ANNEX I

The Self-Consistent 3D Electrostatic Code *ARIADNE*: Solver Improvement and the Parallelization of the Code

John Gr. Pagonakis and John L. Vomvoridis
School of Electrical and Computer Engineering
National Technical University of Athens

INTRODUCTION

The trajectory code ARIADNE has been developed for the self-consistent simulation of three-dimensional electrostatic effects in a gyrotron beam tunnel. For the beam tunnel of a conventional gyrotron it can be used to study effects associated with deviations from cylindrical symmetry, such as those produced by non-uniform electron emission from the cathode, mis-alignments between the mechanical and the magnetic axis, etc. In addition, it can handle fully three-dimensional beam tunnels (with dependence on the polar angle), as is the case of a sheet beam for quasi-optical gyrotron. Some details about the structure and the function of the code have been already presented in the previous year report (Annex I, 2001). During last year, several improvements have been addressed in the code which are described by the following tasks:

- The linear interpolation used for the potential description was upgraded to quadratic.
- The solver of the sparse linear system and the subroutine for the beam calculation were parallelized.
- The user-interface was modified to control all processors of the parallel computer system.

LINEAR vs QUADRATIC INTERPOLATION

For the solving of Laplace and Poisson equations, the code uses the finite element method. In the earlier version of the code, the description of the potential in the region of each tetrahedrical element of the finite element mesh is achieved by linear interpolation of potential values between the four vertexes of each element (four degree of freedom per element). In particular, the potential is described by the equation

$$\phi(\xi, \eta, \zeta) = a_1 + a_2\xi + a_3\eta + a_4\zeta \tag{1}$$

where (ξ, η, ζ) are the spatial variables and the a_s are calculated by the constraint that eq. 1 should satisfy the values of the potential at the four vertexes of the tetrahedron. As a result, the three components of the electric field (which are the derivatives of the potential function) have constant values in each tetrahedron. For this reason, an extreme dense mesh is demanded for the approximation of the electric field. To avoid this difficulty in the improved version of the code, the description of the potential in the region of the elements takes into account not only the values of the potential at the four vertexes but also the potential values at the middle points of the six edges (ten degree of freedom per element). In particular, the potential is expressed by the equation

$$\phi(\xi, \eta, \zeta) = a_1 + a_2\xi + a_3\eta + a_4\zeta + a_5\xi^2 + a_6\xi\eta + a_7\xi\zeta + a_8\eta^2 + a_9\eta\zeta + a_{10}\zeta^2. \tag{2}$$

In this case, the components of the electric field vary linearly in the region of the elements. The cost of this improvement is the increasing of the order and of the number of non-zero values of the sparse linear system which is generated by the application of the finite element method, as well as the need for computer memory and computational time. The solution of this difficulty is the parallelization of the code and the execution in a parallel computer system.

THE CODE STRUCTURE

The newer parallel version of the code uses the library MPI (message passing interface) and is able to be executed in distributed parallel computer systems (clusters). For this reason, the structure of the code has been modified to interconnect the function of all available processors of the cluster and the user (see fig. 1). The process which the code initial executes is called master process while the other available processes are called slaves. The user introduces the command in the appropriate syntax to the command line editor and in the interpreter the command is coded to a simpler form, which is understandable by the control. The slave processes are informed about the user command and the slave control subroutine determines the next step. When the command is to be executed only by the master, a terminal message is sent from each slave to master. Otherwise, the appropriate subroutine is called and executed simultaneously with the master process.

THE PARALLELIZATION OF THE SOLVER AND OF THE PARTICLE PUSHER

The parallelization of the solver subroutine is based on the domain decomposition method. The total mesh is subdivided to a number of domains equal to the available processors. In particular, the element-base partitioning is used, i.e. no elements are split along two domains and therefore all information related to a given element is mapped to the same processor. The geometry of the beam tunnel is longitudinal and the subdivision is achieved along the axis of propagation. So, there are the internal nodes, which belong to the elements of the same domain, and the separator nodes, which belong to the interface between two domains. Care has been taken that the number

of nodes of each domain be almost the same. The enumeration of the nodes, which will determine the form of the sparse matrix of the system (i.e., the position of the non zero values) proceeds first with the internal nodes of each domain and then advances to the separator nodes. Under these arrangements the sparse linear system, with unknowns the potential ϕ_{D_i} at the internal nodes of the domain i and $\phi_{S_{i,j}}$ at the separator nodes between the domains i and j, has the form

$$\begin{bmatrix} D_1 & & & & C^l_{1,2} & & & \\ & D_2 & & & C^r_{1,2} & C^l_{2,3} & & \\ & & \ddots & & & & \ddots & \\ & & & D_{N_D} & & & & C^r_{N_D-1,N_D} \\ C^{l\,T}_{1,2} & C^{r\,T}_{1,2} & & & S_{1,2} & & & \\ & C^{l\,T}_{2,3} & & & & S_{2,3} & & \\ & & \ddots & & & & \ddots & \\ & & & C^{r\,T}_{N_D-1,N_D} & & & & S_{N_D-1,N_D} \end{bmatrix} \begin{bmatrix} \phi_{D_1} \\ \phi_{D_2} \\ \vdots \\ \phi_{D_{N_D}} \\ \phi_{S_{1,2}} \\ \phi_{S_{2,3}} \\ \vdots \\ \phi_{S_{N_D-1,N_D}} \end{bmatrix} = \begin{bmatrix} b_{D_1} \\ b_{D_2} \\ \vdots \\ b_{D_{N_D}} \\ b_{S_{1,2}} \\ b_{S_{2,3}} \\ \vdots \\ b_{S_{N_D-1,N_D}} \end{bmatrix}$$

where, D_i is the symmetric main matrix of the i domain, $S_{i,j}$ is the symmetric main matrix of the separator between the i and j domains, $C^{l}{}_{i,j}$ is the connection matrix of the left separator with the i domain and $C^{r}{}_{i,j}$ is the connection matrix of the right separator of the j domain. Each process is responsible for the generation and storage of the main and connection matrix of one domain while the main separator matrices (which are much smaller than the domain main matrices) are distributed to the processes. The main domain and separator matrices are decomposed $D_i = U^T_{D_i} + G_{D_i} + U_{D_i}$ and $S_i = U^T_{S_i} + G_{S_i} + U_{S_i}$, where G and U demonstrate diagonal and upper triagonal matrix, respectively. The system is solved using the successive over relaxation iterative method. In particular, the algorithm consists of the following steps:

- Each processor initializes the values of the potential at the domain and separator nodes it is responsible for.
- The potential values at the separator nodes are sent to two processors, which are responsible for the two domains separated by the separator.
- The new values of the potential at the internal nodes of all domain are simultaneously calculated by the equation

$$\left[G_{D_i} + \omega U^T_{D_i}\right] \phi^{k+1}_i = \left[(1-\omega)G_{D_i} - \omega U_{D_i}\right] \phi^k_i - \omega C^r_{i-1,i}\phi^k_{i-1,i} - \omega C^l_{i,i+1}\phi^k_{i,i+1} + \omega b_i. \quad (3)$$

- In addition for all subdomains are calculated the help arrays

$$h^{l\,k+1}_i = C^{l\,T}_{i,i+1}\phi^{k+1}_i, \;\; h^{r\,k+1}_i = C^{r\,T}_{i-1,i}\phi^{k+1}_i \quad (4)$$

  and are sent to the processors which are responsible for the left and right separator.
- The new potential values are calculated for the separator nodes by the application of the equation

$$\left[G_{S_{i,i+1}} - \omega U^T_{S_{i,i+1}}\right] \phi^{k+1}_{i,i+1} + h^{l\,k+1}_i + h^{r\,k+1}_{i+1} = \left[(1-\omega)G_{S_{i,i+1}} - \omega U_{S_{i,i+1}}\right] \phi^k_{i,i+1} + \omega b_{i,i+1}. \quad (5)$$

- Finally, if the maximum increment of the potential values exceeds the specified by the user tolerance the process returns to the second step. Otherwise the computation is terminated.

In fig 2 is presented the curve of computational time needed with the number of processes for the simulation of a 140GHz gyrotron gun with the use of a mesh with 199347 nodes.

For the parallelization of the particle pusher no information exchange is needed between the processors of the cluster. The trajectory of each electron is independent of the trajectory of all other beam electrons. So, the beam electrons are distributed to the available processes and the calculation of the beam is achieved in a fully parallel way.

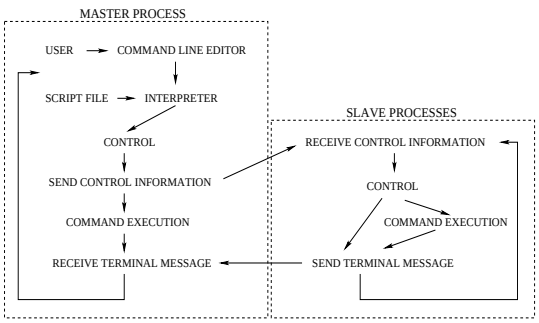

Figure 1: The structure of the code.

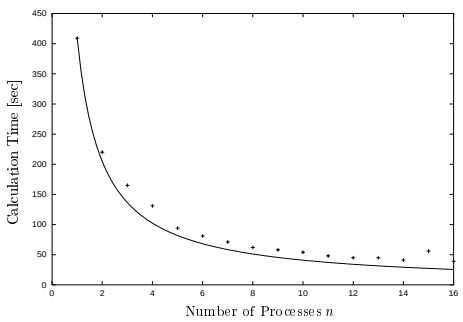

Figure 2: Computer time with number of processes.