J Supercomput (2012) 62:1537–1559
DOI 10.1007/s11227-012-0816-4

Automatic dynamics simplification in Fast Multipole Method: application to large flocking systems

Seyed Naser Razavi · Nicolas Gaud · Abderrafiâa Koukam · Nasser Mozayani

Published online: 16 August 2012

Abstract This paper introduces a novel framework with the ability to adjust simulation's accuracy level dynamically for simplifying the dynamics computation of large particle systems to improve simulation speed. Our new approach follows the overall structure of the well-known Fast Multipole Method (FMM) coming from computational physics. The main difference is that another level of simplification has been introduced by combining the concept of motion levels of detail from computer graphics with the FMM. This enables us to have more control on the FMM execution time and thus to trade accuracy for efficiency whenever possible. At each simulation cycle, the motion levels of detail are updated and the appropriate ones are chosen adaptively to reduce computational costs. The proposed framework has been tested on the simulation of a large dynamical flocking system. The preliminary results show a significant complexity reduction without any remarkable loss in the visual appearance of the simulation, indicating the potential use of the proposed model in more realistic situations such as crowd simulation.

S.N. Razavi (✉) · N. Gaud · A. Koukam · N. Mozayani
SCOMAS Lab, Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
e-mail: razavi@iust.ac.ir

N. Mozayani
e-mail: mozayani@iust.ac.ir

S.N. Razavi · N. Gaud · A. Koukam · N. Mozayani
Multi-Agent Systems and Applications Group, Laboratoire Systèmes et Transports, UTBM, 90010 Belfort, France
url: http://www.multiagent.fr

N. Gaud
e-mail: nicolas.gaud@utbm.fr

A. Koukam
e-mail: abder.koukam@utbm.fr

Keywords Dynamics simplification · Fast Multipole Method · Multi-agent based simulation · Flocking

1 Introduction

In recent years, simulation of large complex systems has attracted many researchers from a growing number of diverse areas and it is appealing to more and more scientific domains such as sociology, biology, physics, chemistry, ecology, economy, etc. In many cases, simulation of a complex system requires to compute all the pairwise interactions between particles. Assuming N particles, there are totally $O(N^2)$ pairwise interactions which result in a computational cost of $O(N^2)$, which is clearly prohibitive for large values of N. That is, computing the pairwise interactions between particles is the main bottleneck in the simulation of many large complex systems. The challenge of efficiently carrying out the related calculations is generally known as the N-*body* problem.

Among several attempts done to reduce this complexity, we can refer to the Fast Multipole Method (FMM) as one of the most successful ones. The FMM is an algorithm originally proposed by Rokhlin [1] as a fast scheme for accelerating the numerical solution of the Laplace equation in two dimensions. It was further improved by Greengard and Rokhlin when it was applied to particle simulations [2, 3], and has since been identified as one of the ten most important algorithmic contributions in the 20th century [4]. It evaluates pairwise interactions in large ensembles of N particles in $O(N \log N)$ or even $O(N)$ time. This is an improvement over the $O(N^2)$ time required by direct methods. Since the invention of the FMM, it has been widely used for problems arising in diverse areas (molecular dynamics, astrophysics, acoustics, fluid mechanics, scattered data interpolation, etc.) because of its ability to achieve linear time and memory in computing dense matrix vector products to a fixed user-prescribed accuracy ε.

The main idea used in the FMM is that it considers well-separated or "far-away" groups of particles as one particle. To implement this idea, it imposes a hierarchical spatial partitioning structure on the computational domains. Based on our previous works [5, 6], we believe that we can take advantage of this hierarchical structure in several ways to further reduce the overall computational cost, of course at the expense of accuracy. In particular, the main goal of this work is to combine the FMM with simulation acceleration techniques from computer graphics to improve simulation speed for N-body simulations while controlling the desired level of accuracy. Simulation acceleration techniques including motion levels of detail often need some hierarchical partitioning of the computational space for dynamics simplification and fortunately this is available in the FMM.

In fact, the FMM algorithm clusters particles into hierarchical groups based on their relative positions in the domain, but it deals with each particle in the same group separately, computing a different potential for every particle in the group resulting in different behaviors for different particles in the same group. However, particle systems generally exhibit a high level spatial coherence, meaning nearby particles behave in approximately the same way. Therefore, the main idea is that we can compute only one approximated behavior for a group, then making all particles in the group to follow the same behavior. This is achieved by replacing particles inside a cluster by one weighted particle and approximating the motion models of those particles in the force computations during FMM.

Although there exist simulation acceleration techniques such as given in [7–10], there is no known algorithm for the *automatic dynamics simplification* of complex physical or biological systems especially when all the pairwise interactions between particles are needed. In this paper, we limit the scope of our investigation to particle systems as a first step towards the design of automatic dynamics simplification. Particle systems are commonly used in computer graphics, physically based modeling, and animation of natural phenomena and group behavior. Additionally, they are often the building blocks of highly complex dynamical systems. Therefore, we hope the results of this paper can lead to the generalization of dynamics simplification for a broader class of dynamical systems (e.g. pedestrians and crowds simulation).

As the preliminary results confirm, this automatic simplification results in a substantial performance improvement, but it introduces a new source of inaccuracy as well. This is a natural trade-off between accuracy and efficiency common in the field of simulation and shared with other simplification techniques. However, to keep the error of the FMM below a user-prescribed level ε, which is an essential component of the FMM, we need to be very careful about choosing when and which group's behavior can be simplified. Therefore, in addition to a hierarchy of approximate motion models, we need a mechanism enabling us to automatically switch between these different levels of details. It is important that any switching should be very smooth so that it does not cause a noticeable change in the flow of the simulation. The creation of the motion levels of detail and the switching mechanism are discussed in Sect. 5.

1.1 Contributions

For dynamics simplification, a mechanism is needed for generating a hierarchy of approximated motion models or motion levels of detail in order to perform automatic dynamics simplification of a particle system. Here, we have decided to use the same *hierarchical physically based subdivision* scheme in the FMM. This way we can take advantage of the approximate computations used in the FMM to reduce the complexity beside our automatic dynamics simplification. Also, a similarity measure is required to compare particles or groups of particles to generate approximated motion models and to select between them. For now, we have used the velocity vector angle ratio and speed ratio for this purpose, both common in the field. However, it seems rational to investigate other possibilities such as entropy from information theory or Boltzmann partition function, but we leave this investigation to another occasion in the future.

To perform automatic dynamics simplification, the original FMM is modified as following: During the upward pass of the FMM, the similarity measures are computed for each box in the FMM tree. We then approximate those boxes or groups of boxes which have satisfied the similarity measure as one weighted particle in the potential computations, and apply the same result to all particles inside those boxes or groups of boxes. This multilevel dynamics simplification can result in a more efficient computation of pairwise interactions while maintaining the total error below a user-prescribed error level, resulting in no loss or minimal loss in the visual appearance and the correctness of the simulation. The complete algorithm is given in Sect. 5.

Therefore, at each simulation step the tree is updated based on changing simulation requirements as defined by the user or by the nature of simulation. Appropriate motion levels of detail are adaptively generated based on this subdivision given the requirements for different regions of the simulation and a desired execution time for each step. We have implemented a prototype system that can automatically generate simplified motion models, select appropriate models, and switch between them seamlessly. Then, the proposed framework is applied to the dynamic simulation of a large ensemble of flocking agents, which is a common example for N-body systems. The preliminary results are very promising, indicating a significant reduction in the complexity of the simulation while maintaining its correctness, and thus the potential to generalize the framework to the dynamic simulation of more systems.

1.2 Outline

The rest of this paper is organized as follows. Section 2 gives a brief review of some related works regarding simulation acceleration techniques and motion levels of detail. Section 3 presents an introduction to the FMM which is used to solve our target application in this paper and reviews some of the key concepts relevant to our proposed model. Section 4 describes the flocking model and its interesting properties which have made this model a well-suited candidate for automatic dynamics simplification. Section 5 describes the proposed framework, including the automatic generation of motion levels of detail and the mechanism used to select appropriate models and to switch between them adaptively. Some experimental results are given in Sect. 6. In particular, it presents the results of our prototype implementation, and compares its performance against the traditional FMM. Finally, we conclude with some future research guidelines and perspectives in Sect. 7.

2 Literature review

In this section, various attempts related to level of details done in the field of interactive computer graphics are summarized. To achieve *dynamic realism* in computer graphics, often the polygonal geometry of small or distant portions of the model is simplified to reduce the rendering cost without a significant loss in the visual content of the scene. Therefore, the goal of polygonal simplification in rendering is to reduce the complexity of a polygonal model to a level that can be rendered at interactive rates. In an offline preprocessing step, multiple versions of each object are created at progressively coarser *levels of detail*, or LODs. Once the LODs have been created and stored for every object in the model, complexity can be regulated at run-time by choosing for each frame which LOD will represent each object. As an object grows more and more distant, the system switches to coarser and coarser LODs. This type of simplification is called geometrical simplification or graphical levels of detail. It is important to distinguish it from simulation level of detail.

Simulation levels of detail or motion levels of detail techniques have been proposed for reducing the computational cost of the dynamic simulation of the character motion. Motion models can be generated from pre-recorded motion sequences, procedural approaches, kinematics, or based on dynamics computation. Some of the earlier human motion models in computer animation exploited this concept implicitly by using procedurally generated motion, simplified dynamics and control algorithms, off-line motion mapping, or motion playback [11–13].

Carlson and Hodgins applied simulation LOD techniques to a graphical environment populated with multiple, physically simulated one-legged robots [7]. Three simulation LOD models represent varying animation qualities: a point-mass model with no animated degrees of freedom, a point-mass model with cinematically animated degrees of freedom, and a fully simulated version. Higher-quality LODs are reserved for characters at very dynamic moments, as when avoiding or experiencing collisions and those near the viewer in the field of view. They demonstrated that a group of characters that dynamically switches between LODs can sufficiently replicate the performance of a fully simulated group. In this work, the generation of simulation LODs, switching and selection are designed by hand for a group of legged creatures. However, their experimental results are indicative of the potential of automatic simplification of general dynamical systems.

By using the space–time constraint dynamics formulation, Popovic and Witkin introduced a motion transformation technique that preserves the essential properties of animated character motion with drastically lesser number of degrees of freedom [10]. Multon et al. suggested a series of simplified walking models for mobilizing on complex terrain, as well as how and when the transition takes place [14].

Other types of simulation acceleration techniques have also been investigated to reduce the total computational simulation costs for a large, complex dynamical system. Chenney and Forsyth proposed view-dependent culling of dynamic systems to speed up the computation of dynamics by ignoring what is not visible to the viewer [8], similar to view culling. Faloutsos et al. developed a system that accomplishes complex tasks by evaluating multiple controllers automatically and selecting an appropriate sequence [15]. The authors use an online machine learning technique to build a database that models the effectiveness of each controller for a character in a given state. Run-time queries of the database identify controllers that best suit the current state of the character and its desired final state. The database of controllers is similar to our simulation LODs because it supports the evaluation, comparison, and selection of controllers. However, the simulation LODs store more controllers and switch between them more frequently.

Additional graphics researchers are investigating ways to simplify physical simulations to reduce computation costs. Grzeszczuk et al. developed a technique that uses neural network approximations to emulate physically simulated characters' equations of motion [9]. After the neural network is trained to sufficiently model the original system, it can produce motions more efficiently than a full dynamic simulation. O'Sullivan and Dingliana investigate the opportunities to replace simulated particles with simplified counterparts when imperceptible to the viewer [16]. In particular, they find inaccurate dynamics are less noticeable in the peripheral vision and when the movements are complex. O'Brien et al. describe a method to automatically simplify particle simulations through clustering into spatially localized groups [17].

It is very important to notice that the practitioners of interactive computer graphics are concerned more with finding better and faster ways to approximate the simulation results than with accurately simulating the physics of the system. This is very different from a situation like ours in which the accuracy of the simulation is paramount. However, if we pay careful attention, we will be able to apply the same ideas to these kinds of applications to reduce the time complexity while preserving accuracy. This is a major feature distinguishing this work from those in the field of computer graphics. Therefore, the work presented in this paper may be considered as a link connecting the rich body of literature on real-time computer graphics with those in N-body simulations. In fact, there are many useful ideas in interactive computer graphics, which can be applied as well to the simulation of large N-body systems to reduce time complexity.

3 An introduction to the Fast Multipole Method

Consider the problem of simulating the evolution of stars in a galaxy under gravitational forces, or of ions in a medium under electrostatic forces. These problems and many other similar ones need to compute the interactions among a system of bodies or particles and are known as N-body problems. In many of these problems the long-range interactions between bodies cannot be ignored; however, there is a nice property which forms the basis for many approximation algorithms including the Fast Multipole Method (FMM) [18]: the magnitude of interactions falls off with distance between the interacting bodies. The hierarchically structured FMM is a very efficient, accurate, and hence very promising algorithm for solving such problems and not surprisingly it has been applied successfully to a wide variety of applications, such as computational astronomy, molecular dynamics, fluid dynamics, radar scattering, etc.

According to the distribution of the particles in the computational domain, there are two main versions of the FMM: the uniform FMM which works very well when the particles in the domain are uniformly distributed, and the so-called adaptive FMM which is the method of choice when the distribution of particles in the domain is highly non-uniform, resulting in a significant saving of both time and space. However, comparing to the uniform FMM, the adaptive one is a much more complicated algorithm both to implement and to understand.

Our example N-body application studies the evolution of a large flocking system of particles acting under the influence of pairwise interactions, interactions with surrounding obstacles, and group objectives (see Sect. 4.2). It is a classical N-body simulation in which every body exerts forces on all others. The simulation proceeds over a large number of time-steps, every time-step computing the net force on every particle and updating its position and other attributes.

The most time-consuming phase in every time-step is by far the computation of the pairwise interactions among all the particles in the system. The simplest method to do this is to compute these pairwise interactions between particles directly. It is clear that this direct method has an $\mathrm{O}(N^2)$ time complexity (with N the number of particles), which is prohibitive for large N. Hierarchical, tree-based methods have therefore been developed that reduce the complexity to $\mathrm{O}(N \log N)$ for general distributions or even to O(N) for uniform distributions, while still maintaining a high degree of accuracy [2]. They do this by exploiting a fundamental insight into the physics of most systems that N-body problems simulate, an insight that was first provided by Isaac Newton in 1687 A.D.: Since the magnitude of interaction between particles falls off rapidly with distance, the effect of a large group of particles may be approximated by a single equivalent particle, if the group of particles is far enough away from the point at which the effect is being evaluated.

The most widely used and promising hierarchical N-body methods are the Barnes–Hut [19] and the Fast Multipole methods [2]. The FMM is more complex to program than the Barnes–Hut method, but provides better control over error and has better asymptotic complexity, particularly for uniform distributions (although the constant factors in the complexity expressions are larger for the FMM than for Barnes–Hut in three-dimensional simulations). In addition to classical N-body problems, the FMM and its variants are used to solve important problems in domains ranging from fluid dynamics to numerical complex analysis, and have inspired breakthrough methods in domains as seemingly unrelated as radiosity calculations in 3D computer graphics [20].

As discussed earlier, several versions of the FMM have been proposed, the simplest one being the two-dimensional uniform algorithm. This is itself far more complex to program than the Barnes–Hut method, but is considerably simpler than the adaptive two-dimensional version and the three-dimensional versions. Since we are interested in non-uniform distributions, and since the principles are very similar for the two and three-dimensional cases, we use the adaptive two-dimensional FMM in this paper. The following section provides a brief review of the adaptive FMM.

3.1 The adaptive Fast Multipole Method in 2D

FMM achieves its performance by introducing a hierarchical partition of a bounding square D, enclosing all particles, and two series expansions for each box at each level of the hierarchy. More precisely, the root of the tree is associated with the square D and referred to as level 0. The boxes (squares) at level $l+1$ are obtained recursively, subdividing each box at level l into four squares, referred to as its children. The tree is constructed so that the leaves contain no more than a certain fixed number of particles (say s). Generally, we refer to s as the *clustering size* of the tree. For non-uniform distributions, this leads to a potentially unbalanced tree, as shown in Fig. 1 (which assumes $s=1$). This tree is the main data structure used by the FMM.

A key concept in understanding the algorithm is that of *well-separatedness*. A point or box is said to be well-separated from a box B if it lies outside the domain of B and B's *neighbors* (neighbors are defined as the boxes having at least one common vertex to B). In Fig. 2, B's neighbors are marked with a "U" label. Using this concept, the FMM translates Newton's insight into the following: If a point P is well-separated from a box B, then B can be represented by a multipole expansion about its center as far as P is concerned.

For example, if you stand on the Earth and look at the sky, you will see the Andromeda (a galaxy approximately 2.5 million light-years away containing about one trillions of stars) as a single star. So, it makes sense to consider it as a single point

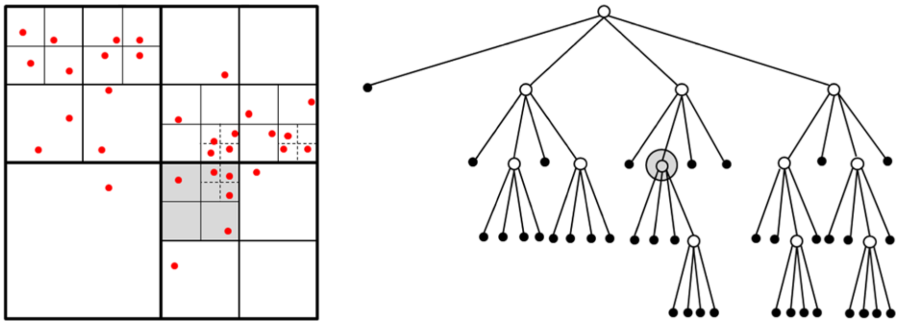
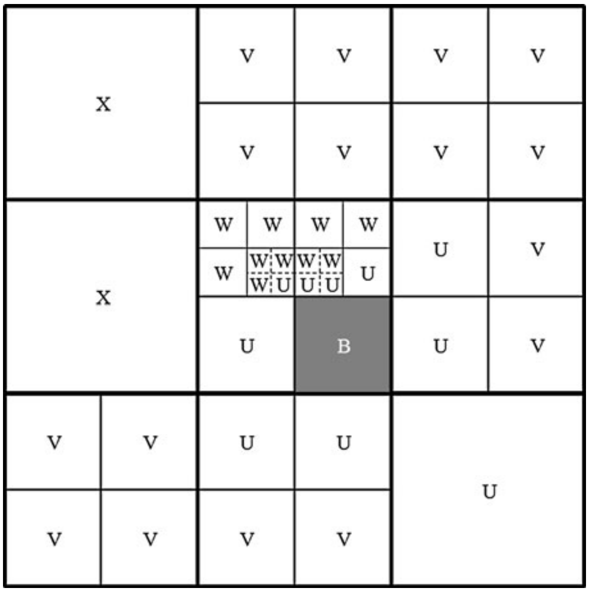


Fig. 1 A 2D particle distribution (*left*) and its corresponding quadtree (*right*)

Fig. 2 Different lists associated with box B in the FMM



when computing the interaction between Earth and Andromeda. The same approximation is taken in the FMM. To compute the force exerted on P due to the particles inside B, given P is far enough from box B, FMM simply evaluates B's *multipole expansion* at P, rather than computing the forces due to each particle inside B separately. The same multipole expansion, computed once, can be evaluated at several points, thus saving a substantial amount of computation.

Thus, the multipole expansion for a box B encodes the potential due to the particles inside it to the *far-field*. The multipole expansion of a box is a series expansion of the properties of the particles within it (expansions of non-leaf boxes are computed from the expansions of their children). If we use an infinite number of terms in the expansion, then the multipole expansion is an exact representation. But in practice we can only use a finite number of terms, say p. This truncation number p determines the accuracy of the representation and as a result the accuracy of the whole method.

Truncating the number of terms in the series is a desired aspect in the FMM. In fact, this truncation serves as a mechanism to trade accuracy for efficiency, which is the main reason justifying the use of an approximation method instead of an exact one.

Turning back to our example, if you look at the Earth from Andromeda, then you will see the Milky Way galaxy as a single point. Again, exploiting this idea will result in a tremendous amount of saving in the computations. This is parallel to the so-called *local expansion* in the FMM. More precisely, representing boxes by their multipole expansions is not the only insight exploited by the FMM. If a point P (be it a particle or the center of a box) is well-separated from box B, then the effects of P on particles inside B can also be represented as a Taylor series or local expansion about the center of B, which can then be evaluated at the particles inside B. Once again, the effects of several such points can be converted just once each and accumulated into B's local expansion, which is then propagated down to B's descendants and evaluated at every particle within B. Thus, the local expansion for a box B encodes the potentials on the particles inside it due to the sources in the far-field of B. The mathematics of computing multipole expansions, translating them to local expansions, and shifting both multipole and local expansions are described in [18].

Now, we have almost all the necessary background to give a brief description of the FMM. But before that, the different lists associated with each box B in the adaptive tree of the FMM algorithm must be defined. These lists contain all boxes whose contributions need to be processed by B itself when we use an adaptive way to partition the computational domain. Contributions from more distant boxes are considered by B's ancestors using local-to-local (L2L) translations. Some of these lists are defined only for leaf boxes of the tree, while others are defined for internal boxes as well. The lists for a leaf box B are described in Fig. 2, and their role is discussed in more detail in [3, 18]:

- For a leaf box, L_B^U contains B and its adjacent boxes. If B is not a leaf box, L_B^U will be empty. Since the boxes in L_B^U are not well-separated from B, their contribution on B should be computed directly.
- L_B^V contains the set of the children of the neighbors of the parent of B, which are not adjacent to B. The interaction from a box $v \in L_B^V$ to B is computed using multipole-to-multipole (M2M) translations since v and B are well-separated.
- For a leaf box, L_B^W includes all the descendants of B's neighbors, which are not adjacent to B but their parents are adjacent to B. Since B is in the far range of $w \in L_B^W$, the contribution from w to B is evaluated using multipole expansion of w. Again, if B is not a leaf box, L_B^W will be empty.
- Finally, if box B is in the list L_A^W, then L_B^X contains box A. The contribution from $x \in L_B^X$ to B is evaluated directly since B is in the near-range of x.

3.2 Overall structure of the FMM

Using the lists described in the previous section, the adaptive FMM proceeds in the following steps:

STEP 1: Space partitioning The tree is built by loading particles into an initially empty root box and then this root box is partitioned recursively until there are no more than s particles in each box. Also, for each box B, the lists L_B^U, L_B^V and L_B^W are constructed explicitly. As L_B^X is the dual of L_B^W, it is not constructed.

STEP 2: Upward Pass Starting from leaf boxes, the multipole expansions of all boxes are computed in an upward pass through the tree. Expansions of leaf boxes are computed from the particles inside them, and expansions of internal boxes from those of their children using multipole-to-multipole (M2M) translations as shown in Fig. 3.

STEP 3: Downward Pass Starting from level 2 in the tree, the local expansions of internal boxes are computed and then recursively propagated down to the leaf boxes in the tree:

3.1. For each box B, the contribution from its parent's far-field is computed by translating the local expansion of B's parent to the center of B (the L2L translation in Fig. 3).

3.2. The multipole expansions of all boxes in the L_B^V list are translated and accumulated into local expansions about the center of B (the M2L translation in Fig. 3).

3.3. If B is a leaf, the multipole expansions of the boxes in L_B^W are evaluated at the particles in B. Since the L_B^X is the dual of the L_B^W and does not need to be constructed explicitly, L_B^X interactions are computed at the same time as the L_B^W list interactions. That is, for every box $w \in L_B^W$, B first computes the contribution from w to the particles inside it and updates the forces on its own particles accordingly, and then computes the L_B^X list interaction and updates the local expansion of w. Since L_B^X interactions are thus computed by leaf boxes, internal boxes compute only their L_B^V interactions.

STEP 4: Final summation In this step, for a leaf box B, the force exerted on each particle inside B is computed by combining the far-field interactions (Step 4.1) and the near-field interactions (Step 4.2). This is done in two steps:

4.1. The resulting local expansion of B (obtained from the downward pass) is evaluated at each particle inside it.

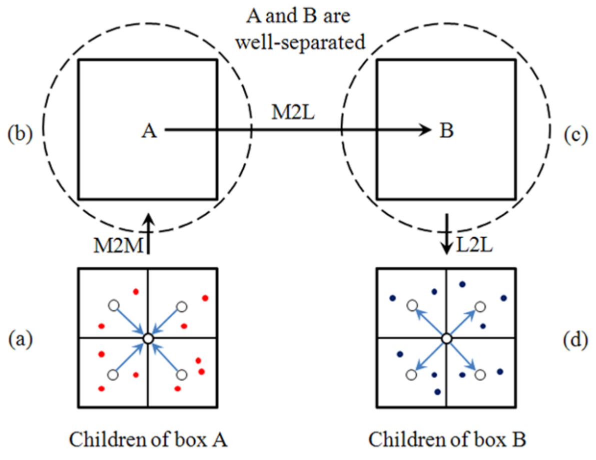


Fig. 3 The Fast Multipole Algorithm mechanisms: (**a**), (**b**) after spatial decomposition, the child boxes use the Multipole to Multipole translation to shift their multipole expansion to the center of the parent box; (**c**) using the Multipole to Local translation, well-separated boxes interact by creating a local expansion at the center of box B due to box A; (**d**) the children of box B feel the potential of box A by using the Local to Local translation to shift the parent's local expansion

4.2. The interactions between all particles in the L_B^U with all particle inside B are computed directly.

Our example application iterates over several hundred time-steps, every time-step executing the above steps as well as one more that updates the velocities and positions of the particles at the end of the time-step. For the problems we have run, almost all the execution time is spent in computing list interactions. The majority of this time (about 60–70 %) is spent in computing V list interactions, next U list (about 20–30 %) and finally the W and X lists (about 10 %). Building the tree and updating the particle properties take less than 1 % of the time in sequential implementations.

3.3 Complexity of the adaptive FMM

Assuming that there are N particles and each box contains at most s particles, there will be about N/s boxes in the tree. Also, each box B has at most 29 boxes in its L_B^V list and 9 boxes in its L_B^U list. Therefore, the total operations count in the adaptive FMM is approximately

$$Np + 29(N/s)p^2 + Np + 9Ns$$

The terms correspond to formation of multipole expansions, translations, evaluation of the local expansions at particles, and direct computation of the near-neighbor interactions. By choosing s almost equal to p, the total operation count will be $40Np$ or simply $O(Np)$. While p is a small constant comparing to N, the adaptive FMM is a linear time algorithm and is applicable for very large number of particles.

It is worth to notice that the richer analytic structure of the FMM permits a large number of modifications and optimizations, which are not available to other hierarchical schemes. These schemes have to do with the use of symmetry relations to reduce the number of shifts [3] as well as schemes which reduce the cost of translation itself [21]. Section 5 discusses how we can use simplified motion models to reduce FMM computational costs for simulation of a large complex system.

4 Flocking

Flocking behavior is the behavior exhibited when a group of birds, called a *flock*, are foraging or in flight. There are similar behaviors in other groups of animals or insects like the shoaling behavior of a group of fishes, the swarming behavior of bees, and the herd behavior of land animals [22]. From the perspective of the mathematical modeler, flocking is a collective motion of a large number of self-propelled entities and is a collective animal behavior exhibited by many living beings such as birds, fish, bacteria, and insects [23]. It is considered an emergent behavior arising from simple rules that are followed by individuals and does not involve any central coordination. Flocking behavior was first simulated on a computer in 1986, when Reynolds introduced the first three heuristic rules for flocking behaviors [24]:

1. Separation—avoid crowding neighbors;
2. Alignment—steer towards average heading of neighbors; and

3. Cohesion—steer towards average positions of neighbors.

Among the physics-inspired theoretical approaches studying flocking particles systems, we may mention the work of [25] that mainly focuses on the study of alignment rules and associated emergent phenomena. References [26] and [27] propose continuum models to study alignment and swarming. In [28], a model based on artificial potential function (APF) in a network with fixed or dynamic topologies is presented. It proposes a centralized algorithm for particle systems that leads to irregular collapse for generic initial states. It also introduces a distributed version that leads to irregular fragmentation (disintegration of a flock into smaller groups combined with violation of inter-agent constraints). Fragmentation and collapse are two well-known pitfalls of flocking algorithms most likely occurring for generic set of initial states and large number of agents [29–32].

Here, we have used a flocking model based on the theoretical framework for design and analysis of distributed flocking algorithms from Refs. [30, 33]. This model may be considered as a modified version of APF in [28] trying to avoid traditional pitfalls of flocking. Reference [34] provides a definition of "flocking" for particle systems that is independent of the method of trajectory generation for particles. In this sense, it has the same role as "Lyapunov stability" for nonlinear dynamical systems. This model is based on a systematic method for construction of collective potential functions for flocking. The flocking behavior is accomplished by a set of physical laws governing the pairwise interactions between flocking agents and also their interactions with the environment.

In this model, a lattice-type structure is used to model the geometry of desired conformation of agents in a flock. In such a conformation, each agent should be equally distanced from all of its neighbors while moving towards its objective that may be static or dynamic. As shown in Fig. 4, this lattice-type structure exhibits a high degree of spatial order among agents and this is exactly where we can use simplified motion models to reduce complexity. Using the terminology of information theory, order means redundancy and the redundancy can be removed from the system with no loss or minimal loss in the information contents. This is an important reason motivating us to use this flocking system as our target application in this work. This type of order exists more or less in many systems and so the work presented in this paper may be considered as a general framework to reduce complexity in simulating those systems.

As flocking occurs in the presence of multiple obstacles, each agent in the flocking model is equipped with obstacle avoidance capability. In fact, each agent has interactions with three kinds of objects in the environment:

- Nearby flockmates (α-agents),
- Obstacles (β-agents), and
- A virtual leader (γ-agent).

Each agent in the flock applies a control input that consists of three different terms which are discussed in the following sections:

$$u_i = u_i^{\alpha} + u_i^{\beta} + u_i^{\gamma}$$

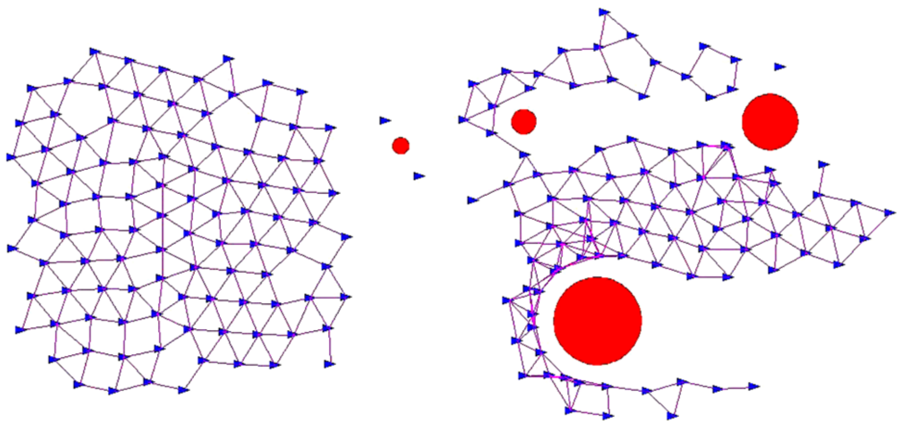


Fig. 4 Lattice-type conformations for a group of flocking agents (*left*) and associated obstacle avoidance

4.1 Interaction with nearby flocking agents

Following the basic rules of Reynolds, each agent in the flock should coordinate itself with nearby flockmates in terms of position and velocity. In this model, an interaction range r is defined between two agents. The neighbors of a flocking agent i are defined to be the set of all nearby flockmates with a distance less than or equal to r. During flocking, agent i tries to maintain an equal distance from its neighbors and also to match its velocity with theirs. The term u_i^α in the control input is responsible for coordinating agent i with its neighbors to form the lattice-type structure as illustrated in the left part of Fig. 4.

4.2 Interaction with the virtual leader

Coordinating each agent with its neighbors is not sufficient to produce a real flocking behavior as it results in a diverging group of agents. In real situations, all agents in the flock have to move towards the same destination (the location of food or the destination of migration). In this model, the target position is represented by a virtual leader (γ-agent). A γ-agent has the role of a virtual leader in charge of navigation and control of the behavior of a flock as a whole. It is a mechanism that provides a common objective for a group of flocking agents. This objective may be static (fixed position) or dynamic (moving object). The term u_i^γ in the control input manages this leader-tracking procedure; it may be considered as a navigational feedback.

4.3 Interaction with surrounding obstacles

Beside interactions with nearby agents and tracking the virtual leader, each agent in the flock has the ability to avoid obstacles. Obstacle avoidance is achieved by introducing a third type of agent called β-agent. Whenever an agent is in close proximity of an obstacle, a β-agent will be induced on the border of that obstacle. The interaction between the α-agent and β-agent produces a short-range impulse which prevents

α-agent to further approach the obstacle and hence helps it to avoid that obstacle (see the right part of Fig. 4). The term u_i^β in the control input accounts for all of the interactions between agent i and its nearby obstacles. For more details on the exact definitions of u_i^α, u_i^β and u_i^γ the reader may refer to [30].

5 Framework for automatic dynamics simplification

This section presents our proposed method for automatic dynamics simplification with application in the simulation of large dynamical systems. Since the proposed method follows the overall structure of the FMM (see Sect. 3.2), this section mainly focuses on the differences rather than the similarities. We start first by presenting the details about computing the *Center of Mass* (COM) particle for each box or group of boxes. We then follow the discussion by introducing the criteria used in the new algorithm to determine when and which boxes in the FMM tree may be merged to simplify calculations.

Before giving the details of the framework, it should be mentioned that a variant of the FMM, called kernel-independent FMM, is used to implement our flocking systems [5]. The kernel-independent FMM follows the overall structure of the FMM but it does not require the analytical expansions (local and multipole expansions) of the kernel and only relies on direct evaluations of the kernel. As it is not obvious to derive these expansions for many complex kernels, using the kernel-independent method results in more flexibility and hence the framework presented here is a general framework applicable to a wide variety of N-body systems.

5.1 Approximating motion models

For each box (a leaf box or an internal box in the FMM tree), dynamics simplification is achieved by approximating its particles dynamics using the particles dynamics of its center of mass. Given a box B consisting of k particles, the corresponding approximated motion model is computed using the following procedure:

1. Compute the position P_{COM} and velocity V_{COM} of the center of mass particle using:

$$P_{\mathrm{COM}} = \frac{\sum_{i=1}^{k} m_i P_i}{\sum_{i=1}^{k} m_i}$$

$$V_{\mathrm{COM}} = \frac{\sum_{i=1}^{k} m_i V_i}{\sum_{i=1}^{k} m_i}$$

   where m_i, P_i and V_i represent the mass, position and velocity of the ith particle inside B. Please note that in a different application m_i could represent some other quantity. For example in Coulombic systems, it represents the charge of ith ion in a medium rather than its mass.
2. Using the standard computations in the FMM, first compute the potential of the COM for box B and then update its velocity and position accordingly.
3. Apply the same results computed for the COM to all particles inside B.

5.2 Pruning the FMM tree

As discussed earlier, we need a mechanism to automatically decide for which boxes we can use approximated motion models. From now on, we refer to such boxes as *simplified boxes*. This is a key factor determining the overall success or failure of the automatic dynamics simplification.

For a given box B, we use the following similarity measures to decide if we can use dynamics simplification for that box: the *speed ratio* and the *velocity vector angle ratio*. These computations differ for a leaf box and an internal box:

- If B is a leaf box, we compute these two measures for B from the velocities of its particles. That is, the relative speed of its particles and also their relative angle should be within certain limits.
- Otherwise, if B has children, the similarity measures for B are computed by considering the COM particles of its children. Again the relative speed of the COM particles and their relative angle are constrained to be within certain limits.

The above computations are done during the upward pass of the FMM algorithm (see Sect. 3.2). If box B satisfies these two conditions, then all of its particles are replaced by its weighted center of mass particle which is computed directly from its particles or indirectly from the center of mass of its children. After this simplification, the new algorithm continues as the classical FMM. However, when the potentials have been computed for every particle (a real particle or a weighted center of mass particle), the computed potentials for a COM particle are applied to all of the particles inside the corresponding simplified box. A snapshot of the developed system for 200 particles is given in Fig. 5. Because of dynamics simplification, the number of particles is reduced from 200 to 89 in this system resulting in a considerable reduction in the execution time.

5.3 Automatic switching between different levels of simulation

Additionally, similarly to geometric simplifications used in computer graphics in which special areas have higher priorities (for example, the areas closer to the viewer and more centered in the viewing frustum) and hence should be simulated in more detail, some events in our framework may require higher resolution to maintain simulation correctness and integrity. Referring to Sect. 4.3, recall that the agents (particles) in our flocking system have the ability to avoid obstacles in their path during flight. This obstacle avoidance capability requires higher priority comparing to the other behaviors of flocking agents. Fortunately, our framework has the ability to cope with these critical situations.

For example, when a large box consisting of a large number of agents is near an obstacle, the simulation automatically switches to a higher resolution level. That is, the box automatically decomposes into its children (if it is a non-leaf box) or into its individual particles (if it is a leaf box). This is a recursive process which continues until reaching a level with a good enough accuracy. As the particles move away from the obstacle, physical properties and spatial positions of those with similar motion may again be regrouped into a box, or the children boxes may be regrouped into their

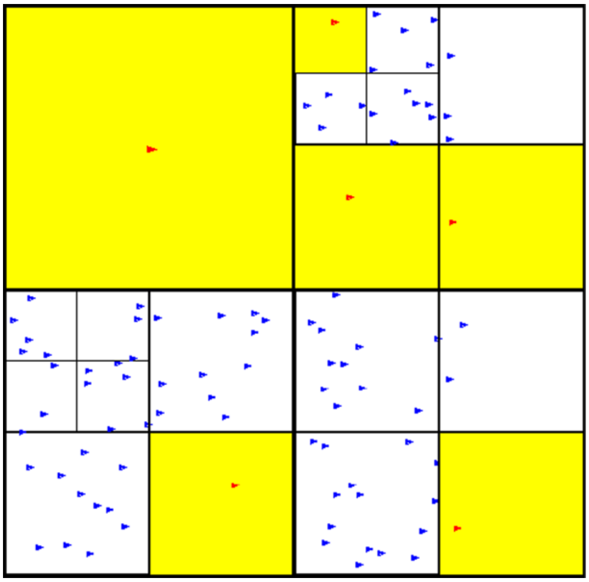


Fig. 5 A snapshot of the developed system with 200 agents. Each colored box represents a box whose particles dynamics are simplified using dynamics of its COM particle (the *red particle* in it)

parent box by ascending in the hierarchical tree. It is very important to notice that all these switches among different levels are done automatically without any interference from the user.

5.4 The complete algorithm

Now we are ready to give the complete algorithm. The complete algorithm is given in Listing 1. As mentioned earlier, the new algorithm follows the overall structure of the original FMM. However, this algorithm is capable to perform automatic dynamics simplification (ADS) based on the measures introduced in Sect. 5.2.

5.5 Real-time simulations

The new algorithm presented in this paper can be used for real-time simulations of large N-body systems, in which efficiency is much more important than accuracy as well as the visual appearance of the simulation is acceptable to an end-user. Given a target execution time and a target fidelity requirements defined by the user, the algorithm uses the physically based subdivision of the entire system produced in the first step of the FMM to generate the hierarchical motion levels of detail. As the simulation proceeds, the hierarchy and hence the motion levels of detail are updated and the user can see the results graphically.

During the simulation, it is possible to monitor the execution time of the last few steps and compare it to the target. If the execution speed is too slow, the system performs more simplifications by enlarging the *maximum cluster size*. This is achieved by allowing larger boxes (those that are located at higher levels in the hierarchical tree) to be merged. The inverse occurs when the speed is more than necessary. By

Listing 1 FMM with automatic dynamics simplification

```
STEP 1—CONSTRUCT TREE $T$ AND LISTS
  build $T$ such that each leaf $B$ contains at most $s$ points
  **for** each box $B$ in *preorder* traversal of $T$ **do**
     build list of nearest neighbors, $L_N^B$ and interaction list, $L_I^B$
  **end for**

STEP 2—UPWARD PASS
  **for** each box $B$ in *postorder* traversal of $T$ **do**
      **if** $B$ is a leaf box **then**
         construct multipole expansion $a_k\{0 \le k \le p\}$, from all source points using *S2M*
         compute the similarity measures for $B$
         **if** $B$ satisfies the similarity measures **then**
            compute COM position and velocity from particles inside $B$
            replace all particles inside $B$ with its COM particle
      **else**
         construct multipole expansion $a_k\{0 \le k \le p\}$ from each of $B$'s children using *M2M*
         compute the similarity measures for $B$
         **if** $B$ satisfies the similarity measures **then**
            compute the COM position and velocity from the COM of $B$'s children
            replace all particles inside $B$'s children with its COM particle
            remove $B$'s children from the tree                              //pruning of the tree
      **end if**
  **end for**

STEP 3—DOWNWARD PASS
  **for** each box $B$ in *preorder* traversal of $T$ **do**
     compute the $B$'s local expansion from its parent's local expansion using the L2L operator
     add to the $B$'s local expansion the contribution from $L_I^B$ list using the M2L operator
  **end for**

STEP 4—FINAL SUMMATION
  **for** each leaf box $B$ in $T$ **do**
      **for** each target location $y$ in $B$ **do**
         Add to the potential of y the contribution from $B$'s local expansion
         Add to the potential of y the contributions from $L_N^B$ using direct calculations
  **end for**
  **for** each simplified box $B$ **do**
     apply the potential computed for COM agent on every target in $B$
```

constraining the size of boxes which can be merged, the simulation runs more accurately while taking more time to be executed. All of these can be performed by introducing a new parameter called *simplification level*. For a given simplification level, only boxes below that level are allowed to be merged. A small value for this parameter means that boxes at higher levels are allowed to be merged and a large value means that only small boxes at lower levels are allowed to be merged. Therefore, each time we need a faster execution, the simplification level is decreased by one and this allows larger boxes in higher levels to be merged. Increasing this parameter does the inverse.

In summary, the proposed framework has several parameters which enable us to control both the error and the execution time of the simulation. These parameters are: the speed ratio and the velocity vector angle ratio, the maximum number of particles in each box and the maximum cluster size or the simplification level. The set of these parameters gives us the potential to apply our framework for real-time applications as well. Future work is already planned to test this real-time capability of the proposed framework.

6 Experimental results

We have implemented a prototype system in Java which can automatically generate simulation levels of detail, select appropriate ones and switch between them. In this section we present the results of our system tested on a large dynamical flocking system which is a classic example for N-body systems (see Sect. 4). All experiments were conducted on a desktop computer configured with one 2.5-GHz Quad-Core Intel Pentium processor and 4-GB RAM running a Windows 7 Professional x32 Edition. Our application was developed in Java and the Java VM used to execute the tests was configured with 1-GB memory.

6.1 Simulation of a large dynamical flocking system

This experiment compares the execution time of the proposed method with the ability to dynamically adjust the levels of motion to the original FMM. The results are shown in Fig. 6 and the corresponding numerical results are reported in Table 1. For each number of agents, the parameters of the proposed method have been set in order to keep the error below a pre-specified level ε. The appropriate values for similarity measures to achieve this are determined by try and error. Also, all the results reported in Table 1 are computed by averaging the execution times over 1000 consecutive runs. To keep the results comparable, we have fixed the truncation number p to 10 and the clustering size s to 100 in all experiments.

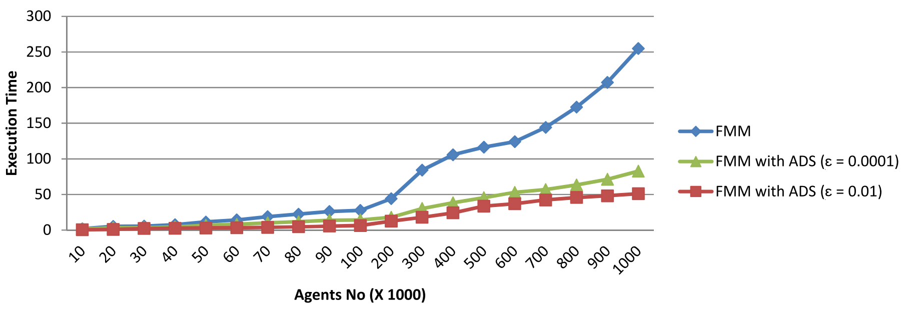


Fig. 6 Execution times for the FMM method with automatic dynamics simplification and the original FMM

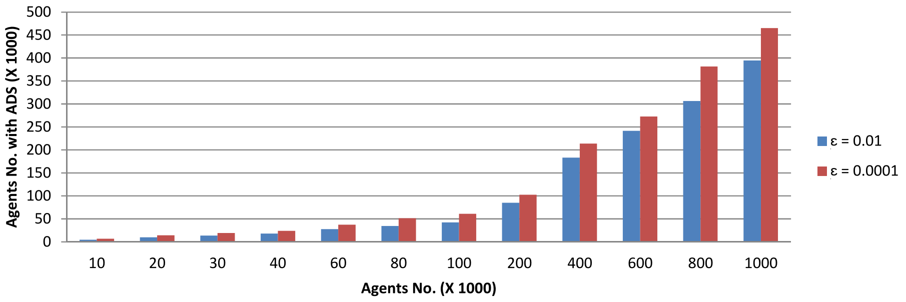


Fig. 7 Number of agents after automatic dynamics simplification

As shown in Fig. 6, the experimental results confirm the theoretical expectation that to obtain more accuracy we need to spend more time. Conversely, more computational time can be saved if the algorithm is allowed to be run less accurately. For example, the FMM with automatic dynamics simplification is nearly five times faster than the original FMM while the error is less than 10^{-2}. Of course, it is possible to achieve more efficiency by relaxing accuracy requirements. This trade-off between accuracy and efficiency enables us to cope with very large and complex systems which would be intractable otherwise. For a better comparison, Fig. 7 shows the number of agents after dynamics simplification for each of the above experiments. From this figure we can see that this factor is approximately $1/2$ for $\varepsilon = 10^{-4}$ and $1/3$ for $\varepsilon = 10^{-2}$. The corresponding numerical results are reported in Table 2.

7 Summary and future work

In this paper we present a new framework by combining the well-known Fast Multipole Method from computational physics with the concept of motion levels of detail from computer graphics and animations. Our goal is to reduce the computational complexity of the FMM by adding automatic dynamics simplification to it and hence to speed up the simulation of large N-body systems consisting of hundreds of thousands or even millions of bodies. Dynamics simplification is achieved by first replacing all particles inside a box with one weighted center of mass particle and then applying the potential computed for the COM particle to all the particles in that box.

This replacement of particles can be done for a box at the bottom level of the FMM tree (i.e. a leaf box) or boxes at higher levels in the tree. This way we have a hierarchy of approximated motion models or motion levels of detail which are updated at each simulation cycle. These motion levels of detail enable us to adjust the level of simulation dynamically and hence to trade accuracy for efficiency. Here, we implemented a prototype of our model and applied it on a large dynamical flocking systems. As can be seen in Sect. 6, the preliminary results are very satisfactory indicating the potential use of our method to more dynamical systems.

However, the work presented in this paper is a first step towards the design of a general multilevel simulation framework with the ability to automatically adjust the simulation level with the hope of managing a good compromise between accuracy of the simulation and execution time. The work can be extended in several dimensions:

- Investigating other possible methods or heuristics for similarity measure instead of speed ratio and velocity vector angle ratio. For example, one possibility is to use the entropy related to the particles inside a box as a measure to decide switching between levels of motion. Also, it is possible to use partition function and energy-based measures for this purpose.
- Designing new experiments to gain a better understanding of the effects of each parameter such as truncation number p and clustering size s on the execution time of the simulation and its accuracy.
- Generalizing the proposed framework to be applicable in more dynamical systems, such as simulation of large crowds or simulation of a large number of pedestrians in urban areas.
- Giving a more complete complexity analysis and error analysis and studying the effects of main parameters such as truncation number and clustering size on the complexity and the error of the new method.
- Designing new test to study the functionality of the proposed algorithm for real-time simulation of large N-body systems. This can be achieved through the analysis of specific constraints such as running the simulation within a fixed predetermined amount of time and the way to minimize the error in these kinds of configurations.

Appendix: Numerical results

Table 1 Computational results for FMM and FMM with ADS ($p = 10, s = 100$)

No. of agents	Execution time (ms)		
	FMM	FMM with automatic dynamics simplification	
		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-4}$
10 000	2012	671	1682
20 000	5310	1061	3466
30 000	5653	2339	3547
40 000	7768	2668	4605
50 000	11481	3062	6833
60 000	14498	3432	8084
70 000	18966	3926	10299
80 000	22398	4909	11834
90 000	26298	5797	13934
100 000	27783	6505	14521
200 000	44231	12719	18044
300 000	84245	18142	30394
400 000	105726	24292	38371
500 000	116406	33675	45655
600 000	124134	37112	52983
700 000	144123	42354	57070
800 000	172635	45893	63502
900 000	207303	48213	71417
1000 000	254816	51126	82789

Table 2 Number of agents after automatic dynamics simplification

No. of agents	FMM with automatic dynamics simplification	
	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-4}$
10 000	4793	6934
20 000	9876	14251
40 000	18176	24214
60 000	27718	37686
80 000	34328	51496
100 000	42119	61307
200 000	84921	102651
400 000	183362	213793
600 000	241402	273151
800 000	306570	381463
1000 000	394537	465121

References

1. Rokhlin V (1983) Rapid solution of integral equations of classical potential theory. J Comput Phys 60(2):187–207
2. Greengard L, Rokhlin V (1987) A fast algorithm for particle simulations. J Comput Phys 73(2):325–348
3. Greengard L, Rokhlin V (1988) Rapid evaluation of potential fields in three dimensions. In: Lecture notes in mathematics, vol 1360. Springer, Berlin, pp 121–141
4. Dongarra J, Sullivan F (2000) The top ten algorithms of the century. Comput Sci Eng 2(1):22–23
5. Razavi SN, Gaud N, Mozayani N, Koukam A (2011) Multi-agent based simulations using fast multipole method: application to large scale simulations of flocking dynamical systems. Artif Intell Rev 35(1):53–72
6. Razavi SN, Gaud N, Koukam A, Mozayani N (2011) Using motion levels of detail in the fast multipole method for simulation of large particle systems. In: WMSCI 2011, Orlando
7. Carlson DA, Hodgins JK (1997) Simulation levels of detail for real-time animation. In: Graphic interface, pp 1–8
8. Chenney S, Forsyth D (1997) View-dependent culling of dynamic systems in virtual environments. In: ACM symposium on interactive 3D graphics, New York
9. Grzeszczuk R, Terzopoulos D, Hinton G (1998) Neuroanimator: fast neural network emulation and control of physics-based models. In: SIGGRAPH, New York, pp 9–29
10. Popovic Z, Witkin A (1999) Physically based motion transformation. In: SIGGRAPH, New York, pp 11–20
11. Brudlerlin A, Calvert TW (1996) Knowledge-driven, interactive animation of human running. In: Graphics interface, pp 213–221
12. Granieri JP, Crabtree J, Badler NI (1995) Production and playback of human figure motion for 3d virtual environments. In: VRAIS, pp 127–135
13. Perlin K (1995) Real time responsive animation with personality. IEEE Trans Vis Comput Graph 1(1):5–15
14. Multon F, Valton B, Jouin B, Cozot R (1999) Motion levels of detail for real-time virtual worlds. In: ASTC-VR'99
15. Faloutsos P, van de Panne M, Terzopoulos D (2001) Composable controllers for physics-based character animation. In: SIGGRAPH 2001, New York, pp 251–260
16. O'Sullivan C, Dingliana J (2001) Collisions and perception. ACM Trans Graph 20(3)
17. O'Brien D, Fisher S, Lin MC (2001) Automatic simplification of particle system dynamics. In: Computer animation, Seoul, pp 210–257
18. Greengard LF (1987) The rapid evaluation of potential fields in Particle systems. Yale University, New Haven, PhD Thesis
19. Barnes JE, Hut P (1986) A hierarchical O($N \log N$) force calculation algorithm. Nature 324(6096):446–449
20. Hanrahan P, Salzman D, Aupperle L (1991) A rapid hierarchical radiosity algorithm. In: SIGGRAPH, New York, pp 197–206
21. Elliott WD, Board JA (1996) Fast Fourier transform accelerated fast multipole algorithm. SIAM J Sci Comput 17(2):398–415
22. Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31(1):61–85
23. O'loan OJ, Evans MR (1999) Alternating steady state in one-dimensional flocking. J Phys, A Math Gen 32(8)
24. Reynolds CW (1987) Flocks, herds, and schools: a distributed behavioral model. Comput Graph 21:25–34
25. Shimoyama N, Sugawara K, Mizuguchi T, Hayakawa Y, Sano M (1996) Collective motion in a system of motile elements. Phys Rev Lett 76(20):3870–3873
26. Mogilner A, Edelstein-Keshet L (1999) A non-local model for a swarm. J Math Biol 38(6):534–570
27. Toner J, Tu Y (1998) Flocks, herds, and schools: a quantitative theory of flocking. Phys Rev E 58(4):4828–4858
28. Tanner HG, Jadbabaie A, Pappas GJ (2003) Stable flocking of mobile agents, part II: dynamic topology. In: 42nd IEEE conference on decision and control, Maui, Hawaii, pp 2016–2021
29. Zhou J, Yu W, Wu X, Small M, Lu JA (2009) Flocking of multi-agent dynamical systems based on pseudo-leader. arXiv:0905.1037v1 [nlin.CD]

30. Olfati-Saber R (2006) Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Trans Autom Control 51(3):401–420
31. Liu H, Fang H, Mao Y, Cao H, Jia R (2010) Distributed flocking control and obstacle avoidance for multi-agent systems. In: Control conference, Beijing, pp 4536–4541
32. Mousavi MSR, Khaghani M, Vossoughi G (2010) Collision avoidance with obstacles in flocking for multi agent systems. In: Industrial electronics, control & robotics (IECR), Orissa, pp 1–5
33. Olfat-Saber R, Murray RM (2003) Flocking with obstacle avoidance: cooperation with limited communication in mobile networks. In: 42nd IEEE conference on in decision and control, Maui, Hawaii, pp 2022–2028
34. Olfati-Saber R, Murray RM (2003) Consensus protocols for networks of dynamic agents. In: American control conference, Denver, pp 951–956