Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 164 (2004) 297–305
Computer Physics Communications
www.elsevier.com/locate/cpc

Implementations of mesh refinement schemes for Particle-In-Cell plasma simulations ☆

J.-L. Vay [a,*], P. Colella [a], A. Friedman [b], D.P. Grote [b], P. McCorquodale [a], D.B. Serafini [a]

[a] *Lawrence Berkeley National Laboratory, CA, USA*
[b] *Lawrence Livermore National Laboratory, CA, USA*

Available online 27 July 2004

Abstract

Plasma simulations are often rendered challenging by the disparity of scales in time and in space which must be resolved. When these disparities are in distinctive zones of the simulation region, a method which has proven to be effective in other areas (e.g., fluid dynamics simulations) is the mesh refinement technique. We briefly discuss the challenges posed by coupling this technique with plasma Particle-In-Cell simulations and present two implementations in more detail, with examples.

Keywords: Mesh refinement; Particle-In-Cell; Plasma

1. Introduction

Numerical simulations of ion beam transport in a Heavy Ion Fusion (HIF) [1] accelerator and reaction chamber currently model different stages of the process separately. A completely self-consistent treatment, which is ultimately needed, requires an end-to-end simulation from the ion source to the fusion target. This represents a real challenge, even extrapolating near-future computer power, and we must consider the use of the most advanced numerical techniques. One of the difficulties of these simulations resides in the disparity of scales in time and in space which must be resolved. When these disparities are in distinctive zones of the simulation domain, a method which has proven to be effective in other areas (e.g., fluid dynamics simulations) is the Adaptive-Mesh-Refinement (AMR) technique. We have begun the exploration of introducing this technique into the Particle-In-Cell (or PIC) method. A collaboration between the Heavy Ion Fusion Virtual National Laboratory (HIF-VNL) and LBNL's Computational Research Division was initiated to develop an AMR library of subroutines [2] targeted at providing AMR capabilities for existing plasma PIC simulation codes [3].

☆ Work performed for USDOE under Contracts DE-AC03-76F00098 at UC-LBNL and W-7405-ENG-48 at UC-LLNL.
* Corresponding author.
E-mail address: jlvay@lbl.gov (J.-L. Vay).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2004.06.075

2. Application of mesh refinement to Particle-In-Cell electrostatic plasma simulation

2.1. Two possible strategies for coarse-fine grid coupling

When solving the Poisson equation with mesh refinement, several strategies can be envisioned to couple a fine grid and the coarser grid in which it is enclosed; we considered two of them. We consider a grid (denoted the "coarse" or "parent" grid) and its refinement patch (denoted the "fine" or "child" grid).

The method which is conceptually the simplest consists of solving the Poisson equation on the coarse grid first, ignoring the presence of the patch, and then solving on the patch alone using Dirichlet boundary condition derived by interpolation of the solution of the coarse grid.

A second method, which is the default in the Chombo package [2], consists of iterating the solution back and forth between the patch and its "parent" grid. After one iteration on the coarse grid, values on the fine grid are interpolated from the coarse grid solution. Then, a specified number of iterations are performed on the fine grid and the fine and coarse grid solutions are reconciled during a "synchronization" step which consists in imposing the fine grid solution on the coarse grid nodes located inside the fine grid patch. This procedure is iterated until convergence [4].

In the rest of the article, we will refer to the first method as "1-pass" and the second as "multipass".

2.2. Issues

While the second method has been shown to be of higher order in accuracy, it violates a discrete version of Gauss' law under a nodal implementation because it modifies the coarse grid solution. Eventually this introduces a nonlinearity that is not present in the coarse-grid solution. This effect may be an issue for accelerator modeling (unharmonic forces).

Also, the use of AMR implies breaking the symmetry in the field solution which in turn introduces a spurious force when gathering the electric field from the potential on the set of grids. This may potentially alter the particle motion to a degree which cannot be neglected. This effect was studied in detail for a one particle test in [5]. It was determined that, when using the "1-pass" solver, a sufficient number of guard cells can be defined on the border of the patch that effectively mitigates most of the effect of the spurious force, since its amplitude grows with the inverse of the distance to the patch border. In effect, the effective area of the patch is delimited by its total area minus the guard cells. No such simple mitigating technique can be applied when using the "multipass" solver.

We refer the reader to [5] for a more detailed discussion of these issues.

3. Two examples of implementations

A prototype AMR Poisson solver was built on the foundations of the WARP axisymmetric (r, z) multigrid Poisson solver, using the "1-pass" scheme for coarse-fine grid coupling and guard cells to reduce the effect of spurious self-force, as described above. While this RZ prototype is allowing us to begin to explore the benefits of AMR on injector simulation, the production-level general three-dimensional AMR-Poisson solver is being developed as part of the Chombo package. We briefly describe the two implementations. We note that in both cases, linear weighting is used for charge deposition and force gathering.

3.1. Prototype implementation in WARP-RZ solver

The implementation of AMR in the axisymmetric (RZ) solver in WARP relies on the use of FORTRAN90 derived types. The type "grid" is defined as

```
type grid
  integer             :: id                  ! grid ID
  integer             :: nr, nz
  real                :: rmin, rmax, zmin, zmax
  real, allocatable   :: phi(:,:), rho(:,:)
  real, allocatable   :: phip(:,:), rhop(:,:) ! in parallel only
  real, allocatable   :: lp(:,:), lpfd(:,:)
  integer             :: gminr, gmaxr, gminz, gmaxz
  integer             :: nlevels, npre, npost
  real                :: mgparam
  type(bndy), pointer :: bnd
  type(grid), pointer :: up, down, next, prev
end type grid
```

The variables `nr` and `nz` define the dimensions of the patch array while the variables `rmin`, `rmax`, `zmin` and `zmax` define its extension in the physical system of coordinates. The arrays `phi` and `rho` store the electric potential and the charge density respectively. The arrays `phip` and `rhop`, which store the same quantities, are used only on parallel platforms for efficiency when a different domain decomposition is used for the fields and particles.

The arrays `lp` and `lpfd` are lookup tables that are used for rapid localization of particles in the grid structure during the steps of charge deposition and force gathering.

The variables `gminr`, `gmaxr`, `gminz` and `gmaxz` are the number of guard cells to be used on each side of the patch for the spurious self-force reduction.

The variables `nlevels`, `npre`, `npost` and `mgparam` control the multigrid solve and define respectively the number of multigrid levels, the number of relaxation steps before and after the coarsening stage and the relaxation parameter used in the Gauss–Seidel relaxation. These parameters can be optimized using a specialized routine. Since their optimized values are dependent upon the geometry and the grid mesh aspect ratio, the optimization is performed at run time. By defining them in the grid type, the optimizing routine can derive a set of optimized parameters for each individual patch.

The variable `bnd`, which is of derived type `bndy` (which we do not describe here), contains the variables that describe the part of the geometry which is enclosed into the considered patch.

The variables `up`, `down`, `next` and `prev` are pointers of type `grid` which are used to construct a tree structure by means of a 2-D linked list. The trunk of the tree is defined by default in WARP and constitutes the main grid covering the entire simulation domain. The user can add patches at run time by using the function add_subgrid at the Python level:

```
add_subgrid(id,nr,nz,dr,dz,rmin,zmin,gminr,gmaxr,gminz,gmaxz)
integer :: id           ! ID of grid to add a patch to (parent)
integer :: nr, nz       ! nb of meshes of patch in r and z
real    :: dr, dz       ! mesh size in r and z
real    :: rmin, zmin   ! min location of patch in physical coord.
real    :: gminr, gmaxr ! number of guard cells in r
real    :: gminz, gmaxz ! number of guard cells in z
```

The tree is maintained internally by WARP. When a patch is added, a new branch is added either to the grid of ID `id` by creating a new `down` pointer or, if `down` has already been allocated, to the last element of the linked-list described by `down.next...next`. The following restrictions apply:

(1) the patch must be entirely enclosed into its parent grid,
(2) the minimum and maximum of the patch are forced to lie on lines of the parent grid; the size of the mesh is resized if necessary.

The first restriction allows for a simple tree structure and avoids complications due to overlaps. The second allows for quicker testing of particle localization relative to the grid structure.

For each macroparticle, the charge deposition is performed on the finest grid which contains it. Once the charge from all the particles has been deposited, the charge of each patch is deposited onto its parent, starting from the finest patch to the main grid, recursively. The Poisson solve can then proceed, starting on the main grid and recursively solving down on each branch of the tree. The Poisson solve is activated by "call solve_allgrids_rz(maingrid,accuracy)" where the subroutine solve_allgrids_rz is:

```
recursive subroutine solve_allgrids_rz(g,accuracy)
  type(grid) :: g
  real       :: accuracy
  if(associated(g%up)) call interpolate(g,g%up,bnd_only)
  call solve_multigridrz(g, accuracy)
  if(associated(g%down)) call solve_allgrids_rz(g%down,accuracy)
  if(associated(g%next)) call solve_allgrids_rz(g%next,accuracy)
end subroutine solve_allgrids_rz
```

The routine checks first if the grid is a child and interpolates the field from the parent grid to the child if it is one (interpolates only on the patch boundary if bnd_only is true or on the entire patch otherwise). The Poisson equation is then solved on `g` and the entire operation is repeated recursively on any child (`g%down`) and 'brother/sister' (`g%next`).

For the force gathering, the field is interpolated from the finest grid containing the macroparticles, excluding the guard cells (in which it is interpolated from the parent grid).

In order to speedup the process of particle localization in the grid structure, the patch ID value is deposited in the array lp of the parent grid at the nodes covered by the entire patch (except the upper bounds). The same operation is performed on the array lpfd but considering the patch without the guard cells (the effective area of the patch). The arrays lp and lpfd are used as lookup tables in a recursion loop starting from the main grid for each macroparticle when depositing the charge and gathering the force. This avoids the cumbersome and inefficient series of boundary tests. Although this method may not be the most efficient that can be devised, it is easy to implement and balances simplicity and efficacy. Thus, it was considered of adequate efficiency for prototyping.

3.2. 2-D and 3-D implementation in the Chombo package

A nodal implementation of a multigrid AMR solver for the Poisson equation using Shortley–Weller ("cut cell") discretization of the Laplacian operator (to account for internal boundaries at subcell resolution) has been developed in Chombo [4]. In our configuration, a library containing Chombo's executable routines is provided to the WARP linker which merges the two packages together (see Fig. 1). Appropriate calls to Chombo routines are made by WARP's FORTRAN routines as enabled by a flag which is set by the user in WARP's Python script interface. For specialized use, some of the Chombo routines, such as its AMR Poisson solver, are callable directly from WARP's Python interface. Both methods for solving the Poisson equation, as described above, are being implemented in Chombo and are being tested and compared.

A prototype PIC interface to Chombo (called ChomboPIC) has been developed using this AMR Poisson solver and integrated into the WARP code. This prototype is purposefully simplistic, and maps straightforwardly to the structure shown in Fig. 1. The design goal for the prototype was to provide sufficient functionality to allow various

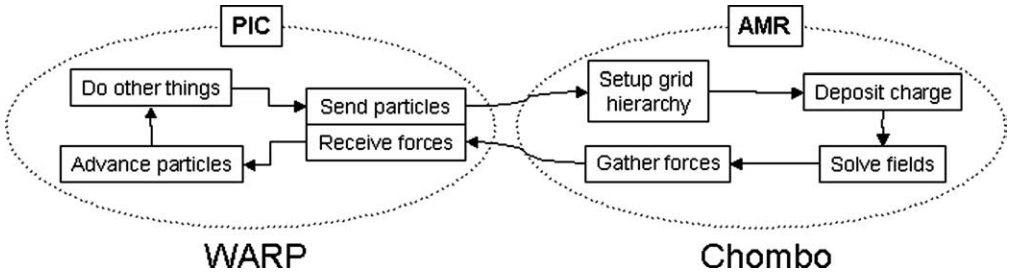

Fig. 1. Diagram of WARP/Chombo configuration.

applications to incorporate Chombo's adaptive mesh capability without significant changes to the application code itself, allowing experimentation with methods and interface details.

The data that must pass across the interface from the application (WARP) to ChomboPIC is limited to the locations and charges of the particles and the potential on the boundaries.[1] The data that passes back from ChomboPIC to the application is the electric field at the particles. In the prototype implementation, the particle data is passed back in the same ordering in which it was given. Because the adaptive method used in Chombo inevitably rearranges the particle data, restoring the original ordering adds an extra computational expense, as a result of copying the data. This expense increases in parallel with the number of processors, so an alternative scheme in which the particles are returned in a different ordering is likely to be needed in the future.

The simplest possible use of ChomboPIC from an application requires calls to 7 subroutines. Four are needed to initialize parameters and clean up afterwards, and usually need to be called only once each. The remaining 3 pass the particle data to Chombo, solve the Poisson problem, and get the results back. These would be called for each iteration around the loop shown in Fig. 1. ChomboPIC handles depositing the charge from the particles and interpolating the electric field from the solution back to the particles.

4. Examples of PIC-AMR simulations

Fig. 2 shows a snapshot taken from a movie of an end-to-end simulation of the High-Current experiment (HCX) [6] at LBNL. It shows the beam emitted from a triode source and traveling through the four accelerating and focusing electrostatic quadrupoles of the injector. The modeling of the triode region is critical since it determines the initial phase-space shape of the beam. It is also a challenging part to model since there is a large range of particle density close to the emitter and an accurate description of the edge of the emitter and the beam is crucial. This makes this problem ideal for testing the mesh refinement technique.

4.1. WARP example

We display in Fig. 3a) a snapshot of the beam taken from a quasi-steady state axisymmetric WARP simulation of the triode. By quasi-steady state, we mean that we run a time-dependent calculation of the beam being emitted from the source, solving for the field every n time steps with $10 < n < 50$ typically. We stop the simulation when an equilibrium is reached, under the assumption that the equilibrium solution exists and is unique (both of these assumptions are not guaranteed but seem to be fulfilled in practice).

Fig. 3b) shows the grid structure that we used when AMR was turned on. A refinement patch covering the whole beam was set up. In order to emulate a more complicated structure of grids covering only the emitting region and

[1] Although the Poisson solver implements complex geometry, due to time constraints the PIC prototype currently does not.

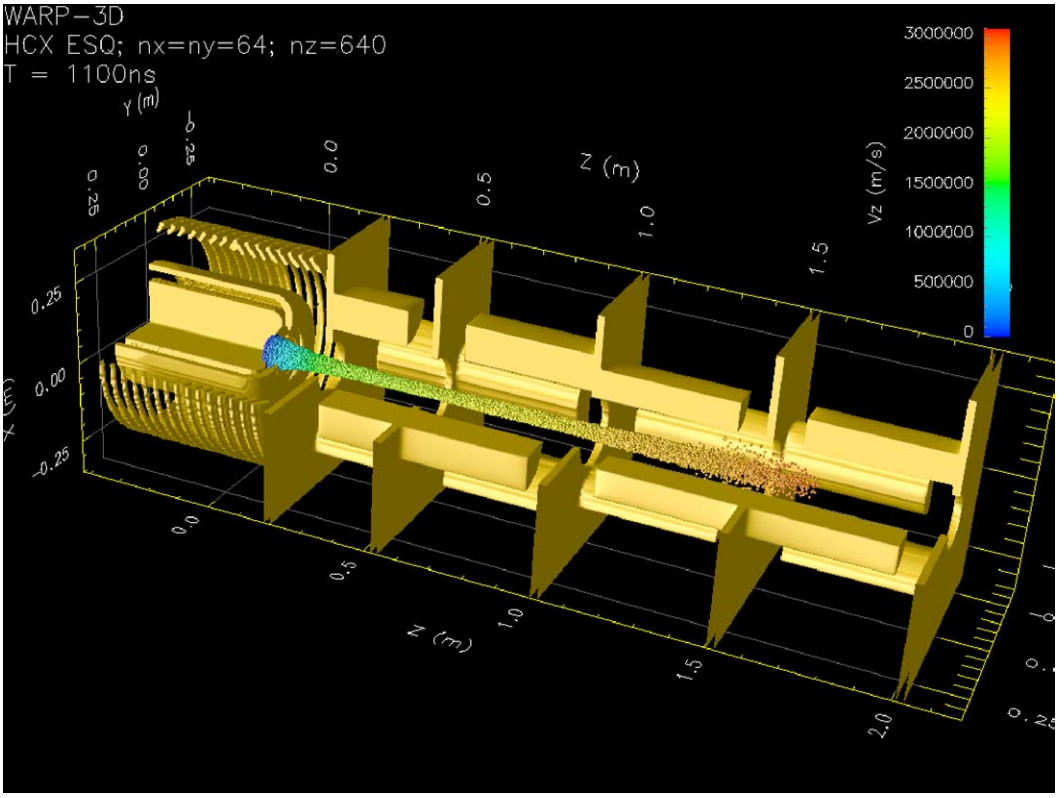

Fig. 2. 3-D rendering of HCX injector simulation from a movie of an end-to-end WARP simulation of the HCX experiment (http://hifweb.lbl.gov/webpages/theory/simulation_movies.html). This shows the beam, emitted from the source (left), propagating through the first quadrupole lenses.

the beam edge, the array *lpfd* (which controls, at the cell level, whatever the field acting on particles is taken from the patch solution or from its parent) was set in a special way such that the field from the main coarse grid was used inside the beam, while the field from the patch solution was used at the beam edge and around the emitter. Thus, the fine gridded area that is depicted in Fig. 3b) corresponds to the effective area of the patch, as defined above. This effective area was reset each time step to adaptively follow the edge of the beam.

The evolution of the RMS normalized emittance versus Z is shown in Fig. 4 for three runs using uniform low, medium and high resolution and a fourth run using medium resolution plus the refinement patch. The jump of resolution from low to medium and medium to high is a factor of 2 in each direction and a factor of 4 in the number of macroparticles, in order to keep the number of macroparticles per cell constant on average. The number of macroparticles used is the same in both of the medium resolution runs, with or without AMR. The results show that the emittance converges downward with increasing resolution and that the high resolution result is recovered from the medium resolution run with AMR at about a fourth of the computational cost.

4.2. Chombo example

The three-dimensional solution of the electrostatic potential in the HCX injector was computed with Chombo using its automatic meshing capability. The criterion for refinement was to refine volumes covering the edge of conductors of the source ($z < 0.1$ m) with a ratio between coarse and fine mesh of four. The result is displayed in

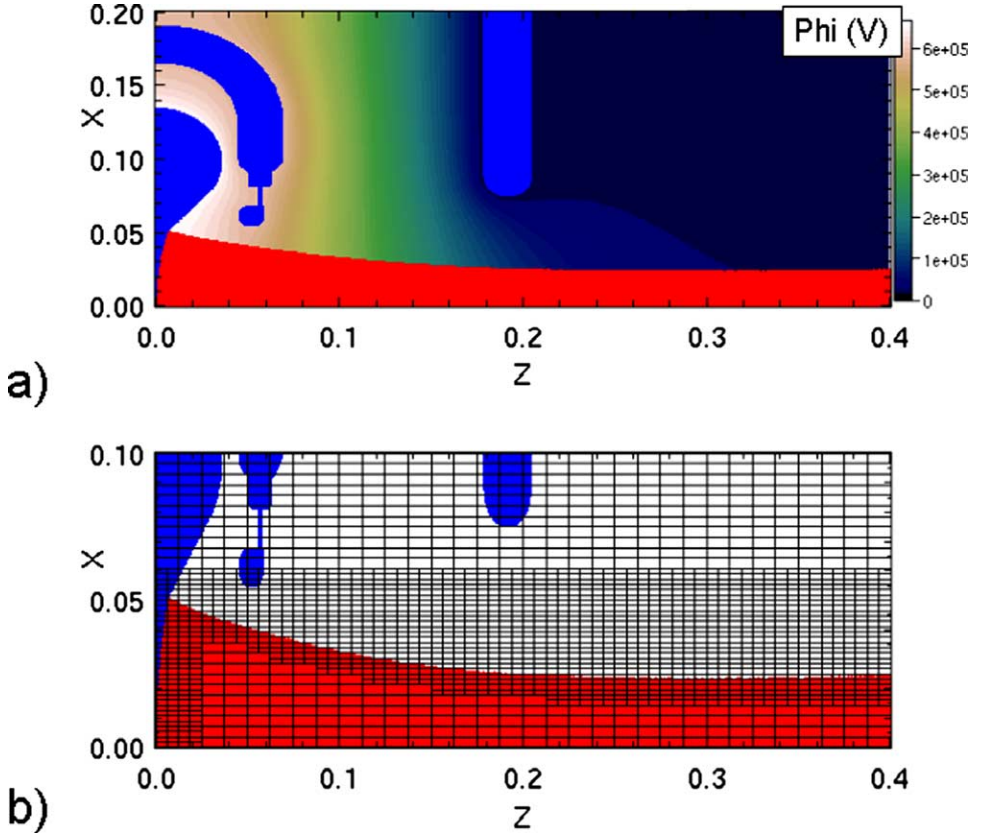

Fig. 3. a) color contour plot of electric potential with triode structure (blue) and beam (red); b) schematic of griding when using AMR: the emitting area and the beam edge are covered with a finer grid.

Fig. 5 and shows how Chombo will handle a complicated structure of grid blocks to get to the required solution optimally. The criterion used for refinement in this case is for demonstration purpose of the capability. Different criterion may be devised for actual modeling of the injector, for example refining the emitting region and the beam edge, as successfully used in the WARP-RZ prototype example described above.

5. Conclusion

We have presented a short overview of our efforts to couple the Particle-In-Cell and mesh refinement techniques. In this paper, we emphasized the description of the implementations of our prototype in WARP and the full-featured production package Chombo. Specific issues have been identified (for example, non-physical forces which arise at the edges of refined areas), and studied in detail using prototypes. Using the prototype developed in WARP, we demonstrated the effectiveness of mesh refinement in Particle-In-Cell simulation of a problem of interest, where a gain of almost four was obtained in computing time and memory requirement. An example of Chombo's three-dimensional field solve with automatic refinement around conductors leading to a complicated structure of refinement boxes was given as a preview of Chombo's capabilities to come. Typical gains of a factor of ten or more are expected with Chombo once discrete-particle support is fully integrated.

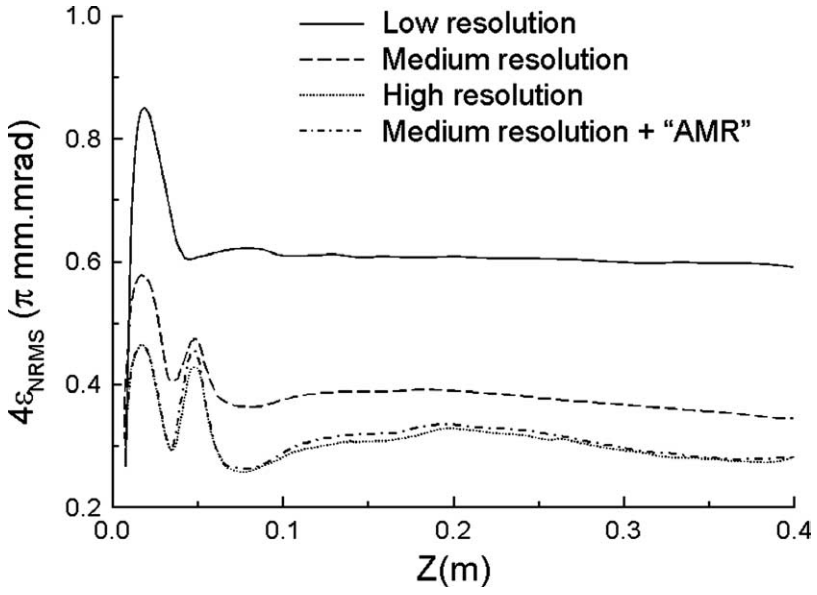

Fig. 4. Beam RMS emittance (a figure of merit for beam quality—the lower the better) as a function of z (numerical noise has been removed for clarity): the emittance converges downward with increasing resolution. The high resolution result is recovered with a run at medium resolution and AMR.

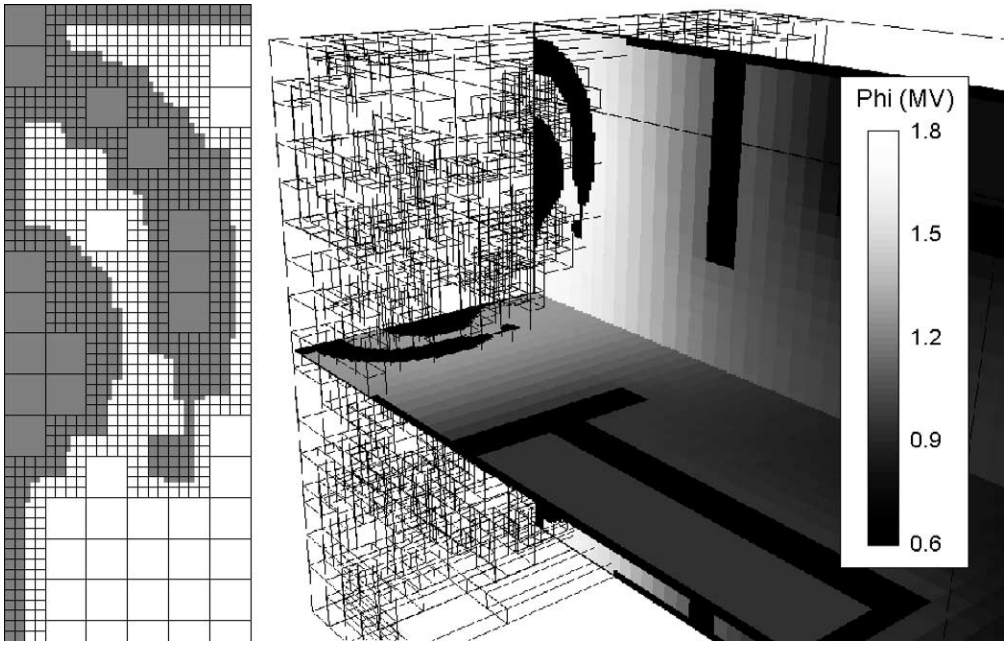

Fig. 5. Three-dimensional solution of the electrostatic potential in the HCX injector, as calculated by CHOMBO. A slice with the actual meshing (left) shows that the regions close to the boundaries of conductors (grey) are described with a finer mesh. The picture on the right shows a three-dimensional rendering that includes two orthogonal slices of the solution (with magnitude of electrostatic potential shown with a grey scale, conductors in black) and the edges of the different domains containing finer mesh spacing (in this case, mesh refinement covered the conductor edges only in the area surrounding the source).

References

[1] http://hif.lbl.gov.
[2] http://seesar.lbl.gov/ANAG/chombo.
[3] http://hif.lbl.gov/theory/WARP_summary.html.
[4] P. McCorquodale, P. Colella, D.P. Grote, J.-L. Vay, A node-centered local refinement algorithm for Poisson's equation in complex geometries, J. Comput. Phys., in press.
[5] J.-L. Vay, P. Colella, P. McCorquodale, B. Van Straalen, A. Friedman, D.P. Grote, Mesh refinement for Particle-In-Cell plasmas simulation: application—and benefits for—heavy ion fusion, Laser and Particle Beams 20 (2002) 569.
[6] http://hifweb.lbl.gov/webpages/experiments/HCX_summary.html.