate Sparse Inverse

(GenAspI) is inherently parallel and can be adjusted by modifying the drptol and lfill parameters.

A. Generic Approximate Sparse Inverse smoothing

Let us consider the ILU(0), [1], factorization of a matrix A,

$$\mathrm{A=LU+E} \tag{4}$$

where E is the error matrix and L and U are the lower and upper factors of the matrix A, retaining the same profile. In order to compute the GenAspI matrix a sparsity pattern must be known a priori. The approximate inverse sparsity pattern is computed through Powers of Sparsified Matrices (PSMs) of the filtered version of the coefficient matrix, [6,7]. Let the sparsified version of the matrix A be described as follows,

$$\tilde{A}_{\mathrm{ij}}=\begin{cases}1,\mathrm{i==j\ and}\ \left|\left(D^{-1/2}\mathrm{AD}^{-1/2}\right)_{\mathrm{i,j}}\right|>\mathrm{drptol}\\ 0,\mathrm{otherwise}\end{cases} \tag{5}$$

where D is a diagonal matrix such that

$$D_{\mathrm{ii}}=\begin{cases}|A_{\mathrm{ii}}|,|A_{\mathrm{ii}}|>0\\ 1,\mathrm{otherwise}\end{cases} \tag{6}$$

and drptol is the so called drop tolerance, [6,7]. The coefficient matrix A is sparsified with the process described by equation (5), which denotes the normalization of each element with the diagonal element, [6,7]. Then each element is compared in absolute value against a given drop tolerance in order to withhold or discard the element. The sparsification process is succeeded by the augmentation of the sparsity pattern by raising it to powers denoted by lfill, [6,7]. The remaining nonzero elements of the coefficient matrix represent the strong connections (neighbors) of each element, by considering the equivalent graph of the matrix, [6,7]. The k–th row of the ℓ –th level sparsity pattern $\tilde{\mathrm{A}}^{\ell}$ can be computed as $\tilde{\mathrm{A}}^{\ell}_{\mathrm{k,:}}=\tilde{\mathrm{A}}_{\mathrm{k,:}}\tilde{\mathrm{A}}^{\ell\text{-}1}$, which denotes that the k–th row of the $\tilde{\mathrm{A}}^{\ell}$ pattern is composed by "fusing" the non–zero indices of columns in the rows of $\tilde{\mathrm{A}}^{\ell-1}$ corresponding to the nonzero columns of the k–th row of the sparsified coefficient matrix $\tilde{A}$, [6,7]. By increasing the levels of fill the approximate inverse tends to the exact inverse of the linear system. The algorithm for the computation of an approximate inverse sparsity pattern is given in [6,7,8]. More information concerning the approximate inverse sparsity patterns can be found in [6,7].

The GenAspI matrix is computed, according to the sparsity pattern defined by the previous procedure, by solving recursively the following system, [8],

$$UM^{lfill}_{drptol}=I\ \ \text{and}\ \ M^{lfill}_{drptol}L=0 \tag{7}$$

where L and U are the lower and upper triangular factors computed by the ILU(0), [1], factorization, (4). The GenAspI algorithm has been presented in [8]. The complexity of the GenAspI algorithm in terms of its nonzero elements and its order can be shown to be $\approx(3/4)\left(nnz(M)^2/n\right)-(3/8)\left(nnz(M)\right)+(5/8)n$ multiplications and $\approx(3/4)\left(nnz(M)^2/n\right)-(3/2)\left(nnz(M)\right)+(3/4)n$ additions, [8].

The GenAspI matrix could be used in conjunction with the iterative scheme (3) in order to derive a parametric parallel smoother. The proposed smoother is inherently parallel because the smoothing procedure is limited to matrix – vector multiplication while complex orderings and parallel algorithmic schemes are avoided. In order for a smoother to be effective the smoothing property must be satisfied, [4, 15, 16, 17, 21, 23]. The smoothing property has been proven for the Optimized Banded Generalized Approximate Inverse (OBGAIM) matrix, [9]. Equivalently, the smoothing property can be proven for the GenAspI matrix. Moreover, sharp estimates for the convergence of multigrid algorithms, for generalized smoothing, have been proven by Bank and Douglas in [2]. Furthermore, Hackbusch has proven the optimal values for the damping parameter for various iterative schemes and various PDEs, [15, 16, 17]. It can be observed that the resulting scheme requires a damping parameter ω, which cannot be defined optimally for every problem. In order to tackle the problem of the value of the damping parameter, the Dynamic Over/Under Relaxation (DOUR) algorithm is used, [18]. The DOUR scheme is predictor – corrector scheme that dynamically determines the value of ω to ensure convergence of the smoothing scheme.

Let us consider the equivalent expression for the relaxation scheme (3),

$$x_l^{(\mathrm{k+1})}=\mathrm{x}_l^{(k)}+\omega\left(S\left(x_l^{(k)}\right)\text{-}\mathrm{x}_l^{(k)}\right) \tag{8}$$

where $S\left(x_l^{(k)}\right)=\mathrm{x}_l^{(k)}+\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)_l\left(f_l\text{-}\mathrm{A}_l x_l^{(k)}\right)$.

By applying the predictor–corrector scheme, [18], we have

$$\tilde{x}_l^{(k)}=\mathrm{x}_l^{(k)}+\omega\left(S\left(x_l^{(k)}\right)\text{-}\mathrm{x}_l^{(k)}\right) \tag{9}$$

$$x_l^{(\mathrm{k+1})}=\mathrm{x}_l^{(k)}+\kappa\left(\Delta x_l^{(k)}\right),\ \Delta x_l^{(k)}=\tilde{x}_l^{(k)}\text{-}\mathrm{x}_l^{(k)} \tag{10}$$

where

$$\kappa=\frac{\left\langle \Delta x_l^{(k)},\mathrm{b}_l\text{-}\mathrm{A}_l\tilde{x}_l^{(k)}\right\rangle}{\left\langle \Delta x_l^{(k)},\mathrm{A}_l\Delta x_l^{(k)}\right\rangle} \tag{11}$$

From (8), (9), (10) and (11) we obtain

$$x_l^{(\mathrm{k+1})}=\mathrm{x}_l^{(k)}+\omega_e\left(S\left(x_l^{(k)}\right)\text{-}\mathrm{x}_l^{(k)}\right),\ \omega_e=\omega(1+\kappa) \tag{12}$$

where ω_e is the effective relaxation parameter and equation (12) is the proposed iterative scheme, [18]. The equation (12) denotes a two stage non-stationary approximate inverse smoother. Further information and convergence analysis of the DOUR algorithm were given in [18].

B. Transfer Operators

The transfer operators are special operators that are used to transfer vectors from coarser to finer grids and finer to coarser grids. The transfer operators for the multigrid method are the restriction and prolongation operators. The

restriction operator is used to transfer vectors from finer to coarser grids. An effective choice for the restriction operator is the full-weighting, which for the three-dimensional case, [21], can be expressed by the following stencil,

$$R=\frac{1}{64}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}^{2h}_{h}\begin{bmatrix}2&4&2\\4&8&4\\2&4&2\end{bmatrix}^{2h}_{h}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}^{2h}_{h}\right]$$

It can be observed that both for the two-dimensional and three-dimensional case the elements of the coarse vector are the weighted average of their neighbors in the finer grid. The prolongation operator is an interpolation procedure used to transfer vectors from coarser to finer grids. An effective choice for the prolongation operator is the tri-linear interpolation. For three-dimensional problems the tri-linear interpolation can be expressed by the following stencil, [21]

$$\mathrm{P}=\frac{1}{16}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}^{\mathrm{h}}_{2\mathrm{h}}\begin{bmatrix}2&4&2\\4&8&4\\2&4&2\end{bmatrix}^{\mathrm{h}}_{2\mathrm{h}}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}^{\mathrm{h}}_{2\mathrm{h}}\right]$$

The full-weighting operator and the tri-linear interpolation operator are related through the Galerkin condition $[P]=c[R]^T$, thus simplifying the mapping on the data, [4, 21]. Transfer operators occupy about 30% of the total computational work of the multigrid algorithm, thus choosing higher order interpolation schemes may lead to excessive computational cost at no extra gain in the convergence behavior, [4, 21]. The prolongation and restriction operators for the proposed schemes are in sparse matrix representation 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 concerning the transfer operators can be found in [3, 4, 21, 23].

C. Cycle Strategy

The cycle strategy is the last component of the multigrid method. The cycle strategy refers to the sequence in which the grids are visited and the respective corrections are obtained, [4,21,23]. The most commonly used cycle strategy is the V-Cycle where the method descends to coarser level 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.

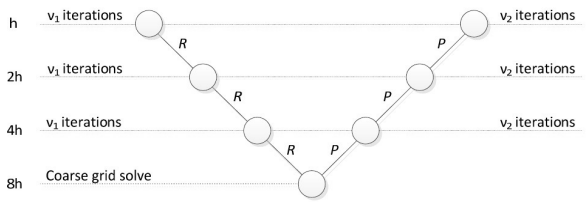

Fig. 2 The multigrid V – Cycle for four levels with ν_1 pre-smoothing iterations and ν_2 post-smoothing iterations.

The V-Cycle is parallelized only in the higher order levels to ensure the efficiency of the scheme, because 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. Thus, levels with many nodes are computed in parallel using multiple threads, however levels with lower order are executed sequentially. As the number of levels is increased the sequential part is occupying lesser 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 is the following, [4,21,23],

$$v^h \leftarrow \mathrm{MGV}\left(A^h,\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^h,\mathrm{v}^h,\mathrm{f}^h,\mathrm{l}\right) \quad (13)$$

if l is the coarsest level

$$\text{relax}\left(A^{\mathrm{lh}},\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^{\mathrm{lh}},\mathrm{v}^{\mathrm{lh}},\mathrm{f}^{\mathrm{lh}}\right)\ \nu_3\ \text{times} \quad (14)$$

else

if order (n) large enough

$$\text{Prelax}\left(A^h,\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^h,\mathrm{v}^h,\mathrm{f}^h\right)\ \nu_1\ \text{times} \quad (15)$$

$$\mathrm{f}^{2\mathrm{h}} \leftarrow \mathrm{Prestrict}\left(\mathrm{f}^{\mathrm{h}}-\mathrm{A}^{\mathrm{h}}\mathrm{v}^{\mathrm{h}}\right) \quad (16)$$

$$\mathrm{v}^{2\mathrm{h}} \leftarrow 0 \quad (17)$$

$$v^{2h} \leftarrow \mathrm{MGV}\left(A^{2h},\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^{2h},\mathrm{v}^{2h},\mathrm{f}^{2h},\mathrm{l}\text{-}1\right) \quad (18)$$

$$v^h \leftarrow v^h + \text{Pprolong}\left(v^h\right) \quad (19)$$

$$\text{Prelax}\left(A^h,\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^h,\mathrm{v}^h,\mathrm{f}^h\right)\ \nu_2\ \text{times} \quad (20)$$

else

$$\text{relax}\left(A^h,\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^h,\mathrm{v}^h,\mathrm{f}^h\right)\ \nu_1\ \text{times} \quad (21)$$

$$f^{2\mathrm{h}} \leftarrow \text{restrict}\left(\mathrm{f}^{\mathrm{h}}-\mathrm{A}^{\mathrm{h}}\mathrm{v}^{\mathrm{h}}\right) \quad (22)$$

$$v^{2h} \leftarrow 0 \quad (23)$$

$$v^{2h} \leftarrow \mathrm{MGV}\left(A^{2h},\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^{2h},\mathrm{v}^{2h},\mathrm{f}^{2h},\mathrm{l}\text{-}1\right) \quad (24)$$

$$v^h \leftarrow v^h + \text{prolong}\left(v^h\right) \quad (25)$$

$$\text{relax}\left(A^h,\left(M^{\mathrm{lfill}}_{\mathrm{drptol}}\right)^h,\mathrm{v}^h,\mathrm{f}^h\right)\ \nu_2\ \text{times} \quad (26)$$

where ν_3 denotes the iterations for the inexact solution on the coarsest level. The prefix "P" denotes the parallel version of the method used. More information concerning the V-Cycle strategy as well as more Cycle schemes can be found in [3,4,21,23].

III. Particle Mesh Method Based On Multigrid Iterative Scheme

The Particle Mesh method (PM), introduced by Hockney in [19], significantly reduces the computational cost of N-body simulation algorithms in cosmological simulations of large scale structure formations. The PM method is based on the computation of the forces of the so-called "superparticles", which are composed by large formations of smaller particles and react together through their gravitational force as a whole.

In the PM method each particle contributes to the density of the concentrated mass along the grid points. Thus, each particle contributes to the neighboring points of the mass by a fraction analogous to the distance of the body from the surrounding nodes. The most commonly known method for adding the contribution of each body to the density of mass in the region is the Nearest Grid Point (NGP), [19]. The NGP method, however, increases the overall error of the PM scheme, because bodies inside a single cell share the same acceleration and force without considering the position of the particle. In order to decrease the computational error introduced during the discretization or the interpolation, the multilinear interpolation is used, namely Cloud in Cell (CIC), [19]. The Cloud In Cell method is used to interpolate the contribution of mass to every nearby grid point and raises significantly the accuracy of the computations. For the computation of the mass density of a particle with mass m_p located at (x_p, y_p, z_p) the Cloud In Cell (CIC) method is used and the respective mass densities for the eight surrounding grid points are given by equations (27), [19].

The grid points where each respectable mass contributes are depicted in Figure 3.

Let us consider the PDE (1) subjected to Dirichlet boundary conditions (1.a), discretized with the Finite Differences method and solved with the parallel GenAspI-MGV method. The solution of the resulting linear system provides the gravity potential in each point of the mesh.

$$\begin{aligned}
\rho_{i,j,k} &= \frac{m_p}{h^6}(h-\Delta_{pi})(h-\Delta_{pj})(h-\Delta_{pk}) \\
\rho_{i,j,k+1} &= \frac{m_p}{h^6}(h-\Delta_{pi})(h-\Delta_{pj})(\Delta_{pk}) \\
\rho_{i,j+1,k} &= \frac{m_p}{h^6}(h-\Delta_{pi})(\Delta_{pj})(h-\Delta_{pk}) \\
\rho_{i,j+1,k+1} &= \frac{m_p}{h^6}(h-\Delta_{pi})(\Delta_{pj})(\Delta_{pk}) \\
\rho_{i+1,j,k} &= \frac{m_p}{h^6}(\Delta_{pi})(h-\Delta_{pj})(h-\Delta_{pk}) \\
\rho_{i+1,j,k+1} &= \frac{m_p}{h^6}(\Delta_{pi})(h-\Delta_{pj})(\Delta_{pk}) \\
\rho_{i+1,j+1,k} &= \frac{m_p}{h^6}(\Delta_{pi})(\Delta_{pj})(h-\Delta_{pk}) \\
\rho_{i+1,j+1,k+1} &= \frac{m_p}{h^6}(\Delta_{pi})(\Delta_{pj})(\Delta_{pk})
\end{aligned} \tag{27}$$

The gravity potential is acquired from the solution of the linear system and the acceleration g is computed by the following equation, viz.

$$g = -\nabla\Phi = -\left(\frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}, \frac{\partial\Phi}{\partial z}\right) \tag{28}$$

The computation of the acceleration in the center of the cells is commencing by using centered differences to retain the accuracy of the computed results. For the x-direction the acceleration can be computed by the following equation,

$$\frac{\partial\Phi}{\partial x} \approx \frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2\mathrm{h}} + O\left(h^2\right) \tag{29}$$

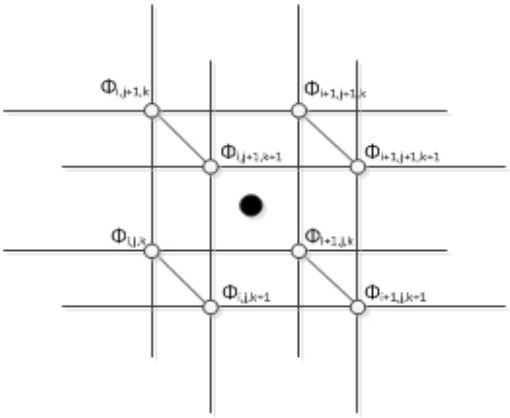

Fig. 3 Neighboring points in a three-dimensional cell of mesh size h.

The acceleration in the centers of the cells is interpolated back to the particles using the CIC method. Applying the CIC method for a particle in the three-dimensional space results in the following formula for the acceleration, [19],

$$\begin{aligned}
g_p = {} & \kappa_1 g_{i,j,k} + \kappa_2 g_{i,j,k+1} + \kappa_3 g_{i,j+1,k} + \kappa_4 g_{i,j+1,k+1} \\
& + \kappa_5 g_{i+1,j,k} + \kappa_6 g_{i+1,j,k+1} + \kappa_7 g_{i+1,j+1,k} \\
& + \kappa_8 g_{i+1,j+1,k+1}
\end{aligned} \tag{30}$$

where

$$\begin{aligned}
\kappa_1 &= (h-\Delta_{pi})(h-\Delta_{pj})(h-\Delta_{pk}) \\
\kappa_2 &= (h-\Delta_{pi})(h-\Delta_{pj})(\Delta_{pk}) \\
\kappa_3 &= (h-\Delta_{pi})(\Delta_{pj})(h-\Delta_{pk}) \\
\kappa_4 &= (h-\Delta_{pi})(\Delta_{pj})(\Delta_{pk}) \\
\kappa_5 &= (\Delta_{pi})(h-\Delta_{pj})(h-\Delta_{pk}) \\
\kappa_6 &= (\Delta_{pi})(h-\Delta_{pj})(\Delta_{pk}) \\
\kappa_7 &= (\Delta_{pi})(\Delta_{pj})(h-\Delta_{pk}) \\
\kappa_8 &= (\Delta_{pi})(\Delta_{pj})(\Delta_{pk})
\end{aligned} \tag{31}$$

while the Δ operator denotes the difference between the two points as denoted by the subscripts.

The acceleration of each mass is subject to weights computed by the distances from the nodal points of the grids. To compute the final displacement of a body, the equations of motion have to be integrated. In this article the velocity Verlet integrator is utilized due to its low computational cost and $O(h^2)$ accuracy, [22]. The Verlet integration of the equations of motion leads to the following equations,

$$\vec{x}(t+\Delta t)=\vec{x}(t)+\vec{u}(t)\Delta t+\frac{1}{2}\vec{a}(t)\Delta t^2 \qquad (32)$$

$$\vec{u}(t+\Delta t)=\vec{u}(t)+\frac{1}{2}\left(\vec{a}(t)+\vec{a}(t+\Delta t)\right)\Delta t \qquad (33)$$

where $\vec{u}, \vec{a}, \vec{x}$ are the three-dimensional vectors of the velocity, the acceleration and the position, respectively. This process is repeated according to the chosen time step and the maximum time of the simulation. It should be noted that particles cannot escape from the domain because the region is forced to be toroidal. The parallelization of the method is simplified and based on loop level parallelism using OpenMP, because the proposed schemes are mainly composed of matrix-vector multiplications and vector-vector computations. Further information concerning the Particle Mesh method can be found in [19].

IV. Numerical Results

In this section the applicability and performance of the proposed scheme is demonstrated by simulating the 3D spaces with different numbers of particles.

The numerical tests were performed on a Dual Socket AMD Opteron Processor 6128 HE, with 16GB RAM, running Ubuntu Linux 12.04.1.

The domain chosen for the simulations was the unit cube. The time step for the method was set to 0.001 and the simulation was executed for 10 consequent time steps.

The 3D region chosen was 10 parsec in each direction and the masses of the bodies were chosen to be 10^{32} kg. These variables were normalized in order to model the system on the unit cube.

The termination criterion was set to $\|r_i\|<1e\text{-}10\|r_0\|$. The pre-smoothing and post-smoothing steps were set to $\nu_1=2$ and $\nu_2=2$. The drop tolerance for the computation of the GenAspI matrices was set to drptol=0.0. The method for lfill=1 presented the fastest performance and required 7 iterations to converge to the desired tolerance. 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.

In Table 1, the overall performance of the designed simulation algorithm based on the GenAspI-MGV method for various numbers of bodies, various numbers of threads and various resolutions, is presented. In Table 2, the speedups of the designed simulation algorithm based on the GenAspI-MGV method for various numbers of bodies, various numbers of threads and various resolutions, are presented. In Table 3, the efficiency of the designed simulation algorithm based on the GenAspI-MGV method for various numbers of bodies, various numbers of threads and various resolutions, is presented.

It should be noted that the preprocessing cost of the designed simulation algorithm concerning the computation of the Generic Approximate Sparse Inverses for resolutions h=1/16 and h=1/32 was t_{pre}=0.030093 seconds and t_{pre}=0.248670 seconds respectively, which is several orders less than the computational time needed for the model problems rendering the method efficient by slightly enlarging the computational cost.

V. Conclusion

The proposed schemes were proven experimentally effective for various choices of bodies and resolutions. Additionally, it should be stated that the proposed scheme presented satisfactory scalability as the number of particles increases and resolution is refined. Research efforts are under way to improve the parallel performance of the method with new hybrid schemes. Moreover, new computational schemes that will enhance convergence behavior and accuracy of the simulation are under further research.

Acknowledgment

The authors would like to express their thanks to Prof. K. G. Margaritis, Parallel Distributed Processing Laboratory, Department of Applied Informatics, University of Macedonia, for the provision of computational facilities.

References

[1] O. Axelsson, *Iterative solution methods.* Cambridge University Press, 1996.

[2] 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, pp. 617-633, 1985.

[3] A. Brandt, "Multi-level adaptive solutions to boundary-value problems", *Math. Comp.,* vol. 31, pp. 333-390, 1977.

[4] L.W. Briggs, V.E. Henson and F.S. McCormick, *A multigrid tutorial.* SIAM, 2000.

[5] 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(4), pp. 1396–1417, 2001.

[6] E. Chow, "A priori sparsity patterns for parallel sparse approximate inverse preconditioners", *SIAM J. Sci. Comput.*, vol. 21, pp. 1804-1822, 2000.

[7] E. Chow, "Parallel implementation and practical use of sparse approximate inverses with a priori sparsity patterns", *Int. J. High Perf. Comput. Appl.*, vol. 15, pp. 56-74, 2001.

[8] C.K. Filelis-Papadopoulos and G.A. Gravvanis, "Generic Approximate Sparse Inverse Matrix Techniques", Report *TR/ECE/ASC-AMA/2012/14*, submitted.

[9] 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(3), pp. 233-253, 2013.

[10] P. Frederickson, "High performance parallel multigrid algorithms for unsctructured grids", *In 1996 Seventh Copper Mountain Conference on Multigrid Methods* CP 3339, NASA, pp. 317-326.

[11] G.A. Gravvanis, "High Performance Inverse Preconditioning", *Archives of Computational Methods in Engineering*, vol. 16(1), pp. 77-108, 2009

[12] G.A. Gravvanis, "Explicit Approximate Inverse Preconditioning Techniques", *Archives of Computational Methods in Engineering*, vol. 9(4), pp. 371-402, 2002.

[13] G. A. Gravvanis, "The rate of convergence of explicit approximate inverse preconditioning", *Inter. J. Comp. Math.*, vol. 60, pp. 77-89, 1996.

[14] G. A. Gravvanis, C. K. Filelis-Papadopoulos and P. I. Matskanidis, "Algebraic multigrid methods based on Generic Approximate Matrix Techniques", Report *TR/ECE/ASC-AMA/2012/13*, submitted.
[15] W. Hackbusch, "Multi-grid convergence theory". *In 1982 Lecture Notes in Mathematics 960*, Springer, pp. 177-219.
[16] W. Hackbusch, "On the convergence of multi-grid iterations", *Numer. Math,* vol. 9, pp. 213-239, 1981.
[17] W. Hackbusch, "Convergence of a multi-grid iteration applied to difference equations", *Math. Comp.*, vol. 34, pp. 425-440, 1980
[18] R. Haelterman, J. Viederndeels and D. Van Heule, "Non–stationary two–stage relaxation based on the principle of aggregation multi-grid", *In 2006 Computational Fluid Dynamics 2006*, Part 3, Springer, pp. 243–248.
[19] R.W. Hockney and J.W. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, 1981.
[20] 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", *Math. and Computers in Simulation*, vol. 29, pp. 1-17, 1987.
[21] U. Trottenberg, C.W. Osterlee and A. Schuller, *Multigrain*. Academic Press, 2000.
[22] J. Verlet, "Computer Experiments on Classical Fluids", *In Physical Review,* vol. 159(1), pp. 98–103, 1967.
[23] P. Wesseling, "Theoretical and practical aspects of a multigrid method", *SIAM J. Sci. Stat. Comput.*, vol. 3, pp. 387-407, 1982.
[24] P. Wesseling, "The rate of convergence of a multiple grid method", Numerical analysis*, In 1980 Proceedings of the 8th bienn. Conference in Lect. Notes Math.,* 773, pp. 164-184.

TABLE I.
THE OVERALL PERFORMANCE OF THE DESIGNED SIMULATION ALGORITHM BASED ON THE GENASPI-MGV METHOD FOR VARIOUS NUMBERS OF BODIES, VARIOUS NUMBERS OF THREADS AND VARIOUS RESOLUTIONS.

N	Threads	h=1/16	h=1/32
10^5	**1**	1.260010	3.182530
	2	0.874061	2.036820
	4	0.534355	1.139530
	8	0.330208	0.722727
	16	0.269090	0.646396
10^6	**1**	10.821700	12.841400
	2	6.224200	7.336500
	4	3.630050	4.041050
	8	2.325110	2.350280
	16	1.488090	1.605920
10^7	**1**	110.123000	119.286000
	2	60.767680	65.667200
	4	34.057200	34.706500
	8	19.708600	19.180160
	16	12.293300	11.013100

TABLE II.
THE SPEEDUPS OF THE DESIGNED SIMULATION ALGORITHM BASED ON THE GENASPI-MGV METHOD FOR VARIOUS NUMBERS OF BODIES, VARIOUS NUMBERS OF THREADS AND VARIOUS RESOLUTIONS.

N	Threads	h=1/16	h=1/32
10^5	**2**	1.44	1.56
	4	2.36	2.79
	8	3.82	4.40
	16	4.68	4.92
10^6	**2**	1.74	1.75
	4	2.98	3.18
	8	4.65	5.46
	16	7.27	8.00
10^7	**2**	1.81	1.82
	4	3.23	3.44
	8	5.59	6.22
	16	8.96	10.83

TABLE III.
THE EFFICIENCY OF THE DESIGNED SIMULATION ALGORITHM BASED ON THE GENASPI-MGV METHOD FOR VARIOUS NUMBERS OF BODIES, VARIOUS NUMBERS OF THREADS AND VARIOUS RESOLUTIONS.

N	Threads	h=1/16	h=1/32
10^5	**2**	0.72	0.78
	4	0.59	0.70
	8	0.48	0.55
	16	0.29	0.31
10^6	**2**	0.87	0.88
	4	0.75	0.79
	8	0.58	0.68
	16	0.45	0.50
10^7	**2**	0.91	0.91
	4	0.81	0.86
	8	0.70	0.78
	16	0.56	0.68