Plasmas Simulations using the Particle-In-Cell Model and the Finite Element Method in Parallel Architecture

G. N. Marques
A. J. Preto
S. Stephany
Laboratório Associado de Computação e Matemática Aplicada – LAC/INPE, Brasil
gleber@lac.inpe.br
airam@lac.inpe.br
stephan@lac.inpe.br

A. Passaro
N. M. Abe
A.C.J.Paes
Virtual Engineering Laboratory, Institute for Advanced Studies – IEAv/CTA, Brasil
angelo@ieav.cta.br
nancy@ieav.cta.br
acjpaes@ieav.cta.br

Abstract

This paper presents the performance results obtained for the parallel version of an object-oriented electrostatic Particle-in-Cell (PIC) code for plasma simulation. In PIC simulations, the plasma is modeled by thousands or even millions of charged particles, which suffer the action of the self-consistent and external electric and magnetic fields. In this work, the self-consistent field is computed by the Finite Element Method. The external and self-consistent fields result in a Lorentz force acting on each particle. Under the action of this force, the particle positions and velocities are computed by integrating the equations of motion using a second-order time-centered quadrature method. The proposed parallel PIC-FEM code exploits the coarse-grained parallelism presented by the charge distribution process, the computation of the self-consistent field and the solution of the equations of motion. The code was written in C++ language and parallelized using calls to the MPI (Message Passing Interface) library.

Keywords: *plasma simulation, high performance cluster computing, finite element method, particle-in-cell model*

1. Introduction

The study of plasmas presents great interest for many scientific and industrial applications such as the investigation of natural plasmas in Space Geophysics [7], [20], plasma propulsion [8], high intensity laser beam [9], discharges in plasmas [14], controlled nuclear fusion [19], magneto-aerodynamics [21] and many others [15]. We have investigated plasma behavior using the Particle-In-Cell (PIC) methodology, which is proper for observing the plasma collective behavior. In this methodology the Vlasov equations are indirectly solved by means of the Maxwell field equations and the Newton motion equations. A short description of the PIC methodology is presented in section 2. The enormous number of particles requires the use of parallel computing, and some parallel strategies are presented in section 3. Parallel PIC codes have to deal with the communication of particle positions and the contribution of each node in the electromagnetic fields. However, since the particles move in the domain while the nodes are fixed entities during the simulation, data dependencies may arise and limit the parallel performance. Usually, Maxwell equations are solved by a Finite Difference Method (FDM) or Spectral Methods, but in this work we adopt the Finite Element Method (FEM). A parallel PIC-FEM (PPIC-FEM) object-oriented code for non-collisional plasma simulation is used for numerical experiments in a distributed memory parallel architecture.

2. The PIC methodology and the FEM

In the PIC model, the plasma is simulated by tracking the motion of the charged particles (electrons and ions) that represent the plasma. The forces acting on the plasma have both internal and external origin. Internal forces are due to the self-consistent electromagnetic fields originated from the charged particles of the plasma.

In order to compute the self-consistent field, the charge of each particle is distributed in the nodal points of the finite element (the cell) that contains it. The self-consistent magnetic field originated from the motion of the charged particles is neglected in the electrostatic approach. Additionally, if there are no external field sources acting on the particle system, the resulting force on the particles may be considered as caused by the self-consistent electric field. Under these considerations and using the scalar potential for

the electric field, the Maxwell equations can be reduced to the Poisson equation:

$$\nabla \cdot \varepsilon \nabla \phi = -4\pi\rho \,, \tag{1}$$

where, ϕ is the electric scalar potential, ρ is the charge density function, and ε is the electric permittivity. Before solving the Poisson equation, the charge density function ρ must be computed.

For this purpose, the domain is decomposed in a finite element mesh that is sufficiently small to resolve the collective behavior of the plasma and is topologically regular. These mesh elements are named cells in PIC literature, and its linear dimensions must be close to the Debye length (λ_D), that is the distance over which the influence of the electric field of an individual charged particle is felt by the other charged particles in the plasma. The charge of each particle inside each cell is distributed among the cell vertexes following some criteria.

At this point one can solve Poisson's equation (1) for the electric potential and then calculate the electric field acting on the particle positions. The Newton's motion equations are used to update particle positions assuming the resulting force is given by Lorentz force equation. Since we are interested in investigating high frequency oscillations, the time step must be sufficiently small to solve the highest frequency oscillation in the plasma, called the electron-plasma frequency. As particles move, the charge distribution changes and so does the electric field. The charge distribution is then recalculated, and this cycle is repeated until a predefined number of time steps is reached.

2.1 Charge distribution

Different strategies can be adopted for the charge distribution calculation, provided the plasma macroscopic neutrality is preserved. The charge contribution of each particle to each cell vertex can be defined by using an area or distance weighted criteria.

When using the FEM for solving the Poisson's equation, the finite element mesh can also be used as the PIC cell structure. It is interesting to note that the finite element shape functions $N_I(\mathbf{x})$ satisfy the partition of unity condition:

$$\sum_{I=1}^{n} N_I(\mathbf{x}) = 1, \tag{2}$$

where, *n* is the number of vertices of the adopted finite element. In this way, each term of the sum

$$\sum_{I=1}^{n} N_I(\mathbf{x}_j) q_j = q_j \tag{3}$$

represents the charge contribution of the j^{-th} particle, p_j, to the i^{-th} vertex of the cell that contains the particle. Fig. 1 shows the four cell elements Ω_e that have node *i* as a vertex.

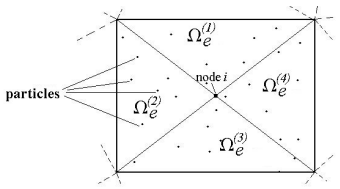

Figure 1. Particles contributing to the charge distribution in node i.

Thus, using Eq. 2, the sum over all charge contributions, yields the total charge Q_i accumulated at node *i* due to all particles p_j inside the corresponding cell element Ω_e [13]:

$$\sum_{p_j \in \cup_e \Omega_e^{(i)}} N_i(\mathbf{x}_j) q_j = Q_i \,. \tag{4}$$

2.2 Field computation

Once the charge distribution is known for all nodes, we must solve the following boundary value problem corresponding to the Poisson's equation (1) for the electric scalar potential $\Phi(\mathbf{x})$:

$$\nabla^2 \Phi(\mathbf{x}) = \rho(\mathbf{x}), \tag{5}$$

$$\Phi(\mathbf{x}) = \phi_{Dir}, \qquad in\ \ \Gamma_{Dir} \in \Gamma \tag{6}$$

$$\frac{\partial \Phi}{\partial n} = f_{Neu}, \qquad in\ \ \Gamma_{Neu} \in \Gamma \tag{7}$$

where, $\Gamma_{Dir} \cap \Gamma_{Neu} = \{\}, \Gamma_{Dir} \cup \Gamma_{Neu} = \Gamma$ is the boundary of the domain, and ρ is known at the nodal points. Let ϕ be an approximation for the unknown function $\Phi(\mathbf{x})$, given by:

$$\Phi \approx \phi(\mathbf{x}) = \sum_i N_i(\mathbf{x}) \phi_i \tag{8}$$

where, $\{N_i\}$ is a linearly independent set of base functions provided by the FEM. Applying the weighted residual method (WRM) procedure, and following the Galerkin approach, one can derive the corresponding discrete form [16]:

$$\mathbf{K}\vec{\phi} = \mathbf{b} \,, \tag{9}$$

where,

$$[\mathbf{K}]_{ij} = \int_{\Omega} \nabla N_j \cdot \nabla N_i \, d\varpi \tag{10}$$

$$[\mathbf{b}]_j = \int_{\Omega} \nabla N_j \rho \, d\varpi \tag{11}$$

$$[\vec{\phi}]_j = \phi_j \,. \tag{12}$$

2.3 Field interpolation

Using the shape function derivatives and the nodal potentials, one can perform the field interpolation. Since the state variable in the presented formulation is

the electric scalar potential, the electric field vector can be calculated by:

$$\mathbf{E}(\mathbf{x}_p) = -\nabla\Phi \cong -\sum_i \phi_i \nabla N_i \,. \quad (18)$$

2.4 Particle Advance

As the resulting force is assumed as due to the sum of the external and self-consistent electric and/or magnetic fields, the acceleration of the j^{-th} particle can be expressed as:

$$\mathbf{a}_j = \frac{q_j}{m_j}\mathbf{E}(\mathbf{x}_j) \cdot \quad (19)$$

Various methods can be used for integrating the time dependent motion equations, such as an Euler method [10], or a second order leap-frog algorithm [2]. We adopted the second one, whose difference equations are given by:

$$\mathbf{v}^{(n+\frac{1}{2})} = \mathbf{v}^{(n-\frac{1}{2})} + \mathbf{a}^{(n)}\Delta t \,, \quad (20)$$

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \mathbf{v}^{(n+\frac{1}{2})}\Delta t \,. \quad (21)$$

Given suitable initial conditions, it is then possible to iteratively compute the evolution of the plasma. The time increment Δt can be taken sufficiently small in function of ϖ_{pe}, the electron-plasma frequency, by $\Delta t = 0.2\varpi_{pe}^{-1}$.

3. A review of parallel strategies for PIC codes

The PIC simulation process can be divided in the four stages that occur in each iteration:

i. Charge distribution;
ii. Field computation;
iii. Field interpolation;
iv. Particles advance.

Although the above decomposition of tasks of a PIC simulation is quite simple and didactic, it hides important aspects of data dependencies that appear when performing the tasks in each one of the four stages. First, the stages *ii* and *iv* work exclusively over the set of nodal points or over the particles set. That is, once we have determined the charge distribution we need only nodal information to solve the Poisson's equation (stage *ii*) and then compute the electric field, and if the electric field is known everywhere in the domain simulation (particularly at the particle positions) the particles advance (stage *iv*) can be performed requiring only particles information, such as mass, charge, velocity and position. Differently, the stages *i* and *iii* require both particles and nodal information to be performed. In the charge distribution (stage *i*), we must perform a loop over the cell structure and then distribute the charge of each particle (particles information) to each cell vertex (nodal information). Also, the field interpolation stage requires both nodal and particles information as the electric field is interpolated at the particle positions by using the nodal values.

These data dependencies limit the performance of parallel PIC simulations, and must be dealt with an efficient data-partitioning scheme. In addition, nodes are fixed entities during the simulation while particles move anywhere in the domain. Therefore, according to the stage of the simulation, a processor may require a particle that is not local to its domain. The main challenge is to reach a trade-off between data decomposition and its implied communication. An efficient parallelization strategy must consider the parallel architecture, the size of the particle system and any relevant characteristic of the physical system evolution.

Lubeck and Faber [12] presented a simple decomposition scheme that consists in a static distribution of particles and nodes among processors. Each processor performs all the calculations related to its local particles and nodes. There is no migration of particles or nodes between processors, but each processor must have global nodal information for performing stages *i* and *iii*. The main advantage of this scheme is that it preserves the load balance in the processors at the cost of replicating field and cells information. This strategy is limited by the local memory availability in each processor.

In order to avoid data replication, Walker [22] proposed that only the nodal information required to perform evolution of particles that are local to each processor must be replicated in the local memory. The data replication is effectively reduced, but data communication is increased, since when particles move from one region to another the required nodal information to perform particle evolution may be stored in the memory of other processor. The performance of this strategy is strongly influenced by the decomposition scheme used to assign the particles to the processors, since the communication patterns depend on it [5]. In other words, the particle decomposition scheme must take into account some *a priori* information about the system evolution for optimizing communication between the processors.

GCPIC (General Concurrent PIC, [11]) uses independent domain decomposition schemes for particles and cells (nodes). The first decomposition divides the domain into regions containing approximately the same number of particles. Each region is assigned to a processor. Similarly, using a second decomposition, the nodes are distributed among processors. During the simulation, if a particle migrates from one region to another, the particle is sent to the processor that manages that region. Each processor is in charge the evolution of its own particles and controls particles that migrate to/from other regions. Since the two decompositions are independent, load balancing for the field solver is

preserved, but load unbalance can happen due to particle migration. If this causes an expressive concentration of particles, a dynamic load-balancing scheme may be required. However, in some cases it may be possible to find a simple, static decomposition scheme that provides an equivalent performance compared to that of a dynamic load balancing strategy.

In the hierarchical domain decomposition with unitary load balancing [5], particles and nodes are distributed among the processors using the same domain decomposition. This decomposition is performed considering some *a priori* information about the system evolution as preferential displacement directions, and predicted concentrations patterns. This reduces the particle migration between the processors, requiring less communication. In the unitary load balancing, the decomposition is done in such a way that the ratio between processing costs of particle advance and field solving is close to 1. Periodical evaluations of processing costs may lead to changes in the decomposition. The overheads for performing the unitary load balancing are equivalent to other algorithms based on particle number thresholds.

Ferraro, Liewer, and Decyk published results of a load-balancing strategy based on particle number threshold applied to the GCPIC scheme [6]. The spatial decomposition relative to the particles is adjusted whenever the number of particles inside a region exceeds a specified threshold. That work demonstrated that dynamic load-balancing algorithms must be used when considerably load unbalance is observed. In other words, a static spatial decomposition scheme properly chosen may also yield good performance.

Another strategy for load balancing is the adoption of a window-based scheme [18]. Windows (sub-regions) are created inside each region whenever load unbalancing occurs, and are assigned to processors with lower processing load. The windows do not affect the field solver stage, since this scheme is related only to particle load balance. The main advantage of this technique is that the domain decomposition is preserved, avoiding a new data decomposition during the simulation and the field solve stage is kept balanced. However, the management of these windows may become a complex and time-consuming task as their number increases. Moreover, when a particle migrates, the processor must determine the new region (or window) of destination, and the corresponding processor.

Some other optimization strategies have been proposed for improve PIC simulation performance, e.g., the reduction of cache miss [4] and the reduction of the number of particles during the simulation [21].

There is no general rule for parallelizing PIC simulations, since the effective performance of the parallel strategies is strongly influenced by patterns of particle concentration and displacement, and also the hardware architecture.

4. Computational implementation

4.1 PIC-FEM parallelization

An efficient parallel strategy depends on the employed algorithms, the dimensions and particularities of the particle system and the available parallel architecture. Distributed memory parallel machines are cost effective, but present a bottleneck of performance due to both the communication latency and bandwidth of the interconnection network. In our implementation the memory needed to store the particle data is approximately eight times the amount required to store the mesh and the sparse matrix data generated by the FEM. Therefore, in order to exploit coarse-grained parallelism we adopted a static domain decomposition only for the particles and each node of the cluster keeps a copy of the entire mesh data.

In the beginning of the simulation, the total number of particles of each type that compose the plasma is defined. The same amount of particles of each type is distributed among the processors. Mesh information and the particle positions are employed by each processor to generate a data structure that associates each particle to its corresponding finite element cell. Copies of the finite element mesh, source vector and the solution vector are kept in each processor memory, in order to proceed with the evolution of the assigned particle set. The methodology includes the following stages:

i) Charge distribution: each processor distributes the charge of each particle among the corresponding cell vertices, according to Eq. (3). These vertices correspond to the FEM nodal points, and so these charge contributions are stored in the local source vector of that type of particle. The local source vectors are then summed up composing the global source vector for each type.

ii) Field computation: the stiffness matrix is computed only in the first iteration since the mesh remains unchanged during the simulation. The field computation stage scales with the number of nodes and is considerably less time consuming than the stages that scale with the number of particles. Depending on the finite element order and on the size of the system to be assembled, parallel implementations of 2D FEM solvers present low efficiency for distributed memory machines connected by Fast Ethernet network. Thus, in this implementation the master processor executes this stage sequentially, and the resulting solution vector is broadcast to the other processors.

iii) Field interpolation and particle advance: each processor performs a loop over its particle set in

order to update their velocities and positions by means of the finite difference equations presented in section 2.4. The field on each particle is computed using the particle-cell association, i.e., each particle has a reference to the cell that contains it. After updating particle position, the particle-cell association data structure is efficiently updated by using the mesh connectivity information.

5. Results

In this section we present the results obtained for the simulation of plasmas composed of two particle types: electrons at an initial temperature of 10000 K, and protons at an initial temperature of 1000 K.

The particles have initially an uniform distribution and the velocities follow the Maxwell-Boltzmann distribution. The domain is a rectangle with dimensions of a hundred times the radius (length) of Debye. No external forces act on the plasma, i.e., the plasma oscillations are due only to the self-consistent field [16], and periodic boundary conditions are assumed for the particles.

Both particle types are advanced in the simulations, electrons and ions. In order to evaluate the speedup and scalability of the implementation, three sets of simulations are considered, with 100,000, 250,000 and 500,000 particles. The FE mesh is the same in all computations and the average number of particles in each cell is respectively 11.5, 28.8, and 57.6, for each number of particles.

The tests were carried out in a cluster of 12 computers with Athlon Barton 2500 processors, with 1 GB of main memory per node, connected by a Fast Ethernet network. Table 1 shows the processing times for different number of processors and Figure 4, the speedups obtained for the three simulations. It can be noticed that the speedup increases with the number of particles. Moreover, the sequential field computation causes small impact on the performance, since speedups are close to linear.

6. Conclusions

There are many parallel strategies for PIC simulations reported in the literature, as discussed in section 3. However, the best option depends on the algorithms, the dimensions and particularities of the particle system, and the available parallel architecture. For many years memory capability limited problem size in PIC simulations, and motivated research in parallel strategies that take advantage of memory scalability of distributed memory parallel machines. However, coarse granularity must be achieved in order to minimize communication costs, that are high is this kind of architecture. Complex schemes for data partitioning and communication have been proposed. On the other hand, strategies like the one reported in by Lubeck and Faber [12] and the presented in this paper precluded large simulations, since memory limited data replication, and communication costs limited the use of many processors. However, given the larger amount of memory and higher communication bandwidth of current machines, these limitations have lessened considerably. In addition, it is possible to obtain a better performance by trading communication for processing by means of data replication.

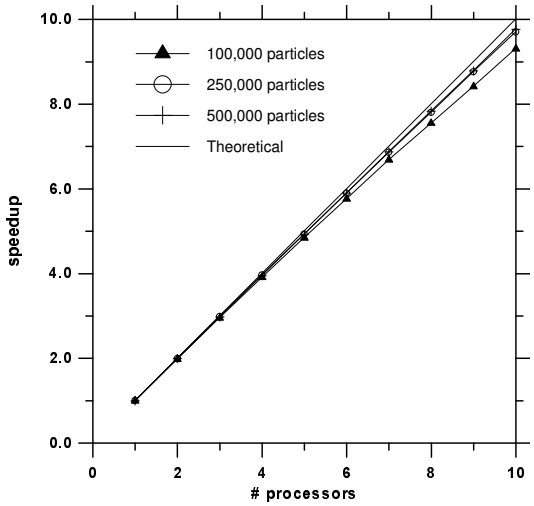

Figure 4: Speedup for 100,000, 250,000, and 500,000 particles (finite element mesh with 8678 first order triangular elements).

Table 1. Execution time in function of the number of processors and the total number of particles.

	Time (s)		
# proc	10^5 particles	2.5 10^5 particles	5.0 10^5 particles
1	12376.6	30610.9	61114.9
2	6246.18	15376.2	30639.7
3	4194.62	10277.6	20549.5
4	3166.7	7721.53	15463.3
5	2555.37	6209.2	12405.0
6	2146.87	5190.57	10354.4
7	1850.58	4461.79	8888.42
8	1639.47	3923.16	7802,04
9	1470.91	3492.87	6957.59
10	1330.06	3154.6	6255.86

The performance results show that the proposed PIC-FEM parallelization strategy, discussed in section 4.1, was appropriated for the given simulations. The scalability of the adopted strategy is still limited by the

amount of local memory and communication bandwidth, but this strategy allows the simulation of relatively large systems. The field computation stage is done sequentially, since its processing time is very small compared to the time spent in stages that depend on the number of particles.

The proposed parallel strategy requires fewer data exchanges and synchronization events between processors. Communication is only required for the reduction of the charge density vector and the broadcast of the field solution vector. This explains the good efficiency and scalability of the PPIC-FEM code, as demonstrated by the performance results obtained.

7. References

[1] Abe, N. M., Passaro, A., Franco, M.A. R., et al., 2002. Um Sistema de Software para Análise de Dispositivos e Componentes de Óptica Integrada, Fibras Ópticas e Microondas In *V Congresso Brasileiro de Eletromagnetismo (CBMag 2002)*.

[2] Birdsall, C. K. & Langdon, A. B., 1985. *Plasma Physics Via Computer Simulation*, McGraw-Hill.

[3] Breitkopf, P. *et al.*, "Explicit form and efficient computation of MLS shape functions and their derivatives", *Int. J. Numer. Meth. Engn.,* **48** (2000) 451-466.

[4] Boewers, K. J., 2001. Accelerating a particle-in-cell simulation using a hybrid counting sort. *Journal of Comput. Physics,* vol. 173, pp.393-411.

[5] Campbell, P. M., Carmona, E. A. & Walker, D. H., 1990. Hierarchical domain decomposition with unitary load balancing for electromagnetic particle-in-cell codes. *IEEE Concurrency,* pp.943-950.

[6] Ferraro, R. D., Liewer, P. C. & Decyk, V. K., 1993. Dynamic load-balancing for a 2D concurrent plasma PIC code. *Journal of Comput. Physics,* vol. 109(**2**), pp.329-341.

[7] Gary, S. P. & Nishimura, K., 2004. Kinetic Alfven waves: Linear *theory* and a particle in cell simulation. *Journal of geophysical and Research-space physics,* vol. 109 (A2).

[8] Garrigues, L., Heron, A., Adam, J. C. & Boeuf, J. P., 2000. Hybrid and *particle* in cell model of a stationary plasma thruster. *Plasma Sources Sci. Technol.*, vol. 9, pp. 219-226.

[9] Hermannsfeldt, W. B., 1987. Some applications of particle-in-cell *codes* to problems of high intensity beams. *Nuclear* Instruments *and Meth. Phys. Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 258(3), pp. 583-588.

[10] Lean, M. H., 1998. Particle simulation of ion cloud in a magnetic field. *IEEE Trans. Magn.*, vol. 34, n. 5, pp. 3122-3125.

[11] Liewer, P. & Decyk, V. K., 1989. A general concurrent algorithm for plasma particle-in-cell simulation codes. *Journal of Comput. Physics,* vol. 85, pp.302.

[12] Lubeck, O. & Faber, V., 1988. Modeling the performance of hypercubes: a case study using the particle-in-cell application. *Parallel Computing 9*, vol. 37, s/p.

[13] Melenk, J. M. & Babuska, I., 1996. The partition of unit finite element method: Basic theory and applications. Comput. *Meth. Appl. Engn.*, vol. 139, pp.289-314.

[14] Nanbu, K., Mitsui, K. & Kondo, S., 2000. Self-consistent particle modelling of dc magnetron discharges of na O_2/Ar mixture. *Journal of Physics D.: Appl. Phys.,* vol. 33, pp. 2274-2283.

[15] Paes, A. C.J., Abe, N.M., Serrão, V.A. e Passaro, A., 2002. Análise de um Mecanismo de Aceleração de Feixe Utilizando um Modelo de Partículas Autoconsistente e o Método dos Elementos Finitos. In *V Congresso Brasileiro de Eletromagnetism (CBMag 2002)*.

[16] Paes, A. C. J., Abe, N.M., Serrão, V. A, 2003. Simulation of plasmas with electrostatic PIC models using the finite element method. *Brazilian Journal of Physics*, vol. 33, pp.411 – 417.

[17] Passaro, A., Franco, M. A. R., Machado, J. M. & Cardoso, J. R., 1999. Modal Analysis of Quasi-Guided Waveguides by the Finite Element Method with Spatial Transformations. In *1999, SBMO/IEEE MTT-S, AP-S and LAOS International Microwave and Optoelectronics Conference (IMOC'99)*.

[18] Plimpton, S. J., Seidel, D. B., Pasik, M. F. *et al., 2003.* A load-balancing algorithm for a parallel electromagnetic particle-in-cell code. *Comput. Physics Commun.,* vol. 152, pp.227-241.

[19] Rantamäki, K. M, Pättikangas, T. J. H., Karttunen, S. J. *et al.,* 1999. Particle-in-cell simulations of parasitic absorption of lower hybrid power in edge plasmas of tokamaks. *Plasma Phys. Control. Fusion*, vol. 41, pp. 1125-1133.

[20] Scales, W. A., Cheng, K. T. & Srivastava, S., 1997. Simulation studies of process associated with stimulated electromagnetic emissions (SEE) in the ionosphere. *Journal of Atmospheric and Solar-Terrestrial Physics,* vol. 59 (18), pp. 2373-2381.

[21] Shang, J. S. "Recent research in magneto-aerodynamics", *Progress in Aerospace Sciences*, **37** (2001) 1-20

[22] Shon, C. H., Lee, H. J. & Lee, J. K., 2001. Method to increase the simulation speed of particle-in-cell (PIC) code. *Computer Physics Commun.,* vol. 141, pp.322-329.

[23] Walker, D. W., 1989. The implementation of a three-*dimensional* particle-in-cell code on a hypercube concurrent processor. In Monterey, CA., *Fourth Conference on Hypercube Concurrent Computers and Applications*.