Algorithm for Rapid Particle Tracing in Arbitrarily Mixed Meshes

ANDREAS HIEKE
Research Proteomics & Computational Physics
Ciphergen Biosystems, Inc.
6611 Dumbarton Circle, Fremont, CA 94555
USA
+1 (510) 505-2201

Abstract: - An algorithm is described which allows the rapid computation of trajectories of large numbers of particles in arbitrary meshes. Laplace (Poisson) and Navier-Stokes solutions are given at node points of a (FEM) mesh consisting of arbitrarily mixed tetrahedral, prismatic, pyramidal, or tetrahedral elements. The computation of the motion of a particle through such a domain requires, at each time step, knowledge of the enclosing element. Brute-force search routines would be prohibitively expensive with mesh data sets on the order of Gbyte, typical numbers of particles ranging from 10^3 to 10^5, and trajectory integration time steps from 10^3 to 10^7.

Key-Words: - algorithm, particle tracing, mesh, FEM, ANSYS, enclosing element, tetrahedral decomposition, search algorithm, trajectory, GEMIOS

1 Introduction

The area of protein research, known as “proteomics”, attempts to identify patterns of proteins (“biomarkers”) which correlate to disease states. ProteinChip arrays serve to selectively capture proteins prior to further detailed analysis in specialized mass spectrometers. A principal and critical step is the creation of ions of large organic molecules performed in laser based ion sources (‘MALDI’ sources; ¼ Nobel Price Chemistry 2002 [1]). Within these sources, ions are subjected to time dependent electromagnetic fields as well as collisions with intentionally introduced background gas molecules in order to reduce the ion temperature. The particle dynamics in such ion sources is extremely complex.
A simulation tool was needed which can provide insight into and a deeper understanding of ion extraction, collisional cooling and guidance processes in the presence of background gases

Therefore, an advanced multi-physics simulation system referred to as GEMIOS (Gas and Electromagnetic Ion Optical Simulator) has been implemented which allows the computation of 3D trajectories and energy exchange of ion beams/clouds under the influence of 3D time-dependent external electromagnetic fields in the presence of 3D rarefied gas flow fields and has therewith capabilities beyond existing tools [2]-[5].

The fundamentally novel aspect of GEMIOS is that it combines electromagnetic field solutions and fluid dynamic field solutions obtained within a given domain to compute charged particle trajectories. GEMIOS provides particle energies, collision rates, translational, thermal, and kinetic temperatures with spatial and temporal resolution. GEMIOS utilizes a combination of codes operating on the same input data set:

- a 3D Monte-Carlo Newtonian Motion and Collision module (MC-NMC), FORTRAN
- a 3D Finite Element (FEM) system [ANSYS-Multiphysics]
- a semi-statistical 2D DSMC (Direct Simulation Monte Carlo)

The focus of this paper is exclusively on the data exchange between the FEM module and the MC-NMC module as well as the critical problem of rapidly computing trajectories of particles through arbitrary FEM meshes.

2 Problem Formulation

The Finite Difference Method is particularly simple and can easily be implemented to obtain solutions

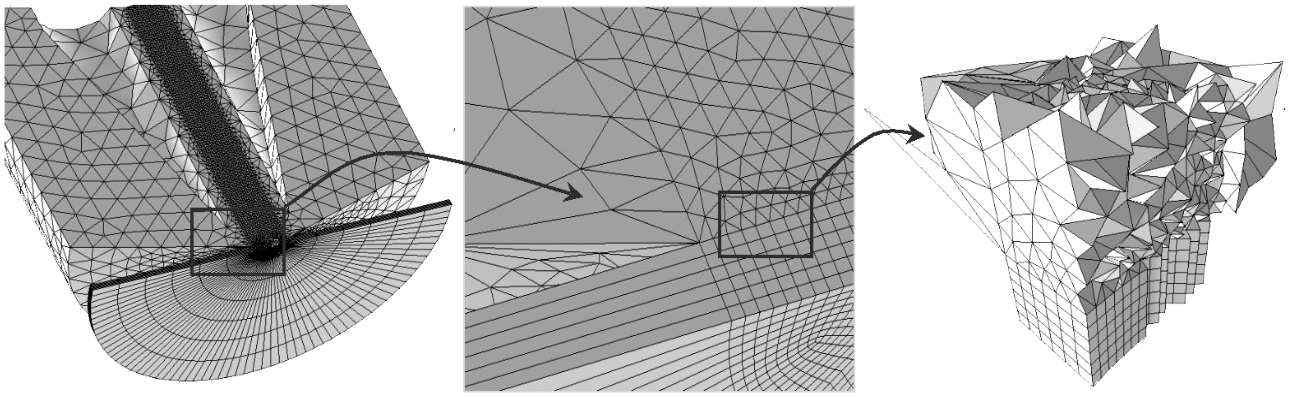

Fig. 1: Typical appearance of a FEM mesh. The large range of element sizes enables to accurately capture small feature in the model, e.g. regions with large electric field value. (model shown: tip of a quadrupole)

of PDEs (such as Laplace and Poisson equations). In addition, tracing particles through the resulting equidistant orthogonal meshes is trivial since based on the current location of a particle and the (constant) mesh size the indices of the enclosing element or cell can instantaneously be obtained. There are many particle tracing codes which take advantage of this simple approach, see for example [6].
However, complicated 3D configurations which contain relevant features on vastly different scales are preferably treated using FEM solvers since the computational requirements for FDM solvers (which are determined by the level of discretization which is determined by the smallest feature size in a model) typically become excessive in such cases.
Even though in general the computational effort per FEM element is much larger then per FDM cell, the total effort for a Laplace/Poisson problem can be lower since FEM meshes are shape-conform and may use a large range of element sizes.

The GEMIOS simulator which uses a FEM system for solid modeling and meshing and provides (on the same mesh):

- a data set of *nbs* orthogonal Laplace base solution (electric potentials and electric fields) (based on a set of canonical Dirichlet boundary conditions) from which any arbitrary static or dynamic electric field configuration is later obtained by superposition in the MC-NMC module
- a data sat of Navier-Stokes solutions for gas pressure, velocity, and temperature of a gas

Said solution data are provided at node points of first or second order elements as well as at element center locations. Each node and each element in a model has a unique number (Fig 2). The used second order (20 node) elements can assume 4 possible shapes: tetrahedral, prismatic, pyramidal, or tetrahedral. The nodes of each element ('*I*'..'*X*' in Fig. 3) are referenced in a specific order such that the actual shape of the element can be derived.

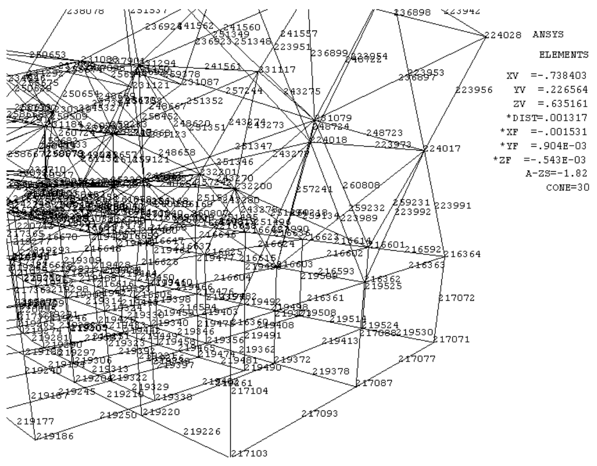

Fig. 2: Mesh elements and nodes are referenced by unique numbers determined by the FEM system

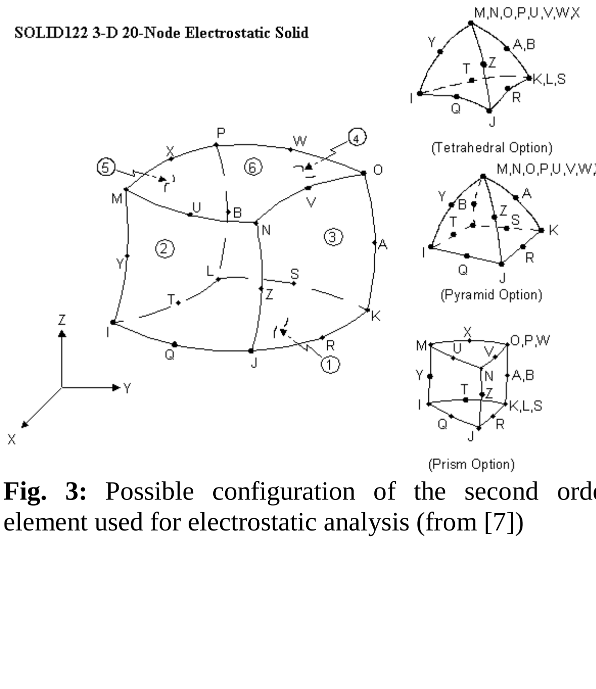

Fig. 3: Possible configuration of the second order element used for electrostatic analysis (from [7])

The computation of the motion of a charged particle through such a domain (trajectory integration + collision probability determination) requires, at each time step, knowledge of the enclosing nodal tetrahedron and the enclosing element since the provided field solutions are used to obtain interpolated values for electric and pneumatic fields. Specifically, the following questions arise:

- Which is the closest node to a given point in space?
- Which are the 4 closest nodes to a given point in space that form a tetrahedron which encloses this point?
- Which element encloses a given point?

The naive approach to find the coordinates and number of the node $\vec{r}_i = [x_i, y_i, z_i]$ closest to a given location $\vec{r} = [x, y, z]$ would be a brute force search over all nodes *nn* in the domain:

```
drmin=1D99
node_drmin=0

 DO i=1,nn
        dx=x-xn(i)
        dy=y-yn(i)
        dz=z-zn(i)
        dr2=dx*dx + dy*dy + dz*dz
         IF (dr2.LT.drmin) THEN
                           drmin=dr2
                           node_drmin=i
                         ENDIF
 ENDDO
```

Fig. 4: Simple code for brute force search of node *node_drmin* closest to point [*x*,*y*,*z*]

With node numbers on the order of 10^5 to 10^7 such an approach is obviously impractical to be applied more than a few times during a simulation due to the required CPU times. For example, on a 2.8GHz Xeon system one measures the following execution times:

$nn = 10^6 \Rightarrow t_{CPU}$=36ms

$nn = 10^7 \Rightarrow t_{CPU}$ =390ms

Typically, numbers of particles range from 10^3 to 10^5 with 10^3 to 10^7 integration time steps for each trajectory. Therefore, it was necessary find an algorithm which can answer above-mentioned questions within a few 10^{-6}s.

3 Problem Solution

Fundamentally, there are two approaches to reduce the required CPU time:

- computing as much as possible outside the actual tracing routine (avoiding repetitions)
- buying speed with memory (using and storing knowledge about the mesh)

Both approaches are being used in the MC-NMC code as outlined below. The FEM system provides the following, minimally required data:

- the number of nodes *nn*
- the number of elements *ne*
- a [*ne* x 20] matrix $NOE_{[1..ne,\ 1..20]}$ containing the node numbers for each element
- 3 *ns* long vectors $x_{1..ns}$ $y_{1..n}$ $z_{1..n}$ containing the Cartesian coordinates for each node
- a [*ne* x 6] matrix $EE_{[1..ne,\ 1..6]}$ containing the numbers of the elements with which a given element shares faces
- 3 *ne* long vectors $xe_{1..ne}$ $ye_{1..ne}$ $ze_{1..ne}$ containing the Cartesian coordinates for each element center
- solution data: Laplace base solutions Φ on nodes as well as electric field $\vec{E}$ and Navier-Stokes solutions data $\vec{v}$, *p*, *T* at element centers

Step 1: The tracing algorithm starts by reading these data in an OS independent manner via ASCII files in order to enable that the FEM code and the MC-NMC code can run on entirely different machines, provided they can share one file system.

Step 2: Before the actual tracing starts the algorithm builds, based on these mesh data,

- a [*nn* x *200*] matrix (lookup table) to find all elements attached so each node and
- a *ne* long vector containing the characteristic length of each element (space diagonal).

The required computational effort is on the order of a few seconds and small in comparison to the actual tracing.

Step 3: The node n_{start} closest to the initial position $\vec{r}_{start}$ is determined. This is a brute force search but executed only one time since n_{start} is identical for all particles. Also, all elements attached to node n_{start} are checked which of them contains a tetrahedron formed by its nodes which encloses $\vec{r}_{start}$.

Step 4: During the actual tracing the algorithm goes for all particles through the following routine assuming that in step 3 the enclosing nodal tetrahedron and the enclosing element for $\vec{r}_{start}$ at *t*=0 was found:

```
(0)   Particle position at $\vec{r}(t=0)$
DO
(1)   One step of trajectory integration and
      collision modeling (field values interpolated
      from neighboring elements).
(2)   New particle position $\vec{r}(t) \leftarrow \vec{r}(t+\Delta t)$
                                   $t \leftarrow t+\Delta t$
(3)   Find new enclosing tetrahedron and enclosing
      element:
(3.1)     · 1st guess:  the same tetrahedron
(3.2)     · 2nd guess: another tetrahedron within the
            same element (dependent on its shape)
(3.3)     · 3rd guess: the element connected to the
            closest node with element center location
            closest to $\vec{r}(t)$
(3.4)     · 4th guess: All elements connected the
           corner  nodes of the old enclosing element
           except the one tested as 3rd guess
(4)  IF no enclosing element found EXIT LOOP
     (the particle left the domain)
ENDDO
```

The inherent assumption is that at least one time step is executed within each element which is being ensured by the trajectory integration routines.
The determination of the enclosing nodal tetrahedron serves two purposes:

- it provides a way to determine the enclosing element via tetrahedral decomposition [10]
- it provides a simple means to obtain a local electric field vector approximation based on electric potentials on 4 nodes (for low accuracy tracing only)

It is by far sufficient to use only first order nodes (letter *I* through *P* in Fig. 3) for the tetrahedral decomposition which reduces the number of possible configurations significantly.
Thus, there are maximal five test required to find the enclosing tetrahedron in a given element as illustrated in Fig.5. The used tetrahedral decompositions are not unique but adequate since the primary purpose is to identify the enclosing

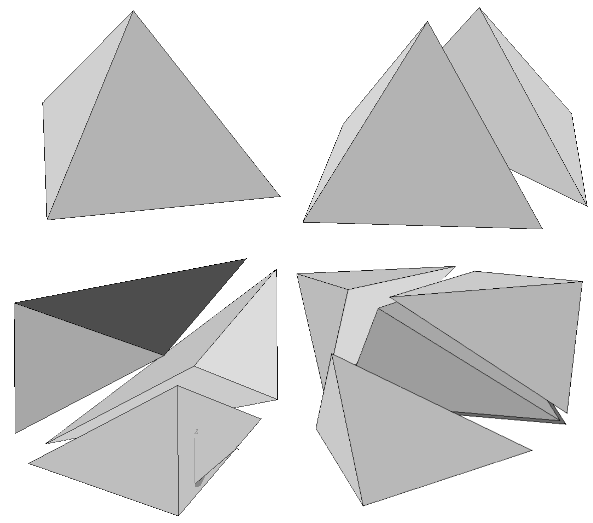

Fig. 5: Possible tetrahedral decomposition dependent on shape of element using only first order nodes: upper left: tetrahedron (1 test), upper right: pyramid (2 tests), lower left: prism (3 tests), lower right: hexahedron (5 tests)

element. (It is unnecessary to determine a possible optimal decomposition scheme since the high accuracy tracing routines interpolate solutions from neighboring elements.)
The determination if a point $\vec{r}$ = [*x*,*y*,*z*] is inside a tetrahedron formed by nodes at $\vec{r}_1 = [x_1, y_1, z_1]$, $\vec{r}_2 = [x_2, y_2, z_2]$, $\vec{r}_3 = [x_3, y_3, z_3]$ and $\vec{r}_4 = [x_4, y_4, z_4]$ can be made as follows: The volume of the tetrahedron is given by

$$V_{tet} = \left| \det \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix} \right| \tag{1}$$

If $\vec{r}$ is in fact inside the tetrahedron it decomposes it into 4 smaller tetrahedrons with the sum of their volumes equal to the volume V_{tet}. Allowing for some finite numerical error ε equation (2) provides a directly implementable formulation as the Fig. 6 shows. This test can be executed in a few 10^{-7}s.

$$\left| -\left|\det\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}\right| + \left|\det\begin{bmatrix} x & y & z & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}\right| + \left|\det\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x & y & z & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}\right| + \left|\det\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x & y & z & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}\right| + \left|\det\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x & y & z & 1 \end{bmatrix}\right| \right| < \varepsilon \tag{2}$$

```
D= ABS(  x1*y2*z3 - x1*y2*z4 - x1*y3*z2 + x1*y3*z4 + x1*y4*z2 - x1*y4*z3   +&
       - x2*y1*z3 + x2*y1*z4 + x2*y3*z1 - x2*y3*z4 - x2*y4*z1 + x2*y4*z3   +&
       + x3*y1*z2 - x3*y1*z4 - x3*y2*z1 + x3*y2*z4 + x3*y4*z1 - x3*y4*z2   +&
       - x4*y1*z2 + x4*y1*z3 + x4*y2*z1 - x4*y2*z3 - x4*y3*z1 + x4*y3*z2      )

D1=ABS(  x *y2*z3 - x *y2*z4 - x *y3*z2 + x *y3*z4 + x *y4*z2 - x *y4*z3   +&
       - x2*y *z3 + x2*y *z4 + x2*y3*z  - x2*y3*z4 - x2*y4*z  + x2*y4*z3   +&
       + x3*y *z2 - x3*y *z4 - x3*y2*z  + x3*y2*z4 + x3*y4*z  - x3*y4*z2   +&
       - x4*y *z2 + x4*y *z3 + x4*y2*z  - x4*y2*z3 - x4*y3*z  + x4*y3*z2      )

D2=ABS(  x1*y *z3 - x1*y *z4 - x1*y3*z  + x1*y3*z4 + x1*y4*z  - x1*y4*z3   +&
       - x *y1*z3 + x *y1*z4 + x *y3*z1 - x *y3*z4 - x *y4*z1 + x *y4*z3   +&
       + x3*y1*z  - x3*y1*z4 - x3*y *z1 + x3*y *z4 + x3*y4*z1 - x3*y4*z    +&
       - x4*y1*z  + x4*y1*z3 + x4*y *z1 - x4*y *z3 - x4*y3*z1 + x4*y3*z       )

D3=ABS(  x1*y2*z  - x1*y2*z4 - x1*y *z2 + x1*y *z4 + x1*y4*z2 - x1*y4*z    +&
      - x2*y1*z  + x2*y1*z4 + x2*y *z1 - x2*y *z4 - x2*y4*z1 + x2*y4*z    +&
       + x *y1*z2 - x *y1*z4 - x *y2*z1 + x *y2*z4 + x *y4*z1 - x *y4*z2   +&
       - x4*y1*z2 + x4*y1*z  + x4*y2*z1 - x4*y2*z  - x4*y *z1 + x4*y *z2      )

D4=ABS(  x1*y2*z3 - x1*y2*z  - x1*y3*z2 + x1*y3*z  + x1*y *z2 - x1*y *z3   +&
       - x2*y1*z3 + x2*y1*z  + x2*y3*z1 - x2*y3*z  - x2*y *z1 + x2*y *z3   +&
       + x3*y1*z2 - x3*y1*z  - x3*y2*z1 + x3*y2*z  + x3*y *z1 - x3*y *z2   +&
       - x *y1*z2 + x *y1*z3 + x *y2*z1 - x *y2*z3 - x *y3*z1 + x *y3*z2      )

sum_D_parts=D1+D2+D3+D4


IF ((abs(D-sum_D_parts)) .LT. 1D-14) THEN
                                  tetrahedron_encloses=.TRUE.
                                  enclosing_tetrahedron_in_this_element=.TRUE.
                        ENDIF
```

Fig. 6: Code according to equation (2) to test if point $[x,y,z]$ is enclosed a tetrahedron formed by nodes at $[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$, $[x_3, y_3, z_3]$, and $[x_4, y_4, z_4]$ which can be executed in a few 10^{-7}s

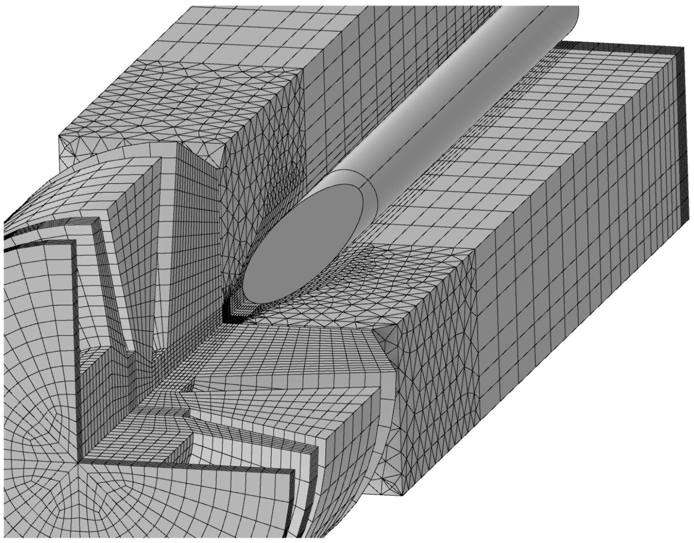

Fig. 7: Meshed FEM domain of a 5-electrode ion generation system coupled to a quadrupole using a large range of element sizes and different element shapes (3/4 of the model + one quadrupole rod are shown)

The described algorithm reduces the CPU time required for finding the next enclosing element during tracing to values equal or smaller than the CPU time necessary for trajectory integration and collision treatment. Specifically, this includes:

- temporal updates of time-dependent electric fields from orthogonal Laplace base solutions,
- interpolation of fields using neighboring elements,
- determination of local ion/gas molecule collision probability (incl. Monte Carlo temperature sampling)
- collision treatment: determination of post-collisional ion velocity vector incl. Monte Carlo sampling of scattering angles

Slightly dependent on the model configuration and tracing parameters, the CPU time for one such trajectory integration time step including updating the enclosing nodal tetrahedron and enclosing element is on average $t_{CPU}=5\cdot10^{-6}$s … $8\cdot10^{-6}$s on a XEON 2.8GHz system. This allows, for example, to compute 10^3 trajectories each with 10^4 time steps in less than one minute.

Details of the multi-stage Monte Carlo collision treatment are beyond the scope of this paper and will be reported elsewhere. Fig. 7 shows a practical example of a complex FEM mesh used to model an ion source coupled to a RF quadrupole ion quide. Noteworthy in terms of tracing is the considerable range of element sizes in combination with mixed element shapes. Using the discussed algorithm, Fig. 8 depicts computed ion trajectories through the domain shown in Fig. 7 based on time-dependent electric as well as pneumatic fields.

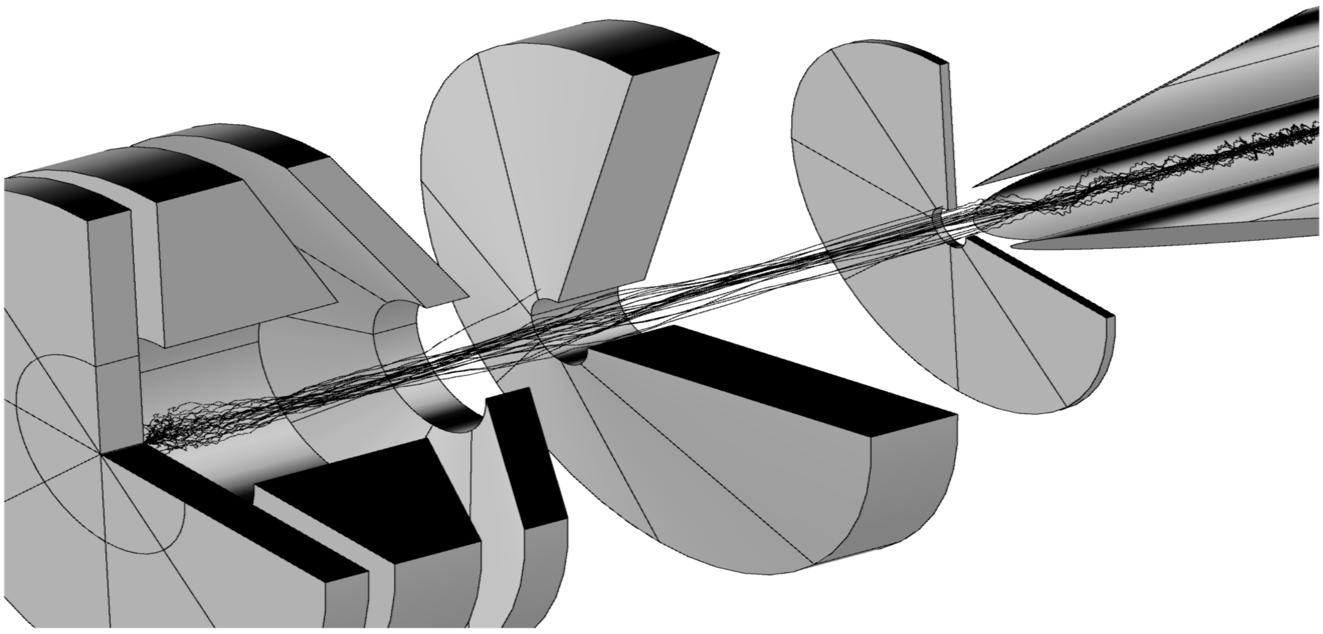

Fig. 8: Simulation of ion trajectories through the domain shown in Fig. 7 using the described tracing algorithm and based on simultaneous consideration of electric DC and RF fields and gas flow fields; ¾ of the solid model is shown

4 Conclusion

An algorithm has been described which allows the rapid computation of particle trajectories through mixed tetrahedral, prismatic, pyramidal, and tetrahedral meshes. It is conceivable to further accelerate this algorithm by building an additional lookup table which contains for each node the number of the element with its center location closest to the node location.

Currently, this element is determined as part of the 2nd guess routine. However, for the particular application discussed here, the possible acceleration is negligible in comparison to the constant computational effort required for each time step. If the GEMIOS code were to be used for tracing in electromagnetic fields only (no collisions) the suggested additional lookup table would be beneficial.

References:

[1] http://www.nobel.se/chemistry/laureates/2002/

[2] Andreas Hieke: “GEMIOS – a 64-Bit multi-physics Gas and Electromagnetic Ion Optical Simulator”, Proceedings of the 51st Conference on Mass Spectrometry and Allied Topics, Montreal, Canada, June 2003

[3] Andreas Hieke: “Theoretical and Implementational Aspects of an Advanced 3D Gas and Electromagnetic Ion Optical Simulator Interfacing with ANSYS Multiphysics”, International Congress on FEM Technology 2003, Potsdam, Germany, November 2003

[4] Andreas Hieke: “Development of an Advanced Simulation System for the Analysis of Particle Dynamics in LASER based Protein Ion Sources”, Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show, Boston, MA, March 2004, Vol. 1, page 180-184, nanotech2004.com

[5] Andreas Hieke: "3D electro-pneumatic Monte-Carlo simulations of ion trajectories and temperatures during RF quadrupole injection in the presence of gas flow fields", Proceedings of the 52nd ASMS Conference on Mass Spectrometry , Nashville, TN, May 2004

[6] www.simion.com

[7] ANSYS Inc., Theory Reference Manual V8.0 www.ansys.com

[8] G. Ramos, W. Enright: “Interpolation of Surfaces over Scattered Data”, Proceedings of IASTED Conference on Visualization, Imaging and Image Processing VIIP’2001, Marbella, Spain, September 3-5 2001

[9] Isaac Amidror: “Scattered data interpolation methods for electronic imaging systems: a survey”, Journal of Electronic Imaging, Vol. 11 No. 2, April 2002, pp. 157-176

[10] David N. Kenwright and David A. Lane: “Optimization of Time-Dependent Particle Tracing Using Tetrahedral Decomposition”, Proceedings of the 6th conference on Visualization, IEEE Visualization , 1995, pp. 321-328