Fakulteta za strojništvo
Univerza v Ljubljani

Podiplomski študij, 1. seminar

PIC simulations and a gridless treecode method - review

(PIC simulacije in brezmrežna metoda “treecode” - pregled)

Avtor:
Janez Krek

Mentor:
prof. Jože Duhovnik

Ljubljana, 30.10.2009

Contents

1 Introduction

Plasma modeling is an attempt to describe a physical device and processes inside plasma with mathematical terms [3],[1],[6]. The mathematical model is later transferred via algorithms to various computer programs which are used to simulate processes inside plasma. Computer simulations are efficient design tool for providing accurate performance predictions in plasma physics applications. They base on various models, which describe plasma and processes in plasma in different ways and are divided into two main areas: fluid and kinetic models. Fluid plasma description represents macroscopic model of representation of plasma, where kinetic model try to model plasma from opposite side - to describe motion of single particle in plasma.

Particle-in-Cell (PIC) programs (codes) employ kinetic model of plasma description as base for simulation. PIC techniques grew from electron trajectory simulation in 1950s [3],[9] and were developed since then: first with employing more that few particles, then to formalize PIC method, adding collisions and using objected-oriented technology for programming [10]. Development of personal computers and easy access to them also helped in development of PIC codes - almost everybody could have a PC and could do their own PIC simulation, without any need on waiting on computer time in computer centers. Despite today's modern and powerful computers, PIC simulations still take a long time to run. Models are describing more and more processes inside plasma and required computer resources are increasing. Currently, PIC simulations are applied to wide set of problems [9],[8]: plasma display panel, fluorescent lamp, low-T processing plasmas, high power microwave devices, magnetic fusion, heavy ion fusion, space plasmas, basic plasmas, etc.

Treecode method represent a way to reduce number of necessary calculations when calculating influences (forces) between particles in plasma. Because number of particles in plasma is usually high (in order of 10^{22} particles or 10^{17} super particles), number of necessary calculations is high and is in range of N^2 (where N is number of particles) [2]. With using treecode method, number of necessary calculations can be reduced to range of $N \log N$. Another advantage of treecode is the fact that is grid-less - method does not relay on fixed (or defined) grid. This means that we have more tree nodes in areas with high particle densities and less nodes in other areas.

Why we should implement treecode method into PIC simulation program, when calculating forces between particles is only part of PIC simulation loop? Because we are running simulations with large number of particles, difference in number of calculations between N^2 and $N \log N$ is huge and creating tree takes less than that difference. Some testing have been done [4] [5], but currently not showing promising side of treecode method.

2 Computer simulations

Computer simulations are efficient design tool for providing accurate performance predictions in plasma physics applications. Computer simulations base on various models, which describe plasma and processes in plasma in different ways. The models generally describe the conservation of mass, energy, charge and also transformations among chemical species. Equations for such models can be derived fundamentally from the general Boltzmann equation describing probability distributions of individual species in velocity space, subject to collisions and external forces and from the set of Maxwell equations describing electro-magnetic field interactions.

Models for computer plasma simulations can be divided into several different ways, including the model dimensionality (complexity), use of kinetic or fluid approach to describe the plasma-governing equations, which processes model takes into account and others. Computer simulations of plasma can be divided into two main areas, based on how they describe plasma [3]:

- fluid description: MHD model, wave equations
- kinetic description: Vlasov, Fokker-Planck codes; particle codes

Two main descriptions differ in a way they describe plasma and how they are trying to approach problems in plasma description. Many modern simulations programs incorporate both fluid and kinetic model in plasma simulation. Hybrid models may treat electrons as fluid, using fluid model to simulate them, and treat ions as particles, using kinetic model for simulation of ions.

2.1 Fluid plasma models

Fluid plasma description represents macroscopic model of representation of plasma. Model is closer to experiments as kinetic model, because it describes what can be observed from outside. Similar as in experiment, where we observe what is happening with plasma but we do no pay attentions for all small details and processes inside plasma. Aim of fluid model is to represent plasma independent of what happens on molecular level - one assumption is that velocities of particles inside volume element can be neglected. With this, fluid variables are functions of position and time. This model is closer to experiments as in experiment we seldom make measurements at microscopic level. It is similar to fluid representation of neutral gas and fluid, extended to include specific behaviour of plasma and processes inside plasma. Fluid (or continuum) plasma models reduce the computational complexity by averaging velocity space effects.

In many cases, plasma could be modelled as one-fluid with *magnetohydrodynamic (MHD) equations*. The fundamental assumption is that fields and fluid fluctuate on the same time and length scale - on scale of slower and heavier ions. In more advanced models, to support rapid wave fluctuations in plasma (due to the fact that electrons are much faster than ions), plasma is treated as two-fluid in *plasma wave equatons*. Like MHD equations, these are also macroscopic equations, but the assumption under them is quite different.

In fluid model of plasma description, one tries to numerically solve magnetohydrodynamics (MHD) equations of plasma, assuming transport coefficients. The fluid (or continuum) equations consist of three main equations, which describe the conservation of particles, momentum and energy for particular particle species. These equations work with quantities that are averages over all particle velocities in a small volume element. Plasma fluid calculations are often valid even through the main free path of the plasma particle is larger than the volume element of the fluid.

2.2 Kinetic plasma models

Kinetic models try to describe the plasma from opposite way than fluid models. Kinetic models describe plasma from microscopic level, starting with description of motion of a single particle. Because they are dealing directly with various particles in plasma they are potentially the most power full models for studying of processes in plasma and basic equation can become quite complex and difficult to solve. With large number of particles, usually present in plasma, kinetic models require huge computational resources regarding memory requirements, CPU power and also in computational time.

For kinetic models to better describe plasma, all physical properties of plasma are defined in six-dimensional space (positions and velocities) and time. Instead of defining density of particles at given position and time, we define *distribution function* that is defined in ($\vec{r}$, $\vec{v}$) and time.

In kinetic description one is trying to solve particle interactions through electromagnetic field. This can be done by numerically solve kinetic equations (Vlasov, Fokker-Planck equations) or by "particle" simulation in which one computes motion of plasma particles (PIC codes).

2.3 PIC simulations computer programs

Particle-in-Cell (PIC) simulation programs employ kinetic model of plasma description as a base for simulation. PIC techniques grew from electron trajectory simulation in 1950s. Early codes employed only few particles, but later in 1960s, PIC scheme were developed in Universities and national laboratories that employed thousands of particles and usually one-dimensional systems. Main focus was on validating physical and numerical models. Between 1960 and 1980 [3], self-consistent PIC method was formalized and put into computer code. Theoretical limitation and methods to overcome those limitations were developed and described more formally. In 1980s, first device models and Monte-Carlo collisions were developed. Those models were in 1990s extended with self-consistent circuit model and improved Monte-Carlo model. PIC codes (computer codes) were developed using object-oriented techniques and for using on parallel computers.

Despite today's modern and power full computers, PIC simulations still take a long time to run. Typical simulation time for a single iteration of common simulation case are in a range of 1 - 10 seconds. This may seem a very short time, but considering that we need up to 1,5 million iterations to achieve "stady state", total simulation times can be up to 50, 60 days.

All PIC simulation applications (programs, computer codes) that base on PIC techniques have similar main loop - loop that form basics of program/application. Basic steps in every PIC simulation code are shown in figure 1.

In addition to these basic steps, PIC simulation codes often offers the possibility to simulate collisions inside plasma, which is quite important, if simulation is performed with high density plasma (where collisions between particles are more likely to occur).

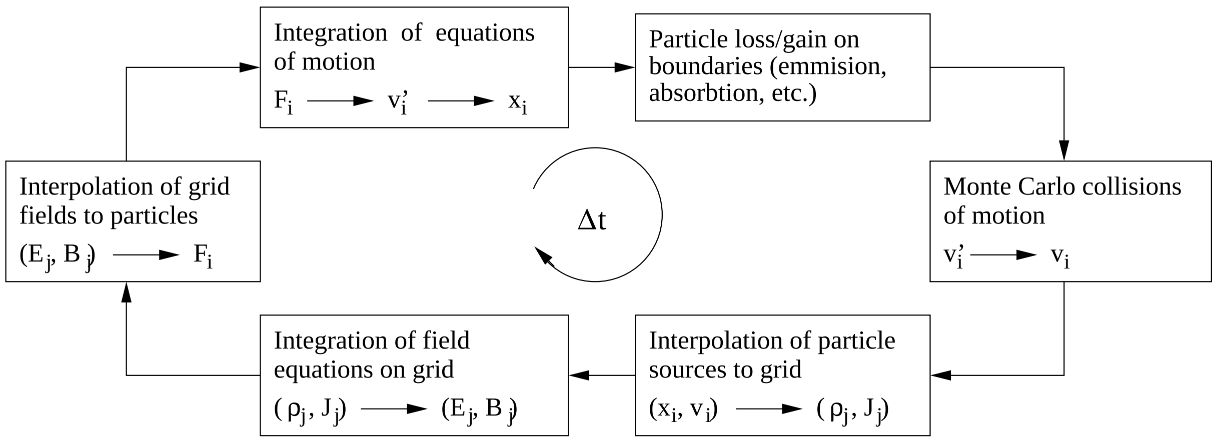
Integration of equations of motion
$F_i \longrightarrow v_i' \longrightarrow x_i$
Particle loss/gain on boundaries (emmision, absorbtion, etc.)
Monte Carlo collisions of motion
$v_i' \longrightarrow v_i$
Interpolation of particle sources to grid
$(x_i, v_i) \longrightarrow (\rho_j, J_j)$
Integration of field equations on grid
$(\rho_j, J_j) \longrightarrow (E_j, B_j)$
Interpolation of grid fields to particles
$(E_j, B_j) \longrightarrow F_i$
Δt

Figure 1: Simple main loop for PIC simulations

3 Treecode (TC) method

Treecode (TC) method is based on treecode algorithm, which was originally developed to compute gravitational forces in astrophysics [2], and was later also used and developed in details in fields of fluid and molecular dynamics. There algorithms are currently not used in fields of plasmas (PIC simulations).

Basic idea behind TC algorithm is to replace particle-particle interaction with particle-cluster interactions in a such way, that we reduce in number of necessary interactions (calculations) required without making to much error. With reduced number of clusters in comparison to number of particles, number of required iterations (and thus required simulation time) is reduced. Number of iterations for calculating interaction between particles is in range of $\mathrm{O}(N^2)$, with the use of TC method, number of interactions particle-cluster falls in range of $\mathrm{O}(N \log N)$ - this is a huge reduction in number of required calculations. In current plasma simulations, where number of particles is in order of 10^{22} (or 10^{17} super particles), number of interactions particle-particle is large even for today's super computers, or at least computers that are usually used to run simulations.

TC method is most simple in 1-D space, but can be also extended in a way, that can be used in 2-D or 3-D space. The difficult part in extension into more dimension is not TC method itself, but underlying equations that are solved using tree generated with TC method.

3.1 Background

There is no TC method without tree representation of a simulation domain. Tree is composed of nodes, links between nodes and leafs (figure 2a), with each node can having 0 or more children. Maximum number of children in each node of a tree is defined by dimensionality of domain, for which tree is created, and can be up to:

- 2 children for 1-D domain; also called binary trees.
- 4 children for 2-D domain; also called quad-trees.
- 8 children for 3-D domain; also called octrees.

There are nodes in tree that have special names: nodes without children are called leafs (nodes named *a*, *h*, *i*, etc. on figure 2) and node without parent is called root node (node named *root* on figure 2). Tree can be asymmetrical and does not need to be full - there can be nodes in tree that have only one child (in 1-D space).

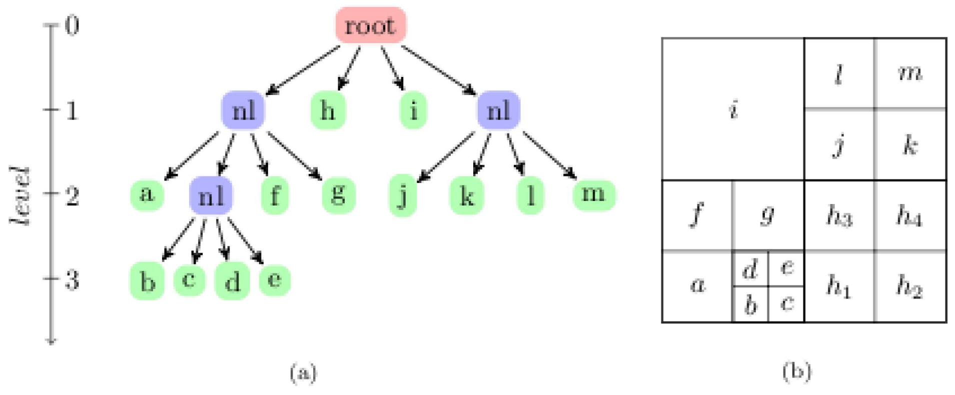

Figure 2: Example of tree representation (a) of square domain (b) (image take from [7])

3.2 Creating tree

The purpose of TC method is to represent relations (positions) between particles in simulating domain. On a base of generated tree, PIC simulation method can later perform faster calculation of forces between particles.

Because TC is similar for 1-D, 2-D and 3-D space, demonstration of tree generation in 1-D space can be easily extended for generation of trees in more dimensions. Simple flow chart for tree generation is shown in figure 3 and individual steps are as follows:

1. Node is generated on a base of input parameters: list of particles in simulating domain, start and end limit of domain (to limit which particles are covered in current node) and some kind of decision value, if current node should be divided further or division into nodes stops at current level.

2. In this step positions of particles are checked and particles that fall inside node limits (given by start and end position) are stored inside node.

3. Now is important decision: should we continue division of node into two (in 1-D space) children or not.

4. If division into nodes is needed, then two child nodes are created: one for left half of current node limits and another for right half of nodes domain. If there is situation that either half of domain does not have particles inside, no child node is created for that half of domain.

5. At the end, created tree node is returned.

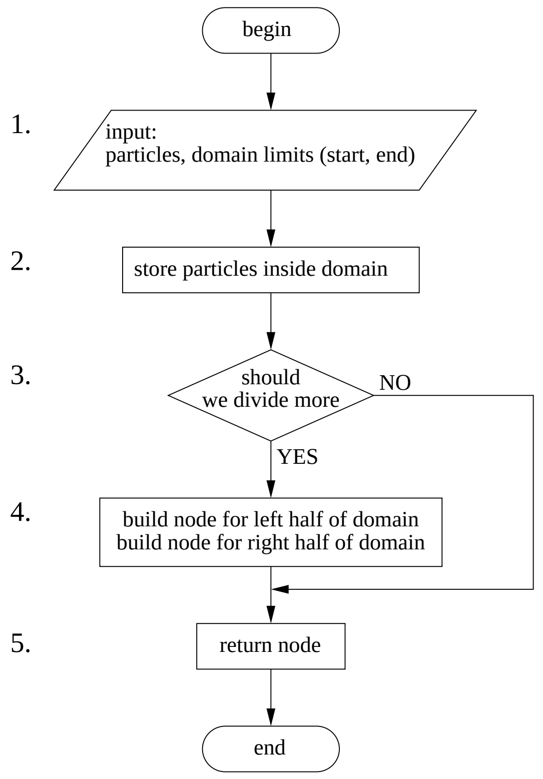
begin
1.
input:
particles, domain limits (start, end)
2.
store particles inside domain
3.
should
we divide more
NO
YES
4.
build node for left half of domain
build node for right half of domain
5.
return node
end

Figure 3: Simple (basic) steps for creating tree

One of most important thing in creating tree is to define a condition, when to stop dividing tree nodes. If condition is too “hard”, tree will have small number of nodes and each node will have large number of particles inside. On the other hand, if condition is set in a way that small number of particles is inside tree node (or even to have only one particle in node), number of nodes will be too high and advantages of TC method will not prevail.

3.3 Example of generated tree in 2-D

The easiest way to demonstrate the generation of a tree is in 2-D space (figure 4).

- All starts from given domain in which we have particles with defined positions. Whole domain is root node of tree that we are building. Root node holds information about all particles in system.
- Then given domain is subdivided into 4 sub-domains of equal size (figure 4:level 1). Each of 4 new sub-domains is node under main (root) node.
- Now process is repeated for all 4 sub-domains: first particles in node are calculated and if numner of particles is larger that given limited number, then domain is subdivided into 4 domains of equal size. If domain is subdivided, new sub-domains become sub-nodes of current node.
- This procedure is repeated until there are domains that can be sub-divided into sub-domains.

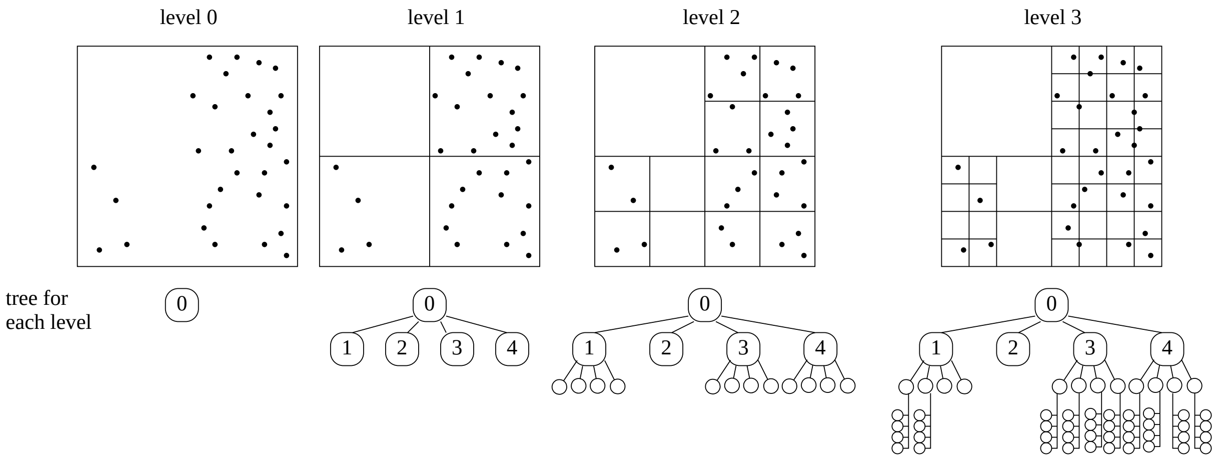

Figure 4: Generation of tree and dividing domain into sub-domains.

4 Treecode model in PIC simulation

TC method can be used in PIC simulation to replace some steps in main PIC simulation loop - figure 5 shows steps in main PIC loop that are replaced with TC method. New (with TC method) PIC main loop is shown on figure 6. Because in each time step (in each loop) in PIC simulation new particles are injected into simulation domain, tree has to be generated again.

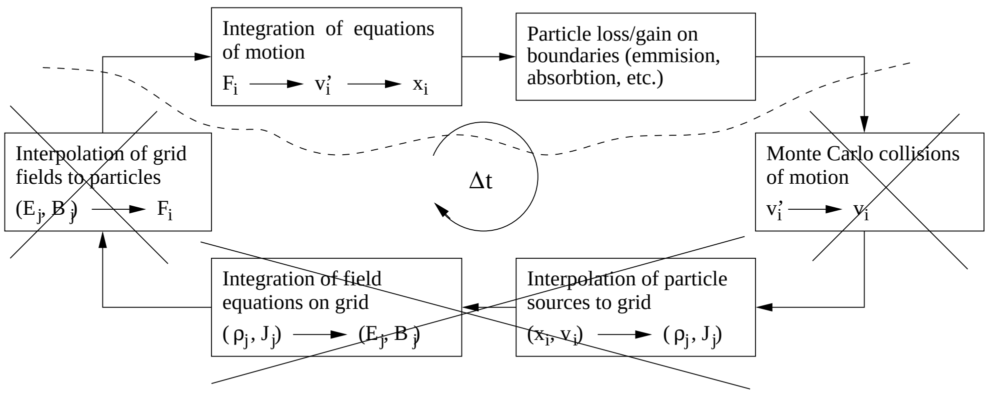
Integration of equations of motion
$F_i \rightarrow v_i' \rightarrow x_i$
Particle loss/gain on boundaries (emmision, absorbtion, etc.)
Monte Carlo collisions of motion
$v_i' \rightarrow v_i$
Interpolation of particle sources to grid
$(x_i, v_i) \rightarrow (\rho_j, J_j)$
Integration of field equations on grid
$(\rho_j, J_j) \rightarrow (E_j, B_j)$
Interpolation of grid fields to particles
$(E_j, B_j) \rightarrow F_i$
Δt

Figure 5: Steps in main PIC loop that are not necessary when using treecode

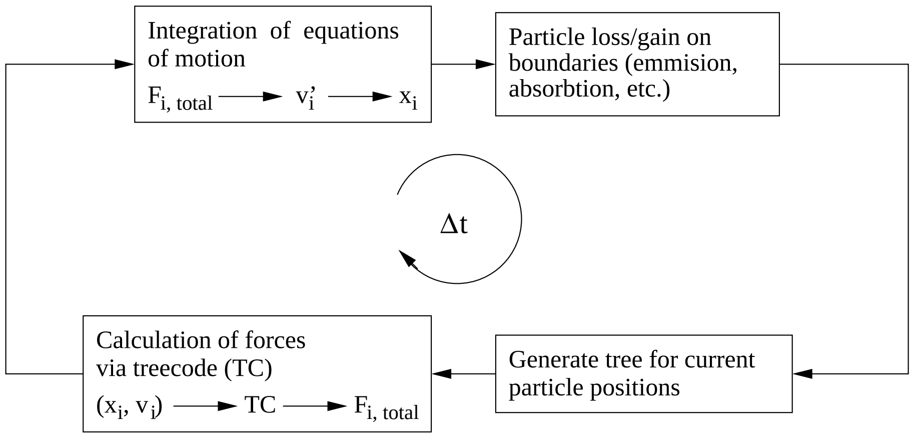
Integration of equations of motion
$F_{i,total} \rightarrow v_i' \rightarrow x_i$
Particle loss/gain on boundaries (emmision, absorbtion, etc.)
Generate tree for current particle positions
Calculation of forces via treecode (TC)
$(x_i, v_i) \rightarrow TC \rightarrow F_{i,total}$
Δt

Figure 6: Simple PIC main loop with treecode

General idea for calculating the force on current particle is to replace summation between particles with summation between particle and clusters. It is important to distinguish forces in treecode method from force in PIC simulations. Force calculated in treecode method is total force and we can write

(1) $$F_{i,total} = F_i + \tilde{F} = F_{PIC} + F_{pert}$$

where F_{total} is force calculated in TC method, F_{PIC} is force calculated in PIC simulation (locally averaged force) and F_{pert} is a force that takes into accont perturbations (for example, collusions).

Instead of calculating forces between particle (particle-particle interaction), we can use particle-cluster interaction, defined with tree generated in first step of TC method. Instead of

(2) $$\vec{F}_{i,total} = \frac{q_i}{4\pi\varepsilon_0} \sum_{j=1}^{N_C} \frac{q_j}{(r_i - r_j)^2}$$

we can use

(3)
$$\vec{F}_{i,total} = \frac{q_i}{4\pi\varepsilon_0}\frac{q_C}{R^2}$$

where N_C is number of particles in cluster, j is index in cluster (figure 7). Of course is q_C calculated with summation, but is calculated only once - in process of generating a tree.

(4)
$$q_C = \sum_{j=1}^{N_C} q_j$$

Center of cluster is calculated with

(5)
$$x_C = \frac{1}{N_C}\sum_{j=1}^{N_C} x_j$$

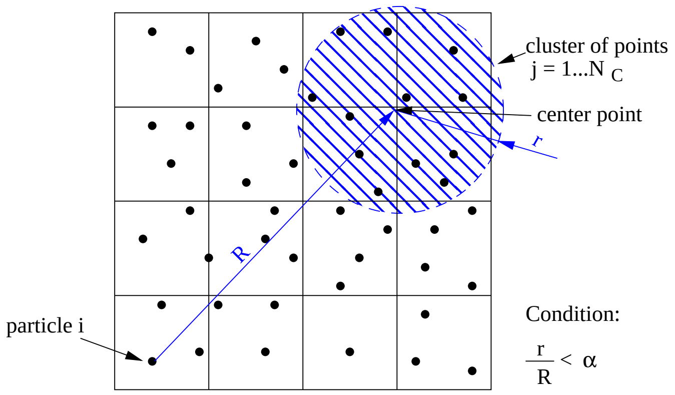

Figure 7: How clusters are defined and used in force calculation

If we create tree that has much less clusters (leafs in tree) than particles, we can expect reduction of simulation times. Calculating of force on a single particle (following algorithm has to be repeated for all particles in system) is composed from following steps (figure 8):

1. For calculating the force, one has to define all particles in a domain, current particle for which to calculate force and of course generated tree in form of node from which to start. At the beginning of calculation of force, root node of tree is given.

2. This step is important: here we define if we can use particle-cluster interaction to calculate force or should we do direct summation/continue down the tree. Condition is usually defined by distance between particle and center of cluster (for 2-D representation see figure 8). We define error tolerance parameter α, with values $\alpha < 1$. If relation $R > r_c/\alpha$, then we can use particle-cluster interaction. R is distance between particle and center of cluster, r is cluster radius (figure 7).

3. If particle "is too close to cluster" (based on condition, described before), then we check if current node is leaf (node without children). In that case, we use particle-particle interaction, i.e. direct summation of forces between particles.

4. In case node has children, we calculate force on current particle by summing all forces calculated for all children in current node.

5. At the end, calculated force is returned.

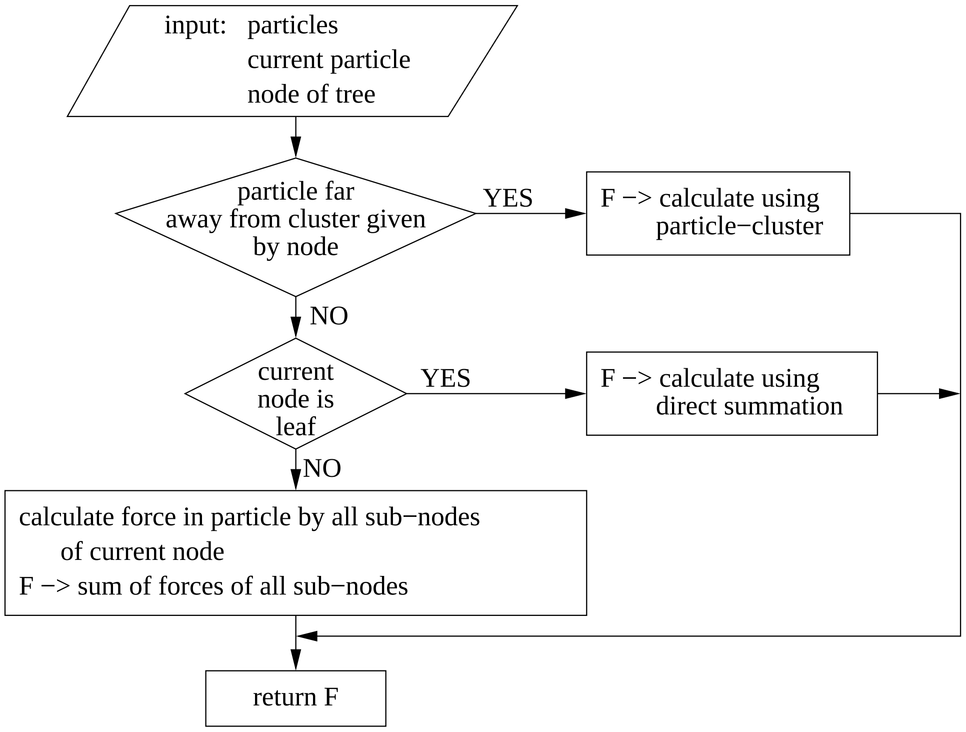
input: particles
current particle
node of tree
particle far away from cluster given by node
YES
F -> calculate using particle−cluster
NO
current node is leaf
YES
F -> calculate using direct summation
NO
calculate force in particle by all sub−nodes of current node
F -> sum of forces of all sub−nodes
return F

Figure 8: Steps in calculating force on a single particle

5 Conclusion

Current status of treecode in PIC simulation promises a lot - it seams that with proper selection of parameters for treecode (when to stop creation of clusters, . . .) we could gain a lot regarding shorter computing times.

Current available testing between PIC and TC method (as presented in [5]) is comparison in program made in MatLab. Results are not promising as TC method is almost 70% slower than PIC method, but a lot faster than direct summation. Because testing was done in MatLab, one can assume that program in languages that have better control over memory management and with better algorithms will be faster. Despite not promising results from testing presented in [5], TC method is promising - we can expect to reduce times required by PIC simulation if algorithms would be optimized and program would be done in C/C++ or any similar programming language.

TC method will be first tested as a stand alone PIC simulation for specific similation case and tested against current PIC simulation codes. In first step, TC method will be programmed in programming language C. After first tests and comparisons made, if it turnes out that method is promising, it will be rewritten in C++ and included in OOPD1 - PIC software developed in Berkeley Plasma Theory and Simulation Group from University of California, Berkeley, USA.

References

[1] *Introduction to Plasma Physics, with space and Laboratory Applications*. Cambridge University Press, 2005.

[2] J. Barnes and P. Hut. A hierarchical o(nlogn) force-calculation algorithm. , 324:446–449, dec 1986.

[3] C. K. Birdsal and A. B. Langdon. *Plasma Physics via Computer Simulation*. McGraw-Hill, 1985.

[4] A. J. Christlieb, R. Krasny, and J. P. Verboncoeur. A treecode algorithm for simulating electron dynamics in a penning-malmberg trap. *Computer Physics Communications*, 164(1-3):306–310, 2004.

[5] Andrew J. Christlieb, R. Krasny, and John P. Verboncoeur. Efficient particle simulation of a virtual cathode using a grod-free treecode poisson solver. *IEEE Transactions on plasma science*, 32(2):384–389, April 2004.

[6] Suhas V. Patankar. *Numerical Heat Transfer And Fluid Flow*. Tylor&Francis, 1980.

[7] Hari Sundar, Rahul S. Sampath, Santi S. Adavani, Christos Davatzikos, and George Biros. Low-constant parallel algorithms for finite element simulations using linear octrees. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[8] John P. Verboncoeur. Particle simulation of plasmas: review and advances. *Plasma Physics and Controlled Fusion*, 47:A231–A260, 2005.

[9] John P. Verboncoeur. Partice-in-cell techniques. Technical report, Department of Nuclear Engineeringn, University of California, Berkeley, CA-94720-1730, May 2007.

[10] J.P. Verboncoeur, A.B. Langdon, and N.T. Gladd. An object-oriented electromagnetic pic code. Technical Report UCB/ERL M94/71, EECS Department, University of California, Berkeley, 1994.