SIMION Appended Course Notes

David J. Manura
(c) Scientific Instrument Services, Inc.
2010-11-11/2005-05

1 Abstract

These notes can be used as part of an introductory course or lecture on SIMION. The notes are designed to help the user understand the *underlying concepts* in SIMION so that the user has a firm foundation to stand on when later using or learning to use SIMION. What SIMION can do and how it does it are overviewed. These notes do not provide a step-by-step tutorial on using SIMION; that may be done later. These notes reflect SIMION version 8.0 (8.0.5).

2 Points covered

- SIMION GUI
- Workbench concept (IOB file)
    - Contains PA and PA0 instances, REC, FLY/2, and ION files
    - Supports multiple PAs, e.g. overlapping electrostatic and magnetic, optimization, and higher density volumes. (chapter 9)
    - Questions:
        - What are the issues with having multiple PA instance rather than a single one?
        - What are the instance selection rules?
- Coordinate system : azimuth, elevation
    - Units: gu, mm, in
    - Workbench and PA coordinates
    - Orienting PAs and ions
- View
    - Viewing, rotating, cuttings of 3D models, asymmetric zooming
    - Displaying position/potential/gauss at the current cursor location (note: averaging effect in 3D)
    - Grid units (gu) and mm units / orientations
    - Questions:
        - Demonstrate viewing and cutting to see inside a system.
- Contours
    - Potential and gradient contours
    - Auto, use, 3D functions
    - Questions:
        - What are the two different types of contours supported in SIMION?
        - How does one define contours?
- Potential energy maps
    - Viewing, zooming
    - Positive/negative ion
    - 2D v.s. 3D
    - Questions:

        - A potential energy map is 2D, but a system is usually 3D. How are potential energy maps used in 3D systems?
- Definition ions
    - Ways to define ions: by groups and individually or by .lua program.
    - By groups: .FLY/2 file
        - Using deltas
    - Individually: .ION file
    - By .lua
    - Orientating in PA or workbench cords
    - Questions:
        - When are .ION, .FLY/2, and .lua files suitable? What are the advantages/disadvantages of each?
- Space-charge
    - Capabilities and limitations
    - Questions:
        - What are the limitations of SIMION's space-charge methods?
- Data recording
    - Recording to screen or to file
    - The concept: what, when, and how
        - E.g. crossing a test plane
    - Formatting: excel output, precision, etc.
    - Lua techniques
    - Questions:
        - How do you record data for import into Excel?
- PA, PA#, and other files
    - PA# files – 1V, 2V, …
    - Visually inspect the PA#, PA1, PA2, PA0, .. files
    - Grid points v.s. solid electrode points (5-5)
    - Questions:
        - What is the difference between .PA and .PA# files? When are these suitable? What are the issues in each?
- Refining files
    - Solves the Laplace equation
    - Questions:
        - What does refining do?
- Fast adjusting
    - What this means
    - Accessible from two locations: from main screen and from workbench
    - Questions:
        - What does "fast adjusting" or "fast scaling" mean?
        - What two places can you fast adjust from?
- Issues with saving files: PA0, PA, PA#, IOB (when saving is necessary) + auto-loading .FLY/2 and .REC files
    - Values not retained in the .IOB file (e.g. view settings, computational quality.
    - Questions:

        - Which of these files can meaningfully be placed into a workbench: .PA, .PA#, .PA0, .PA1?
        - How is a PA0 file created?
        - How is a PA1 file created? When is a .PA2 file created?
- Magnetic PAs
    - Limitations of
    - ng factor
    - setting via Lua program (measured magnetic field)
    - Questions:
        - What capabilities does SIMION have for magnetic fields?
        - What is the ng factor?
        - What advantage does defining a magnetic field in a Lua program have?
- Trajectory calculation
    - Runge-Kutta
    - Variable time steps
    - Adjusting time steps – via computational quality (advancedcourse-3), "time markers". Reasons to-voltage changes or RF.
    - Observation of time steps in Einzel lens
    - Questions:
        - What factors adjust time steps? What is the "default" time step? (Appendix H)
        - How to you determine or see what time steps are being used?
- Lua programs
    - Capabilities
        - Randomizing ions via user programs
        - Easy voltage control
        - Extending the span of user program control
        - Data recording – mark, message
        - Controlling time steps
        - Modifying ion motions
        - Viscous or collisional cooling
        - Tuning (e.g. _Tune example)
        - Ion jumping tricks—(advanced course)
    - Segments
    - Variables + types of variables
    - Tricks
    - Debugging
    - Questions:
        - What can Lua programs do?
        - What are the differences between static and adjustable variables?
        - What segments and variables are available to use?
- Ways to create geometries:
    - Modify, GEM, CAD import, SL Libraries
        - Q: When are each suitable? What are the advantages/disadvantages of each?

    - CAD import
    - Modify function
        - The concept – filling and cutting
        - Drawing electrode and non-electrode points at certain voltages and shapes
        - Changing dimensions and symmetry/mirroring
        - Find function – e.g. changing points from non-electrodes to electrodes – note: trick in generating linear reflectron
    - GEM files
        - Creating a PA file from a GEM file
        - Debugging – GeomF button in Modify
        - Scaling in physical units rather than grid units.
        - Writing a program to interpolate variables into GEM files
        - Questions:
            - Explain what a .GEM file is.
            - How does one debug a .GEM file?
- Files: PA*, IOB, FLY2/FLY/ION, TRJ, REC, GEM, Lua, STL
    - Question: what are these files?
- Limitations of SIMION
    - Space-charge
    - Open systems
    - Grid accuracy (e.g. 5-2)
    - Time step accuracy
    - Frequency: static, low (quasistatic), high
    - ~200M points (max 2-4GB) in 8.0, >4GB in 8.1
    - Questions:
        - Identify some limitations of SIMION.
        - How much memory does a 50M point array require?
- Managing memory
    - Process manager, swap file, 8 bytes/point
- Common problems
    - Maximum PA or ion limit
    - mm/gu scale factor
    - Differences between PA/PA#/PA0
    - Ion trajectories display slows down computation; ion trajectory recording consumes too much disk space
    - See FAQ page for more
- Windows tricks
    - Copy to clipboard (Print Screen key)
    - Drag and drop files from Windows Explorer into text editor
    - Bugs: File directory size limit + initializing random number generator
    - Setting custom text editor
- Other tricks
    - Running two or more instances of SIMION
- Definitions
    - Laplace

    - Fringing effect
    - Runge-Kutta
- Theory
    - Laplace + Refining
    - Runge-Kutta
- Resources
    - simion.com – FAQ, tutorials, papers
    - manual + course notes

3 Tasks

- Quick tour of SIMION
- Introduce main concepts + special topics
- Go through lab sessions
- Review of material
- Questions

4 What SIMION Does (A Brief Overview)

- Calculates electric fields and to some extent magnetic fields.
- Calculates the trajectories of charges particles through electric and/or magnetic field, including time-dependent fields.

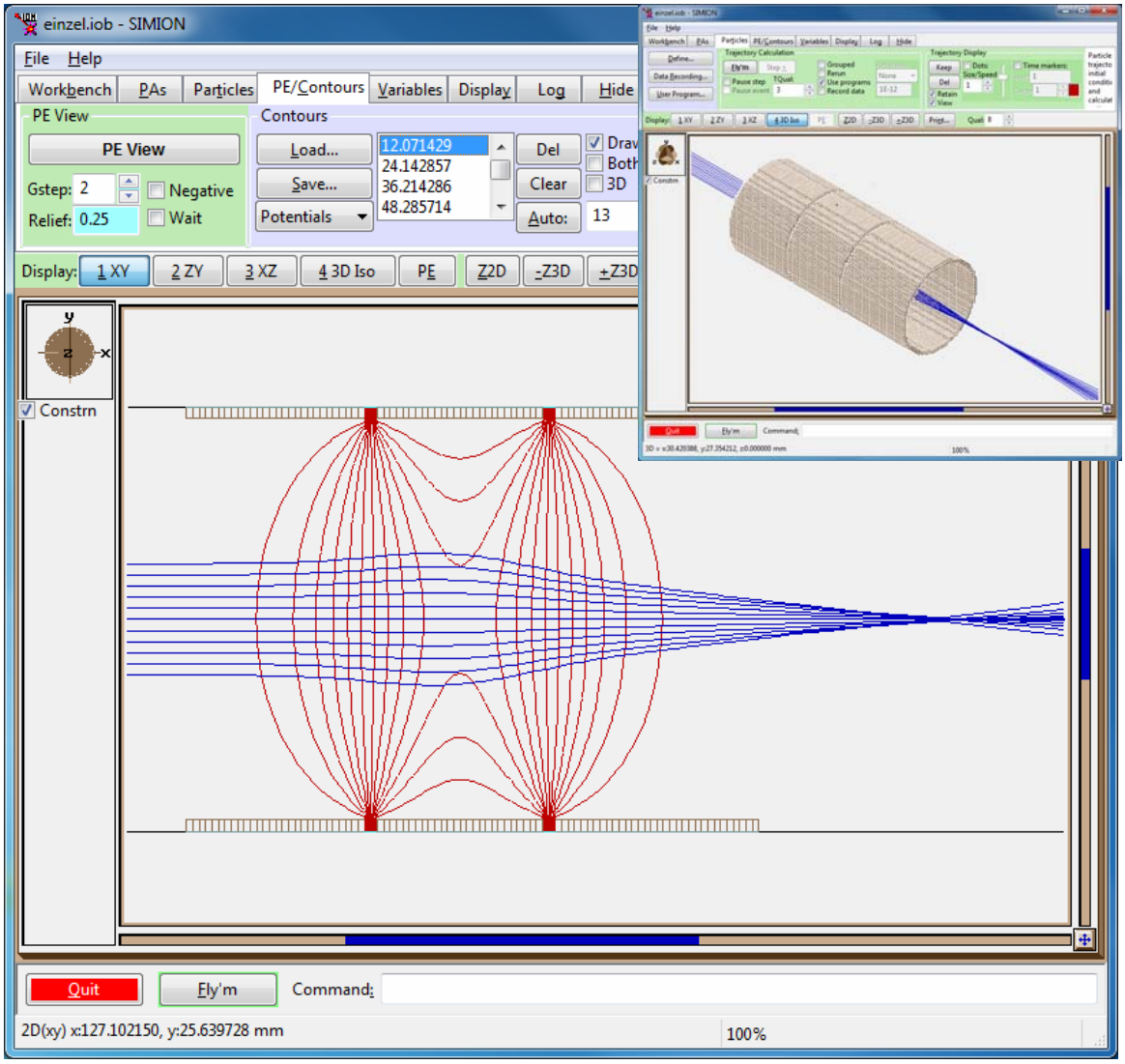


Figure 1: Screenshots of SIMION simulation of three electrode einzel lens, with calculated electric field (red lines of equipotential) and calculated particle trajectories (blue lines).

4.1 SIMION calculates electric fields (E)

As a simple example (below), consider a long metal box that is open on the ends and has different voltages on each of the four sides. We wish to calculate the electric field in the center cross section (green). In this simple example, we will assume the box is effectively infinitely long so that the problem can be solved in 2D.

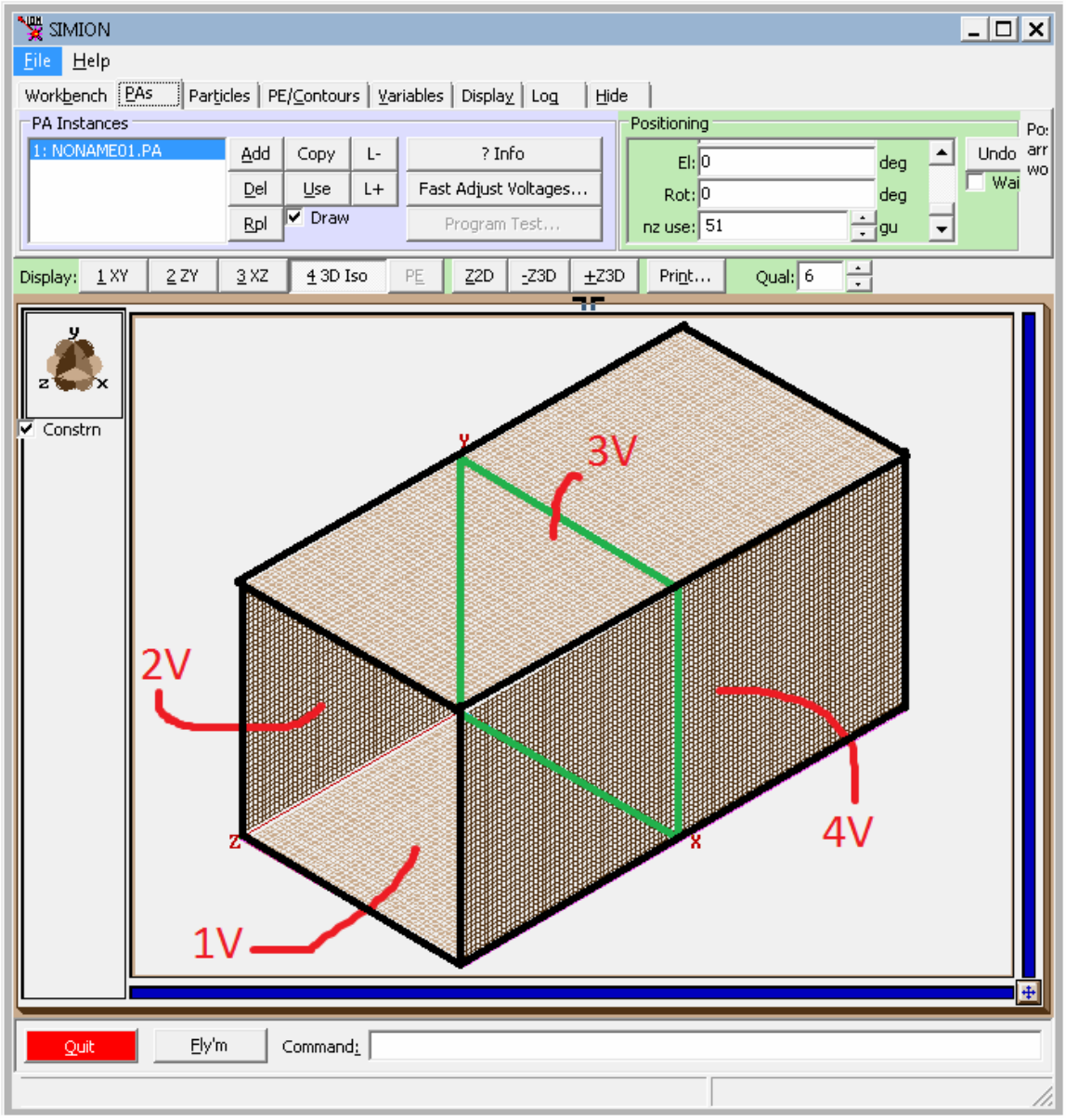


Figure 2: Simple 2D system of electrodes.

SIMION represents an electric field via a "potential array", which is a 2D or 3D grid of evenly spaced data points. Each data point contains the potential at that location. The above system can be represented as a potential array like this:

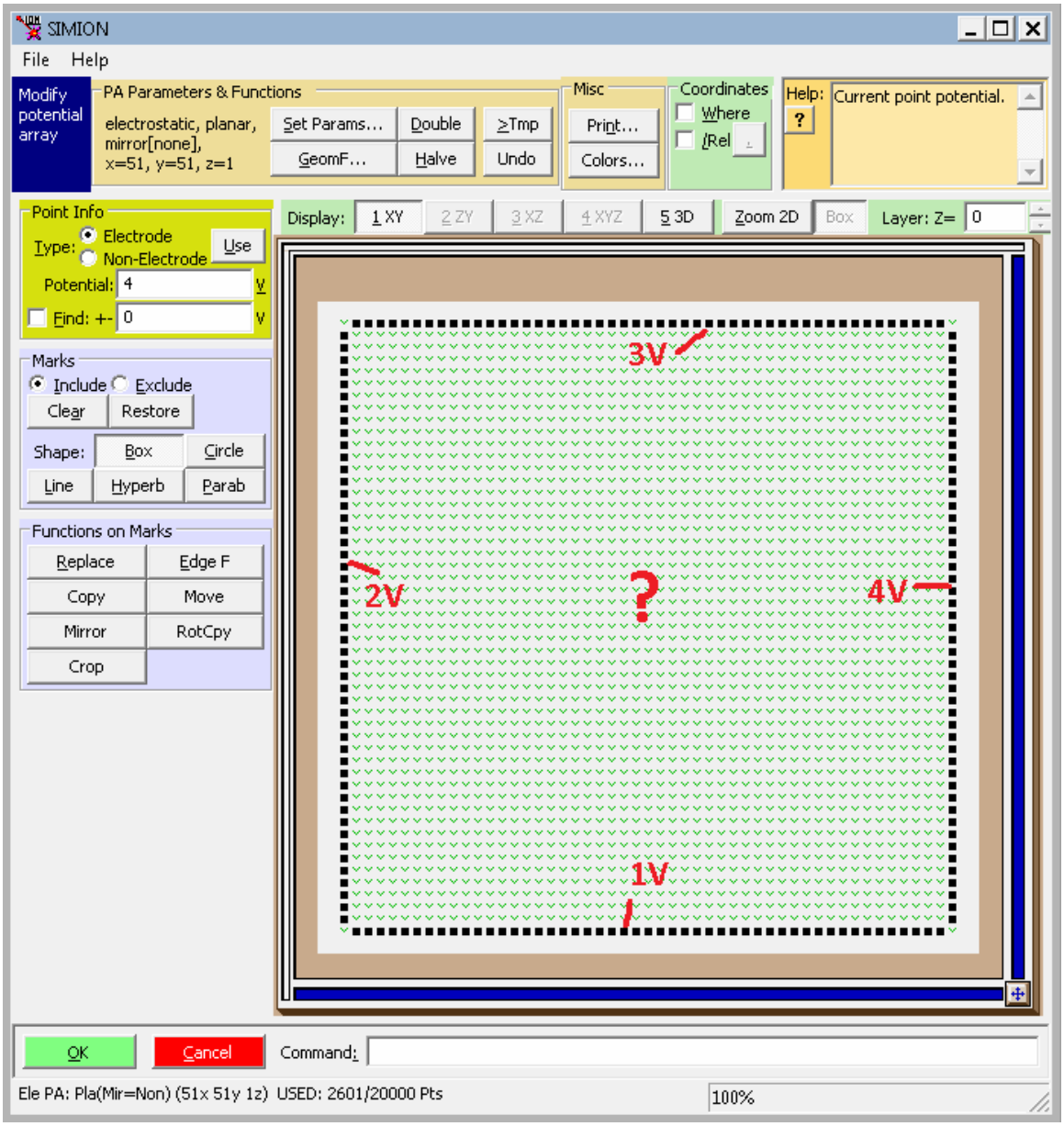


Figure 3 – View of a 2D SIMION potential array.

The potentials at some points (usually electrode surfaces) are known by the user and entered as electrode points (solid black). The potentials of all other points (green) are calculated by SIMION.

Once SIMION calculates unknown potentials ("refines" the array in SIMION lingo), SIMION can plot the potential array in various ways.

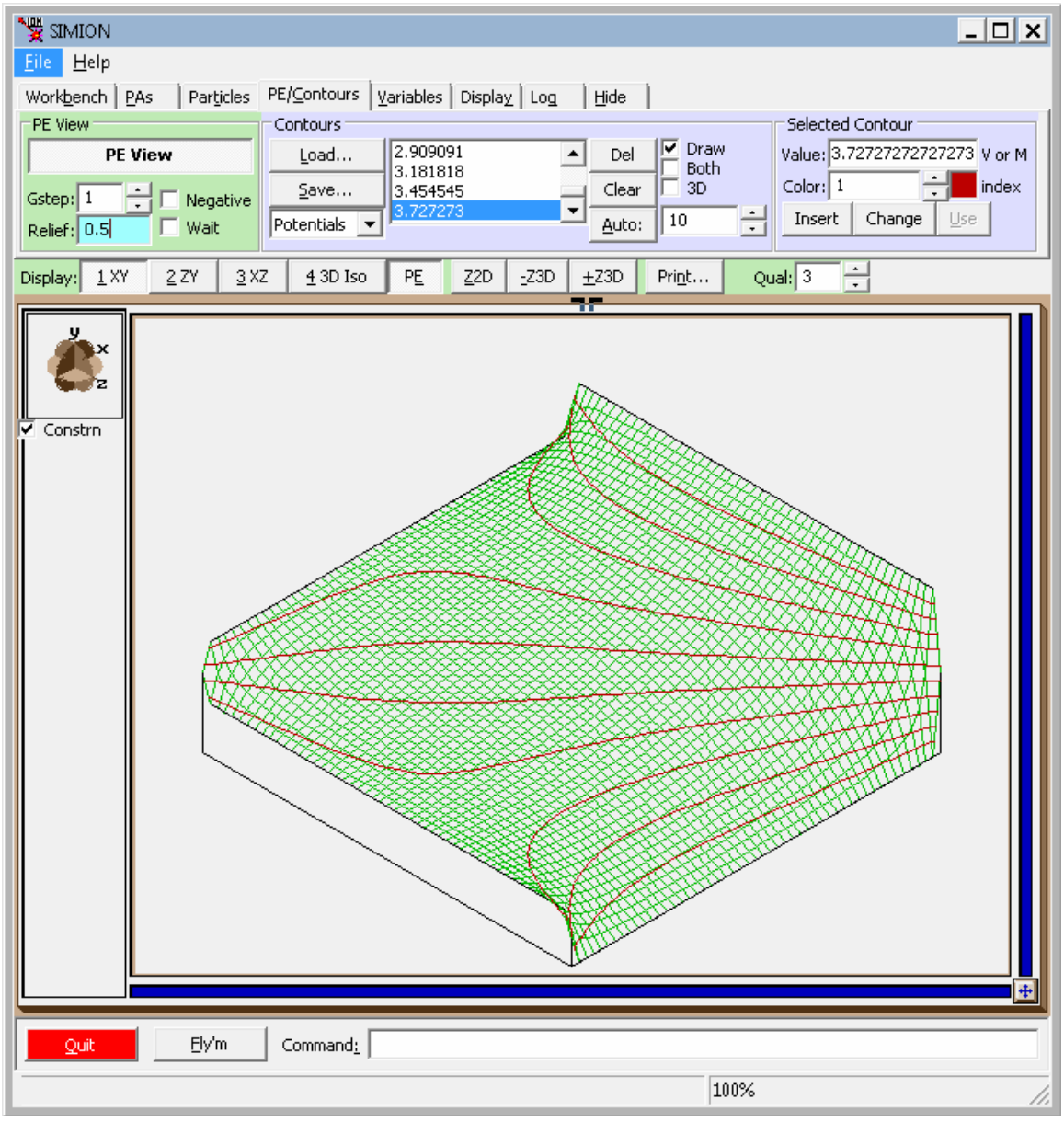


Figure 4: SIMION potential energy map view of potential array.

In the above figure, the vertical direction represents the potential at each corresponding point in the array.

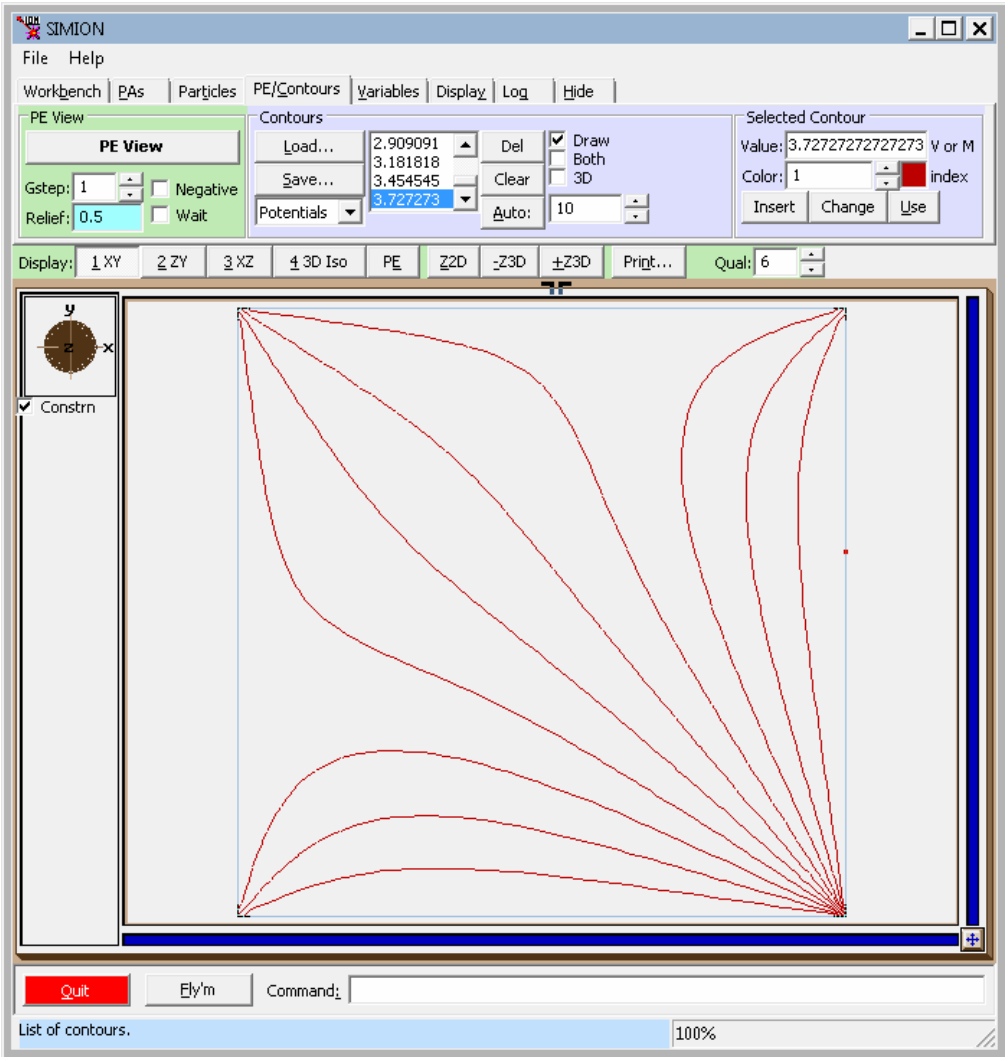


Figure 5: SIMION plot of equipotential lines in potential array.

Lines of equipotential (which are perpendicular to field lines) can be plotted as well. The electric field is determined directly from the potential V(x). $\mathbf{E} = -\nabla V$

4.2 SIMION calculates the trajectories of charged particles through fields.

Now that we have an electric field fully known, we often want to see how particles move through that field.

If we tell SIMION the initial conditions on a set of particles (e.g. position, velocity, mass, and charge) as time t=0 in the presence of the previously calculated electric field, SIMION can calculate the states of those particles in the future (t > 0).

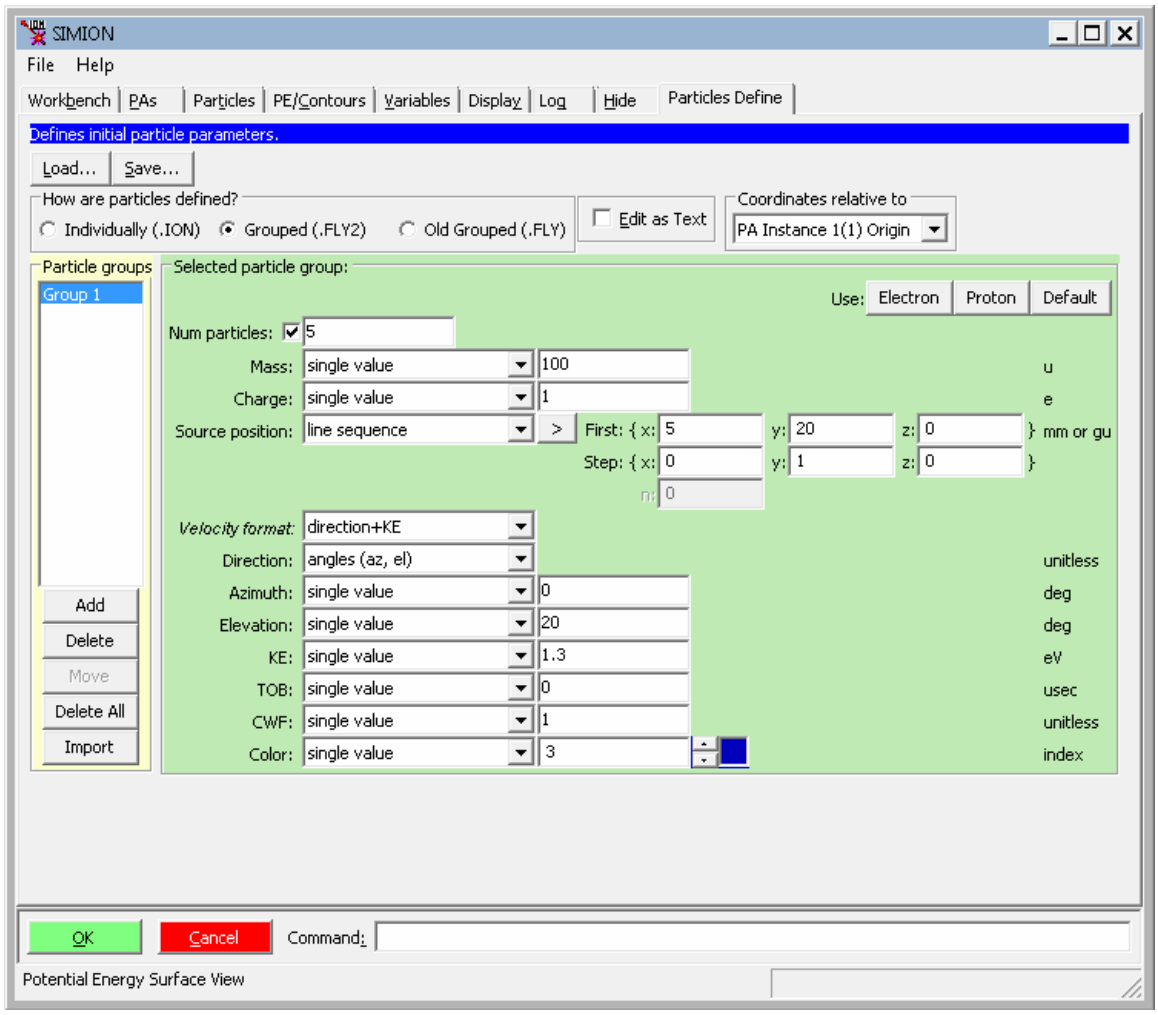


Figure 6: Define initial positions, velocities, charges, and masses of particles.

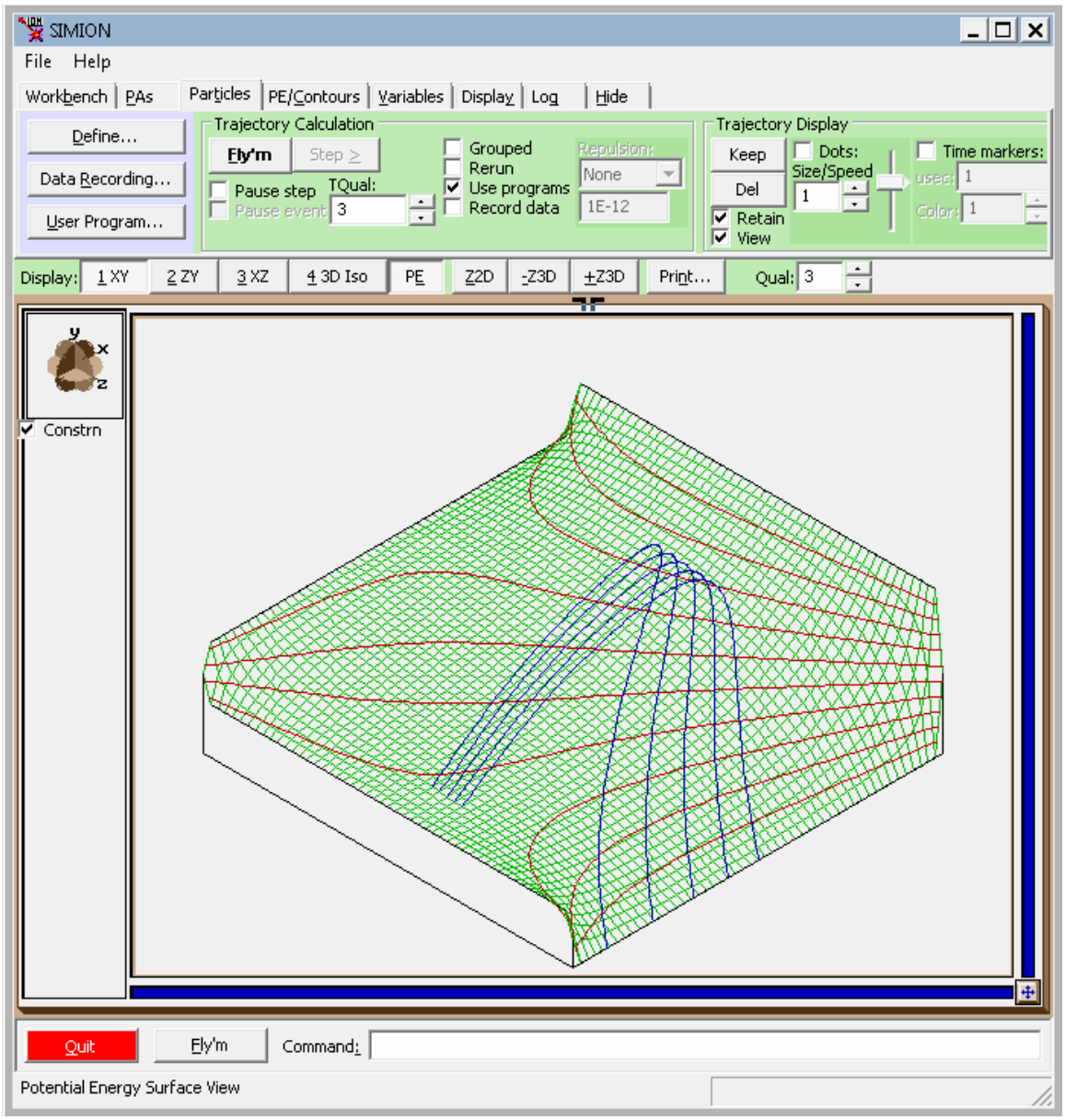


Figure 7: Ion trajectories calculated by SIMION and superimposed on the potential energy map.

SIMION promotes intuition (e.g. the ions are balls, and the potential energy map is a hill); it is clear now why the ions move as they do. Positive ions starting at the left approaching and being repelled by the more positive plate on the right.

What else can SIMION do?

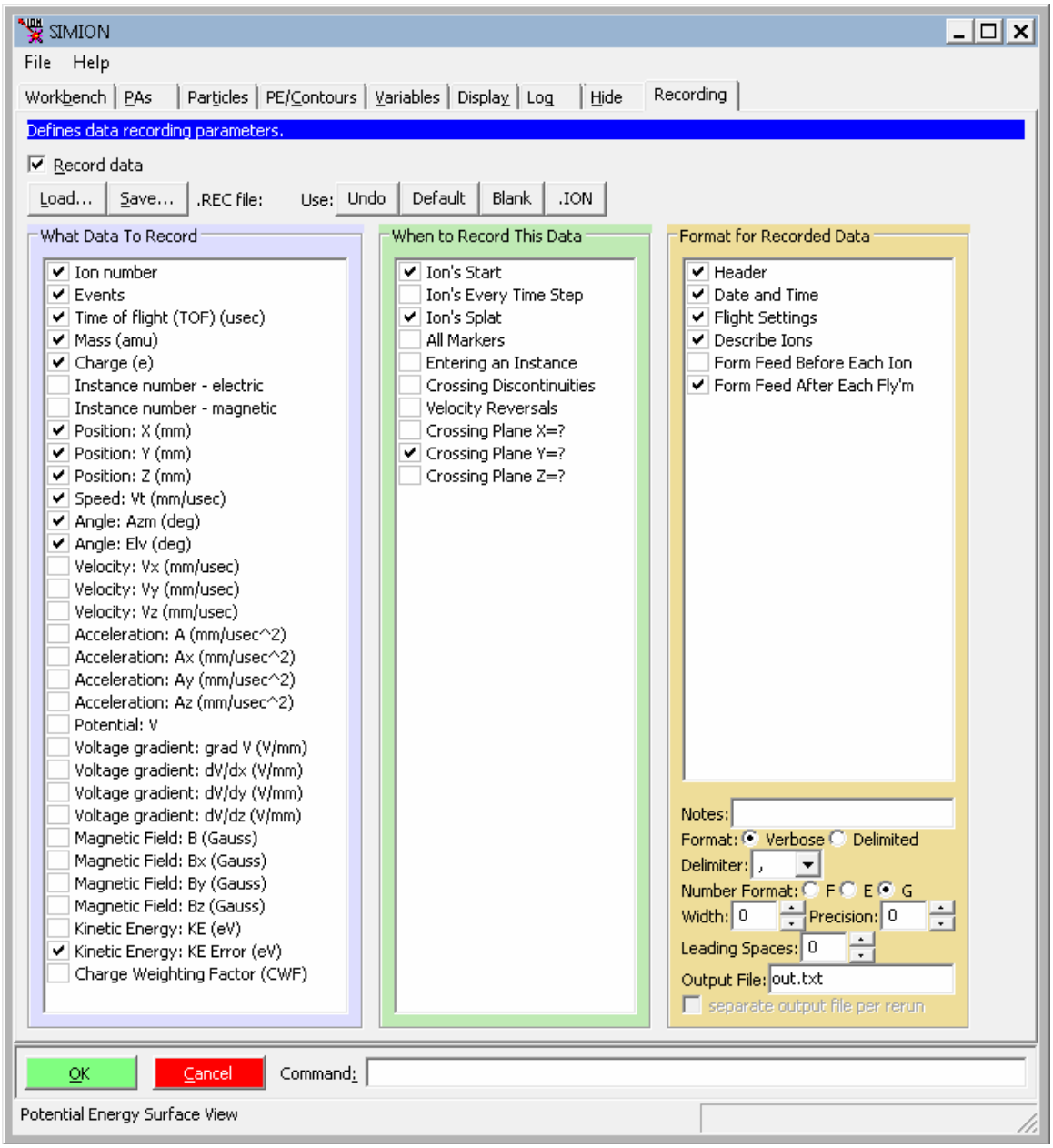


Figure 8: Define data to record

SIMION's data recording options can be used to output data on particles over time. For example, you can record particle termination positions to a text file for further analysis in Excel.

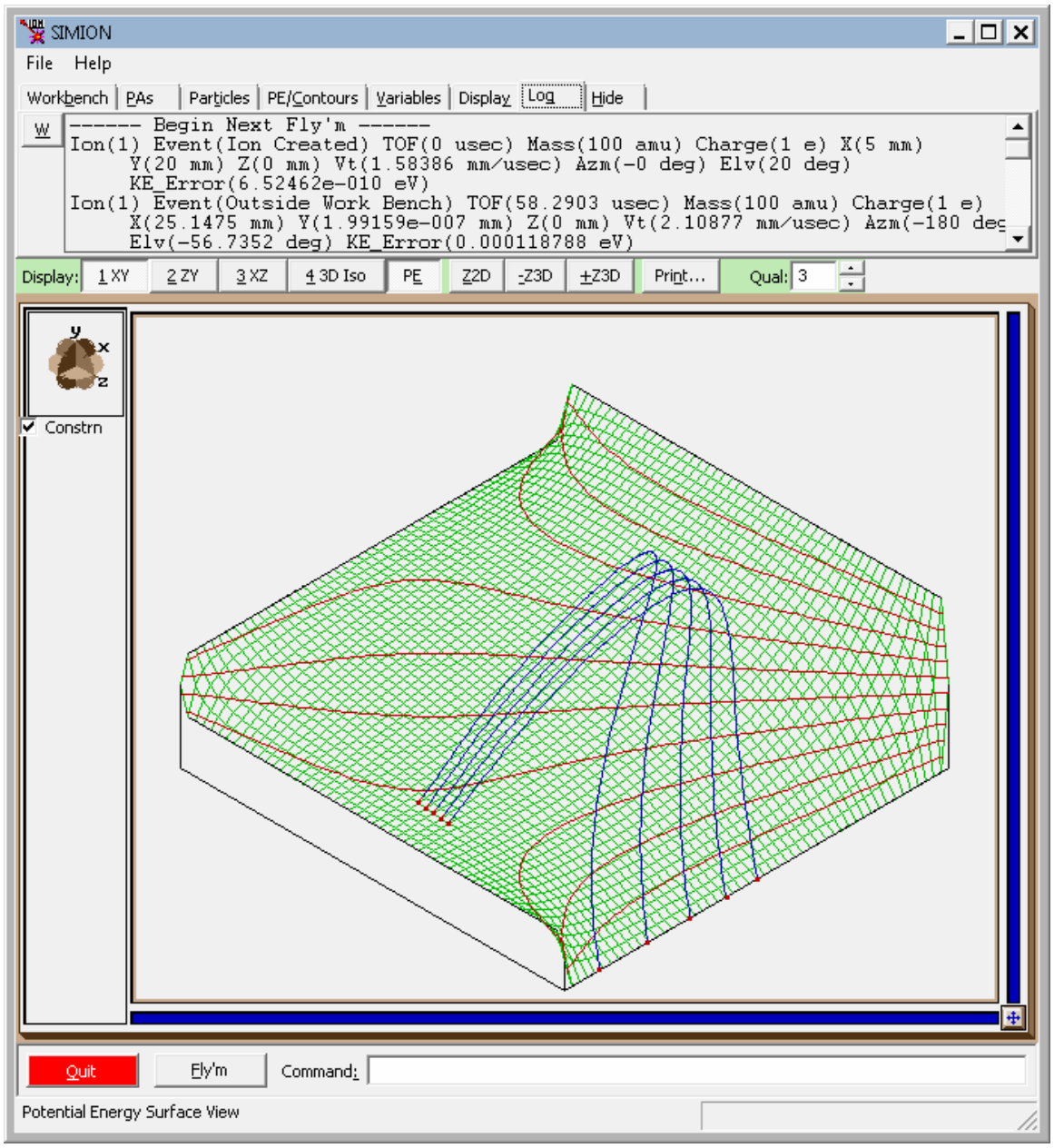


Figure 9: Recorded data shown in top "Log" window.

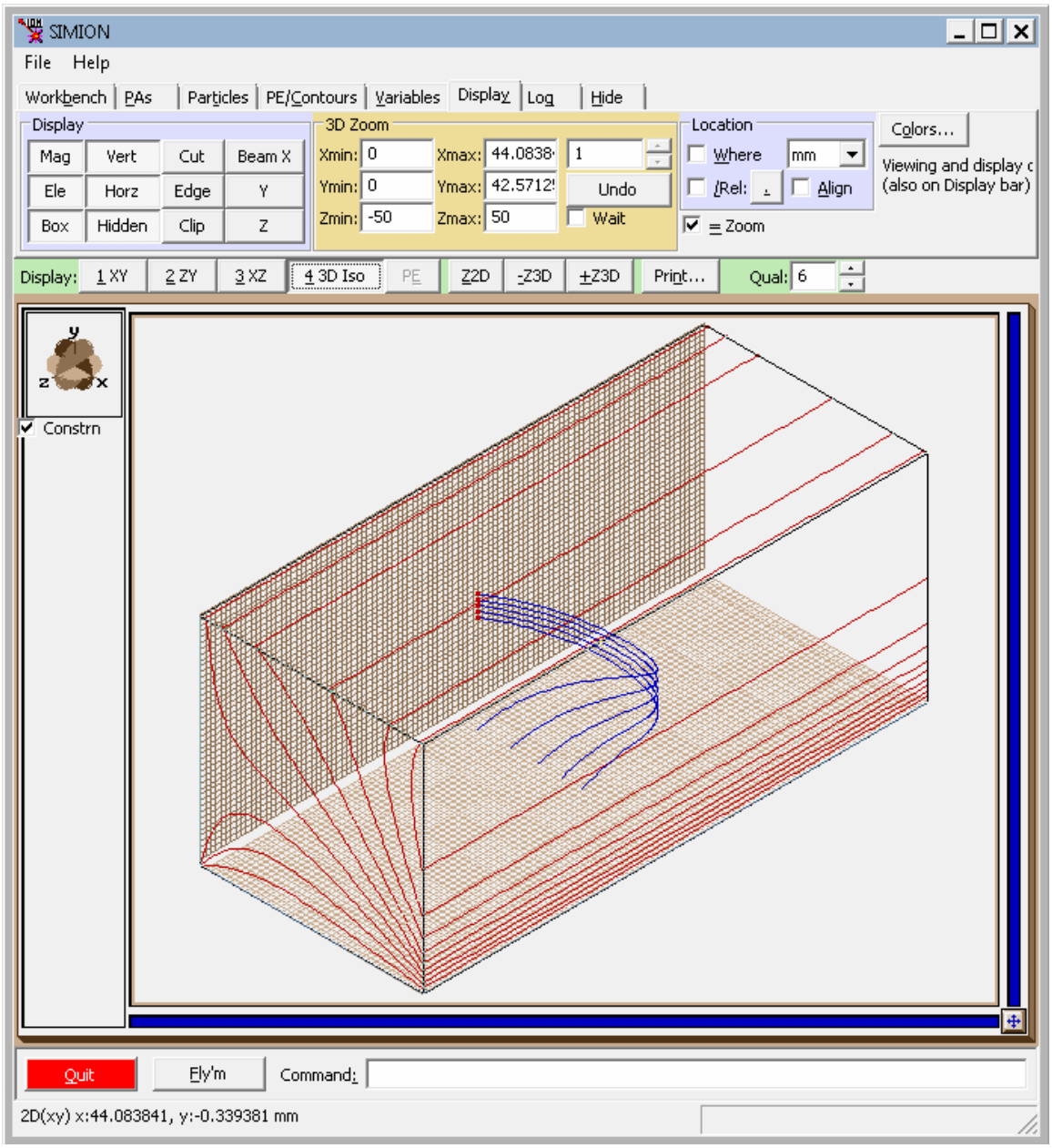


Figure 10: SIMION works in 3D as well and can solve full 3D problems. Here the above example is shown with the top and right plates removed (so that you can see inside).

4.3 Magnetic Fields

SIMION has some ability to solve for magnetic fields. It has some limitations in solving (e.g. it solves for scalar magnetic potential via the Laplace equation), but an arbitrary can be applied to particles.

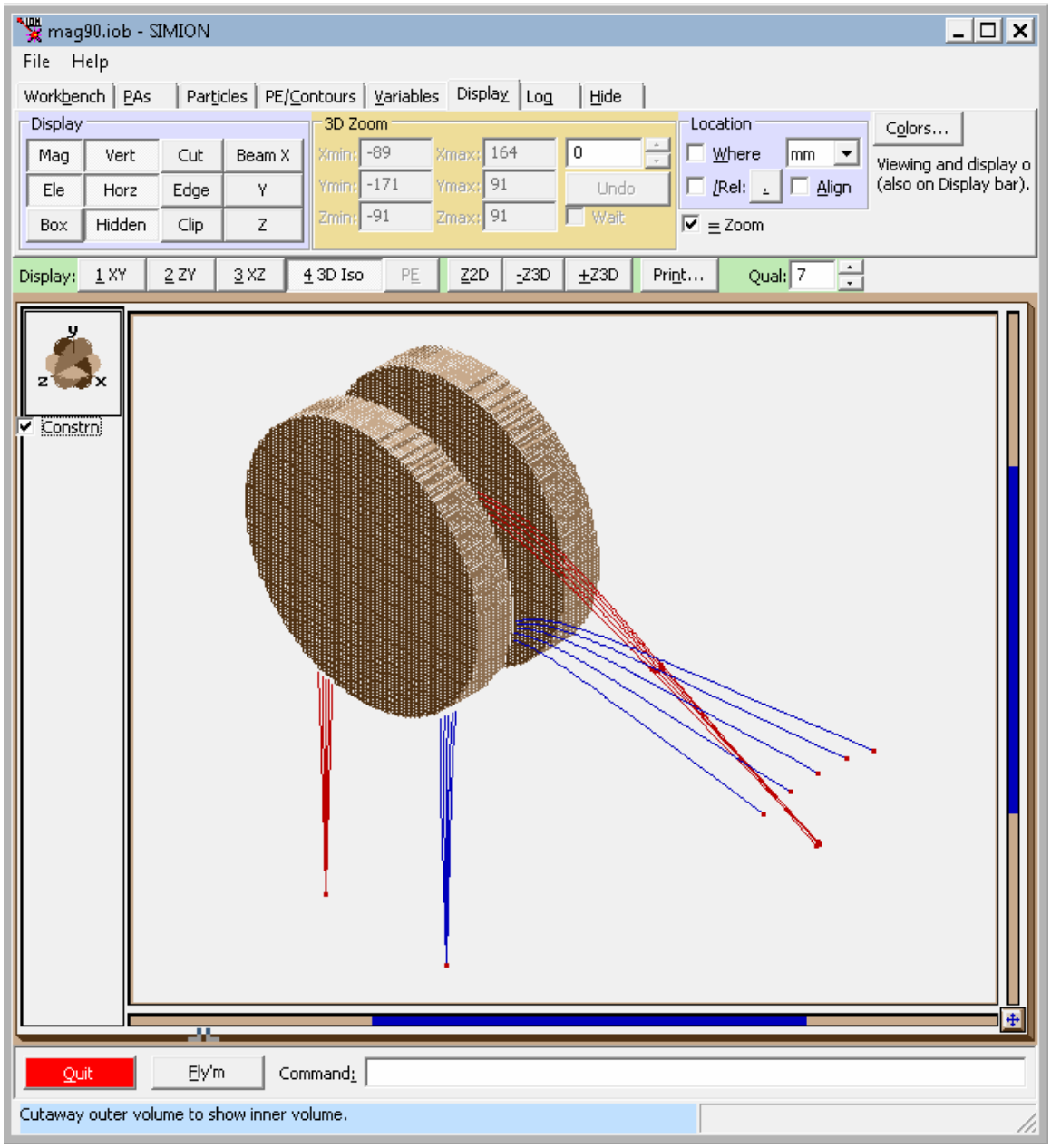


Figure 11: Ions bent by magnet.

4.4 Space Charge

Normally, fields and trajectories can be solved independently in SIMION. However, in high current beams, the current in the beam itself can alter the surrounding electric field, which can in turn alter the trajectories, so it becomes incorrect to solve these independently. This is called a "space charge" problem and is more difficult to solve. SIMION has some limited capabilities to handle this though.

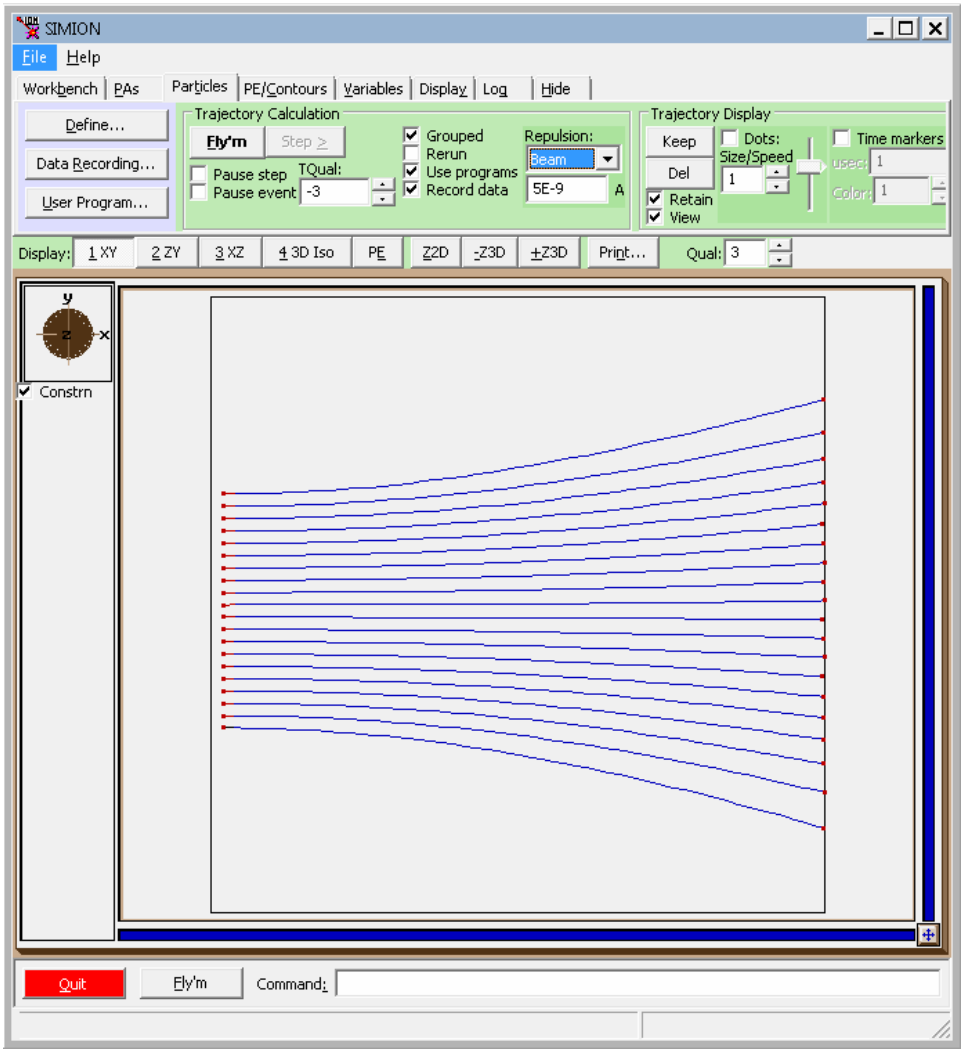


Figure 12: Ions mutually repelled by Coulombic repulsion.

4.5 Lua Programs

Many additional capabilities are possible via SIMION's user programming feature (Lua scripting language).

- Oscillate electrode voltages (e.g. RF quadropole rods)
- Define your own electric or magnetic fields, or adjust the calculated field.

- Modify ion trajectories—randomize ion positions and energies according to a distribution, add viscosity or randomized on-gas collision effects, and create secondary ions.
- Perform calculations, output data, and add logic to the simulation

```
-- test.lua
simion.workbench_program()
adjustable electrode_voltage = 500 -- voltage cycle magnitude
adjustable switch_time = 2.0       -- change time (microseconds)

-- electrode voltage control
function segment.fast_adjust()
    -- start undulating at time switch_time
    if ion_time_of_flight < switch_time then
        adj_elect01 = electrode_voltage
            * cos(1 + ion_time_of_flight/10)
    else
        adj_elect01 = 0
    end
end
```

Figure 13: Example Lua code. At a given time, this starts oscillating an electrode voltage according to the given sinusoidal equation.

We can see the effect of particles oscillations due to this oscillating electric field:

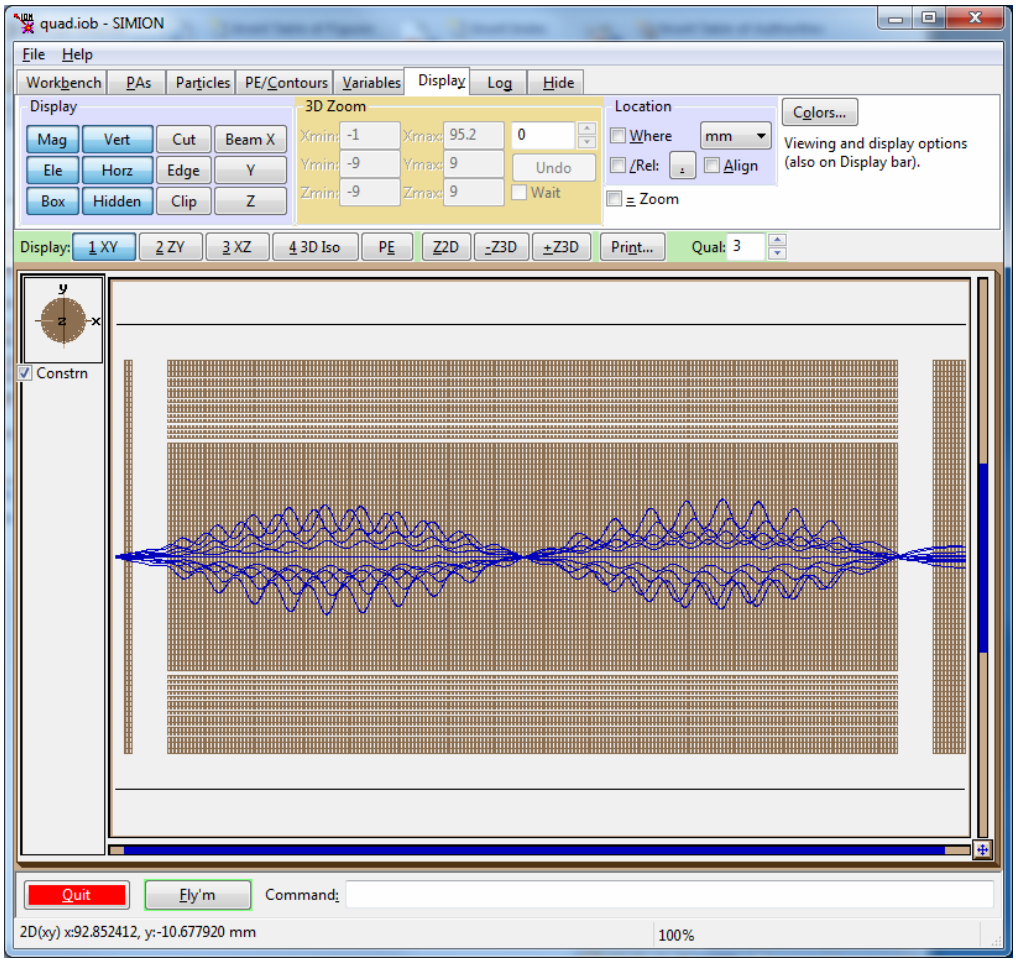


Figure 14 – Plot of SIMON trajectories

5 Theory

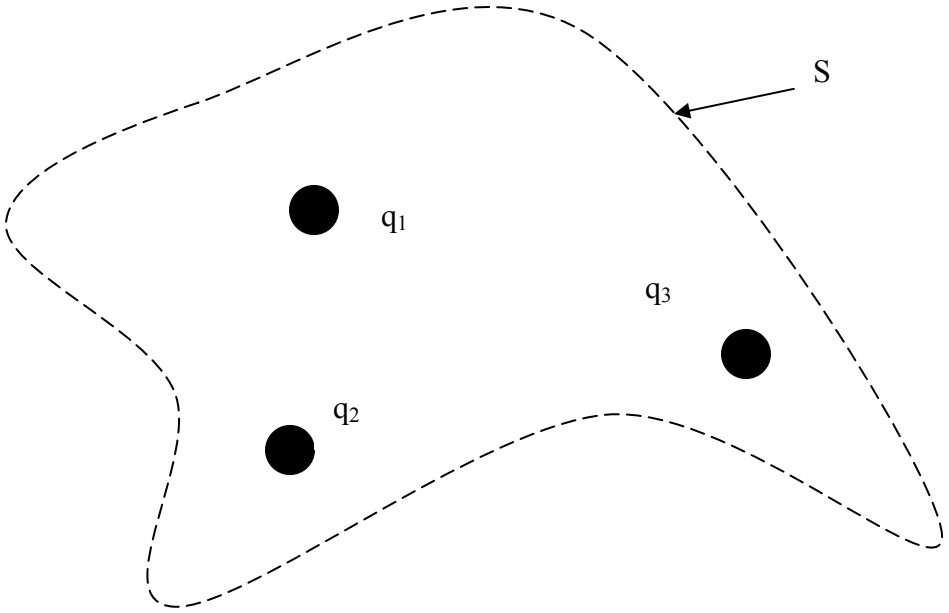
Major theorems in electrodynamics are described here. You need not use these because SIMION takes care of this. However, it's nice to know what these are to have a general idea of what SIMION is doing, and it can help you avoid pitfalls.

5.1 Gauss's Law

$$\int_S \mathbf{E} \cdot d\mathbf{a} = \frac{1}{\varepsilon_0} Q \qquad (1)$$

S is a closed surface
$\mathbf{E} = \mathbf{E}(x) = (E_x, E_y, E_z)$ is the electric field at point $\mathbf{x}=(x,y,z)$.
$\mathbf{a} = \mathbf{a}(\mathbf{x}) = (a_x, a_y, a_z)$ represents an infinitesimal portion of the surface.
a's has magnitude equal to the surface area and direction normal to the surface
ε_0 is the permittivity of free space (a constant)
Q is the total charge enclosed inside the volume bounded by the surface.

This equation relates the electric field on a surface to the charge within the volume bounded by the surface. Implication: the electric field on the surface is independent of how the charge is distributed within the volume!



5.2 Gauss's Law in differential form

$$\nabla \cdot \mathbf{E} = \frac{1}{\varepsilon_0} p \qquad (2)$$

$\mathbf{E} = \mathbf{E}(\mathbf{x}) = (E_x, E_y, E_z)$ is the electric field at point $\mathbf{x}=(x,y,z)$.
ε_0 is the permittivity of free space (a constant)
$\rho = \rho(\mathbf{x})$ is the charge density (charge per volume) at point $\mathbf{x}=(x,y,z)$

This is a simple reformulation of Gauss's Law by applying the divergence theorem (an important theorem in calculus). Essentially, let the size of S approach zero

5.3 Relationship between Electric Field and Potential

$$\mathbf{E} = -\nabla V \qquad (3)$$

$\mathbf{E} = \mathbf{E}(\mathbf{x}) = (E_x, E_y, E_z)$ is the electric field at point $\mathbf{x}=(x,y,z)$.
$V = V(\mathbf{x})$ is the potential energy at point $\mathbf{x}=(x,y,z)$.

The electric field can be computed easily from the potential energy map.

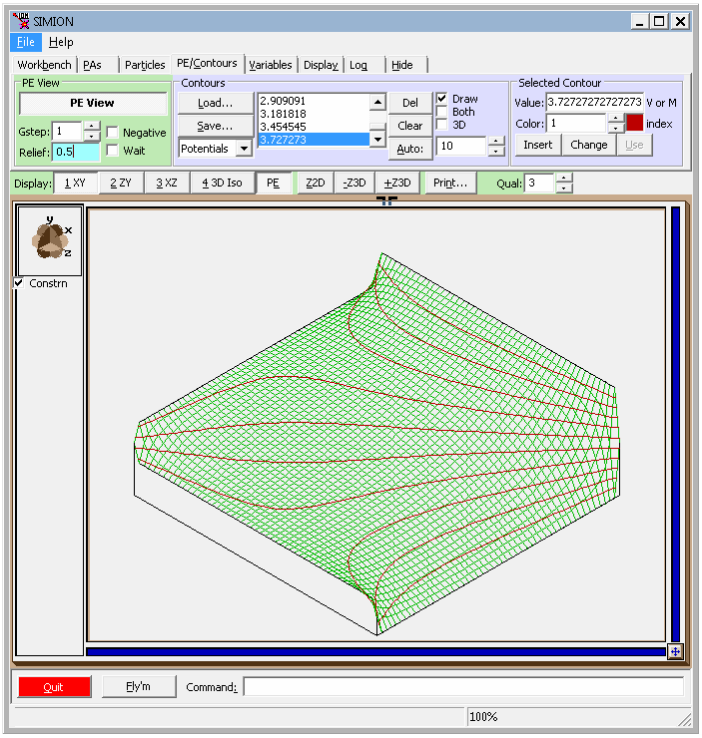


Figure 15: Potential energy map. Equipotential contours have been superimposed.

Conversely, though we usually don't have a need to do so in using SIMION, the potential energy map can be computed from the electric field (though there is an undefined "constant of integration" since $\mathbf{E} = -\nabla(V + C)$ for any constant C).

5.4 Poisson's Equation

Putting the above equations together, we get the very important *Poission equation*:

$$\nabla^2 V = -\frac{p}{\varepsilon_0} \qquad (4)$$

$V = V(\mathbf{x})$ is the potential energy at point $\mathbf{x}=(x,y,z)$.
$\rho = p(\mathbf{x})$ is the charge density (charge per volume) at point $\mathbf{x}=(x,y,z)$
ε_0 is the permittivity of free space (a constant)

This is a simple derivation from Gauss's Law in differential form:

$$\frac{1}{\varepsilon_0} p = \nabla \cdot \mathbf{E} = \nabla \cdot (-\nabla V) = -\nabla^2 V \quad (5)$$

When ρ > 0, we say that the system has "space-charge." Space-charge occurs when ions are too close to each other (i.e. the charges in the particle beam are significant enough to affect the trajectories of the particles in the beams). Though SIMION 8.0 (unlike SIMION 8.1) does not solve the Poisson equation, it does support some methods for roughly estimating space-charge effects.

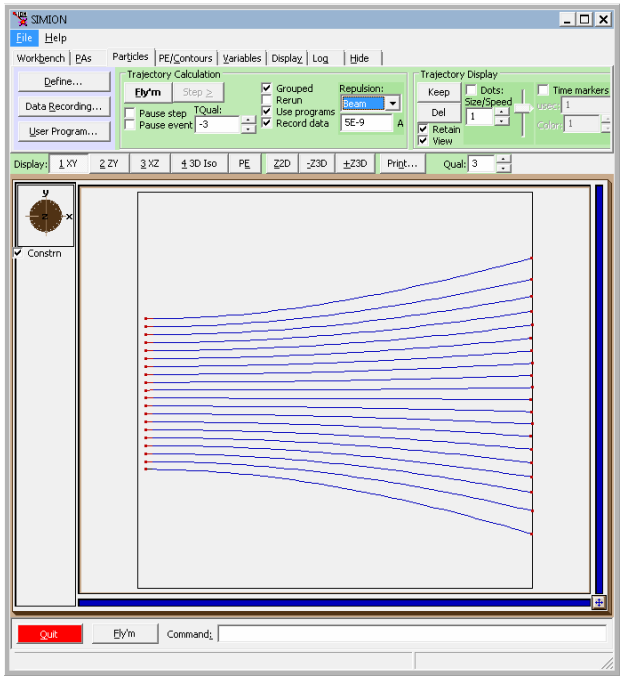


Figure 16: ions initially traveling parallel to each other but repel each other.

5.5 Laplace's Equation

$$\nabla^2 V = 0 \quad (6)$$

This is a direct result of Poisson equation when ρ=0. It is often the case that ρ = 0 (or approximately so) in space. We say that "space-charge" is negligible.

The Laplace equation determines which electric potentials (or fields) are permissible. Along with a boundary condition (your electrode voltages),

SIMION solves the Laplace equation in order to compute space potentials (and hence electric fields as well).

SIMION solves the Laplace equation using the numerical method called the "Finite Difference Method (FDM)," which is rather straightforward. It is essentially an averaging process resulting from this theoretical result:

$$V = \frac{1}{4\pi r^2} \oint_S V da \qquad (7)$$

In words, the voltage at a point in space equals the average of the voltages of the points around it. This also implies that the potential energy map from a static electric field can contain no local minimum or maximum (which we'll discuss later).

5.6 Numerically Solving the Laplace equation.

There are many methods to solve the Laplace equation numerically.

Could we calculate these potentials using Excel? It's easier than you think! We apply the relationship $V = \frac{1}{4\pi r^2} \oint_S V da$. To calculate this numerically, we instead apply its approximation that each point is the average of the four neighboring points.

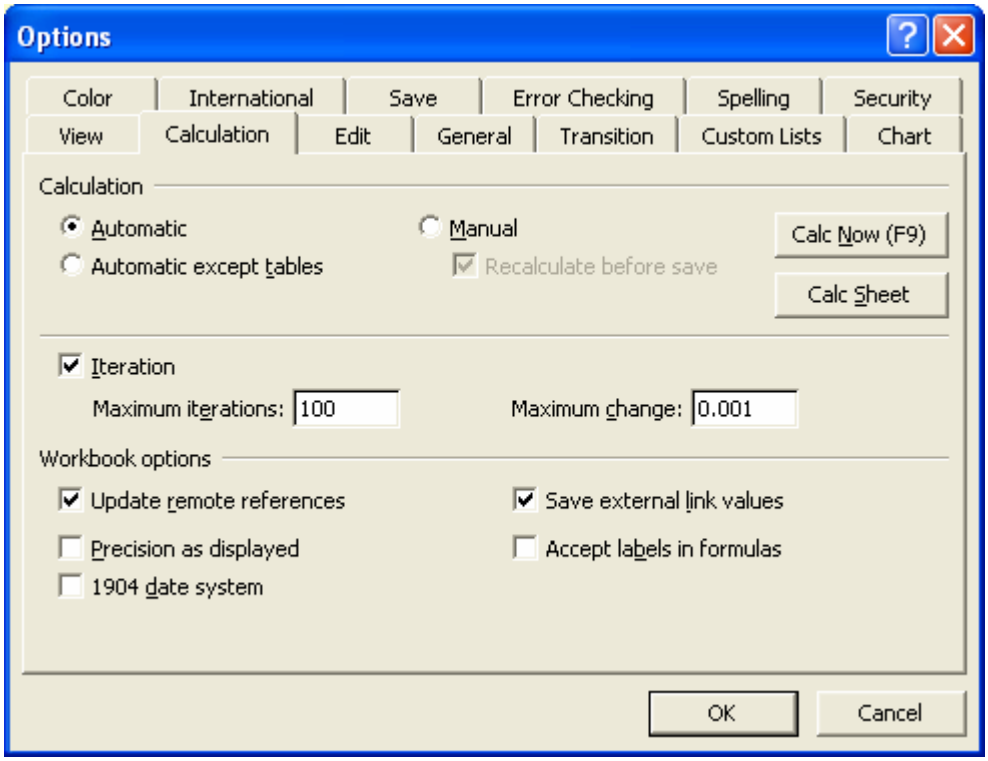


Figure 17: Excel iterative calculation options.

First, we must enable "Iteration" since circular references will be used.

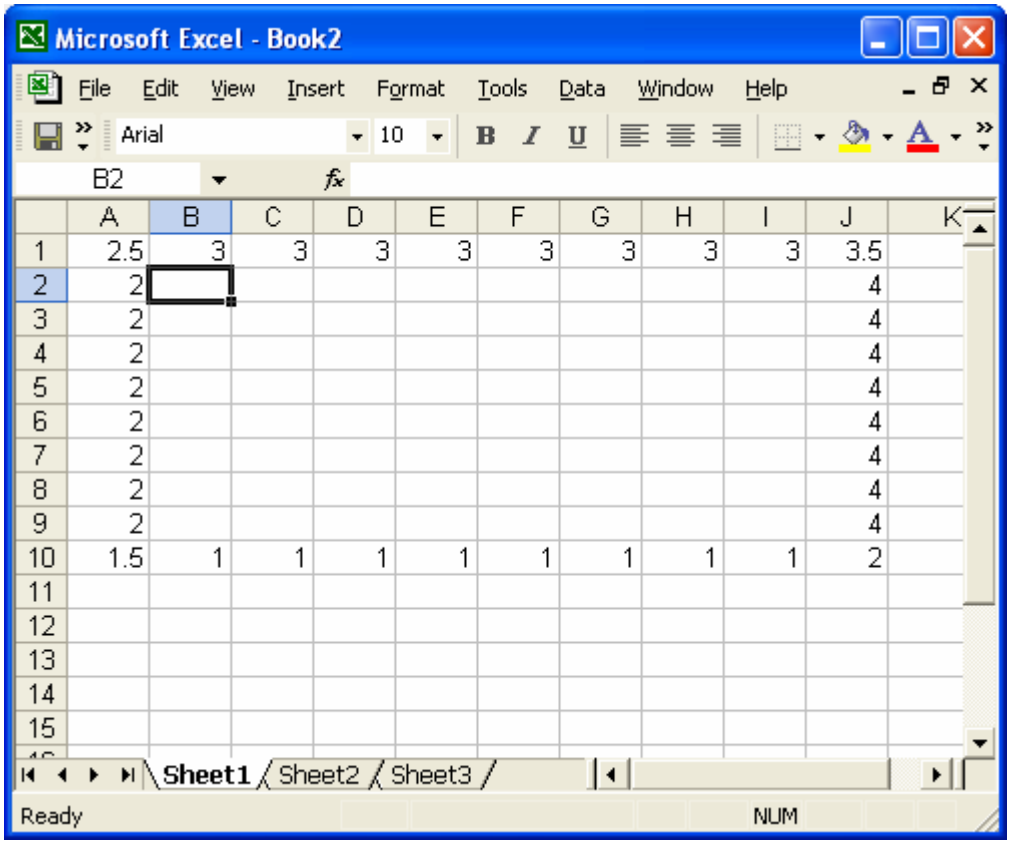


Figure 18: Boundary potentials defined in Excel.

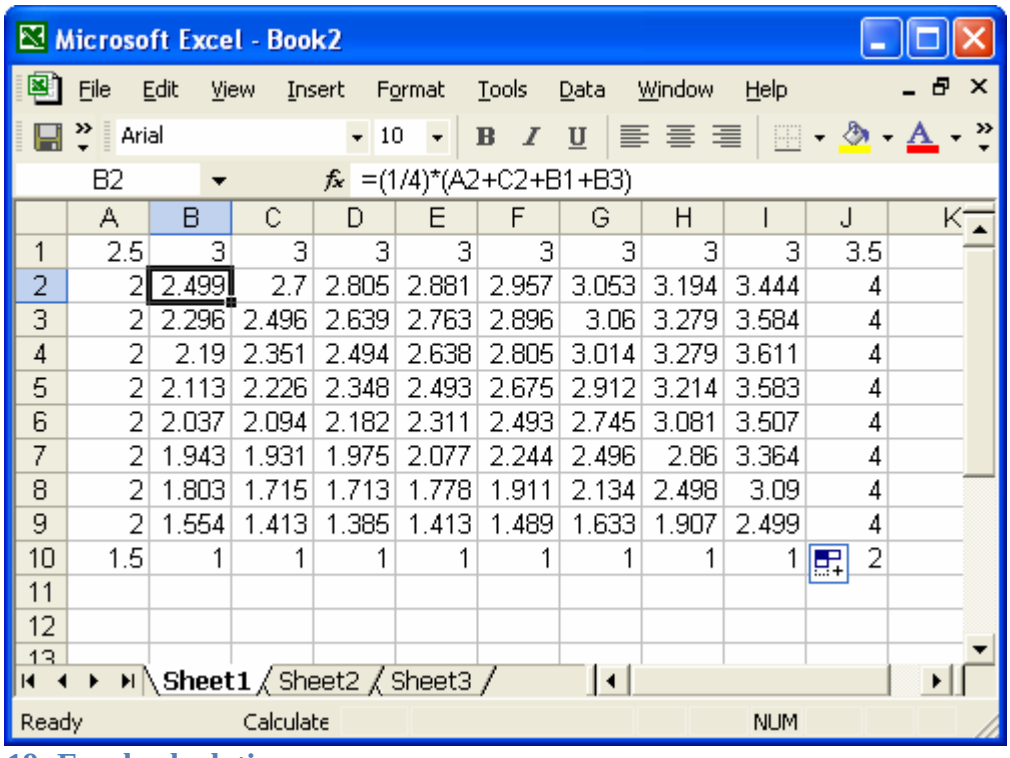


Figure 19: Excel calculation.

Set B2 = (1/4)*(A2+C2+B1+B3) (i.e. average of four neighbors). Then fill down and right. The converged result displayed.

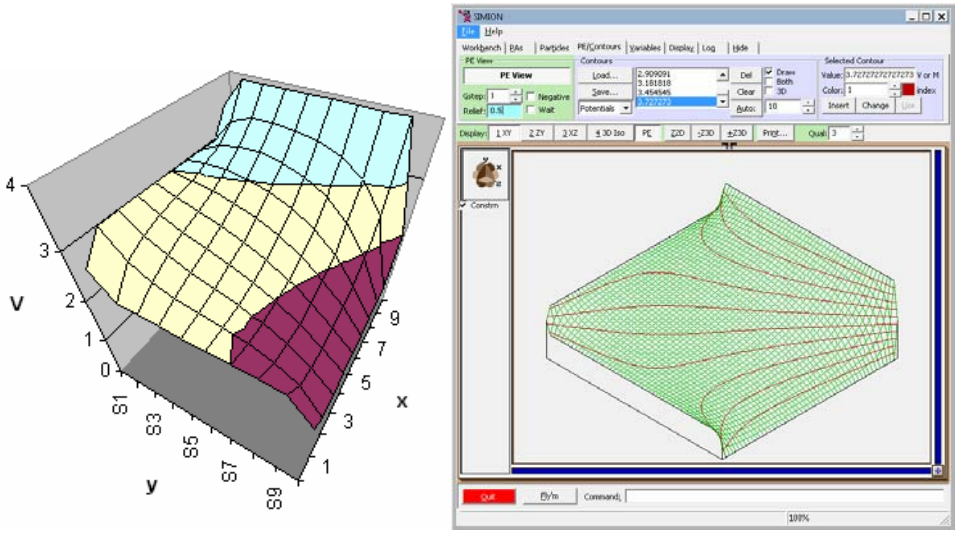


Figure 20: (left) Potential energy plot from Excel, (right) potential energy plot from SIMION

That's basically the calculation that SIMION does…except more efficiently and more generally.

5.7 Earnshaw's Theorem

"A charged particle cannot be held in a stable equilibrium by electrostatic force alone." (Griffiths, 1999)

Why does a quadruple or ion-trap oscillate pole voltages?
Why does an ICR cell contain a magnetic field?
Because **a charged particle cannot be held in a stable equilibrium by electrostatic force alone**.

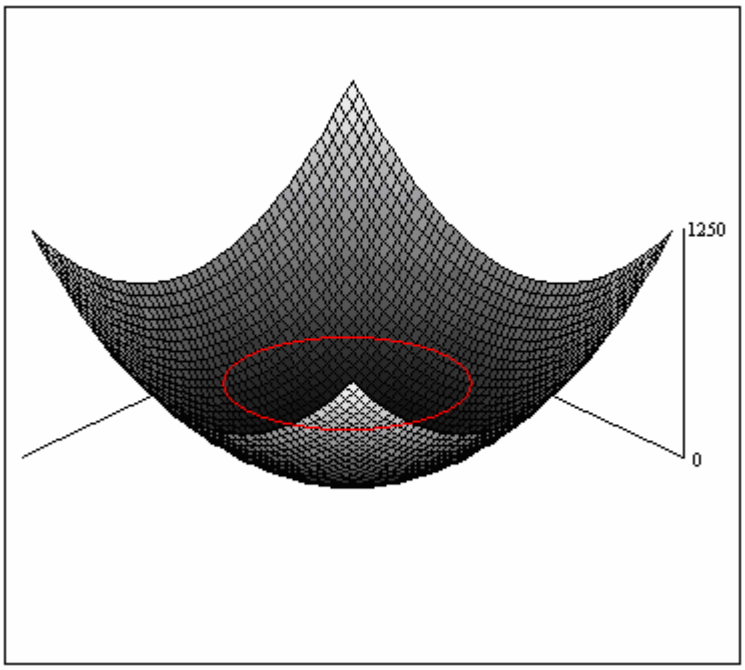


Figure 21: Potential energy well (local minimum)

This potential energy well (with local minimum to trap a particle whose path is shown in red) is **not possible** in free-space (of zero space-charge) with **any** configuration of electrodes of static potential. If it did exist, it would violate the result $V = \frac{1}{4\pi r^2}\oint_S V da$ mentioned previously.

However, you can trap a particle with a **static magnetic field** or **oscillating electric fields**.

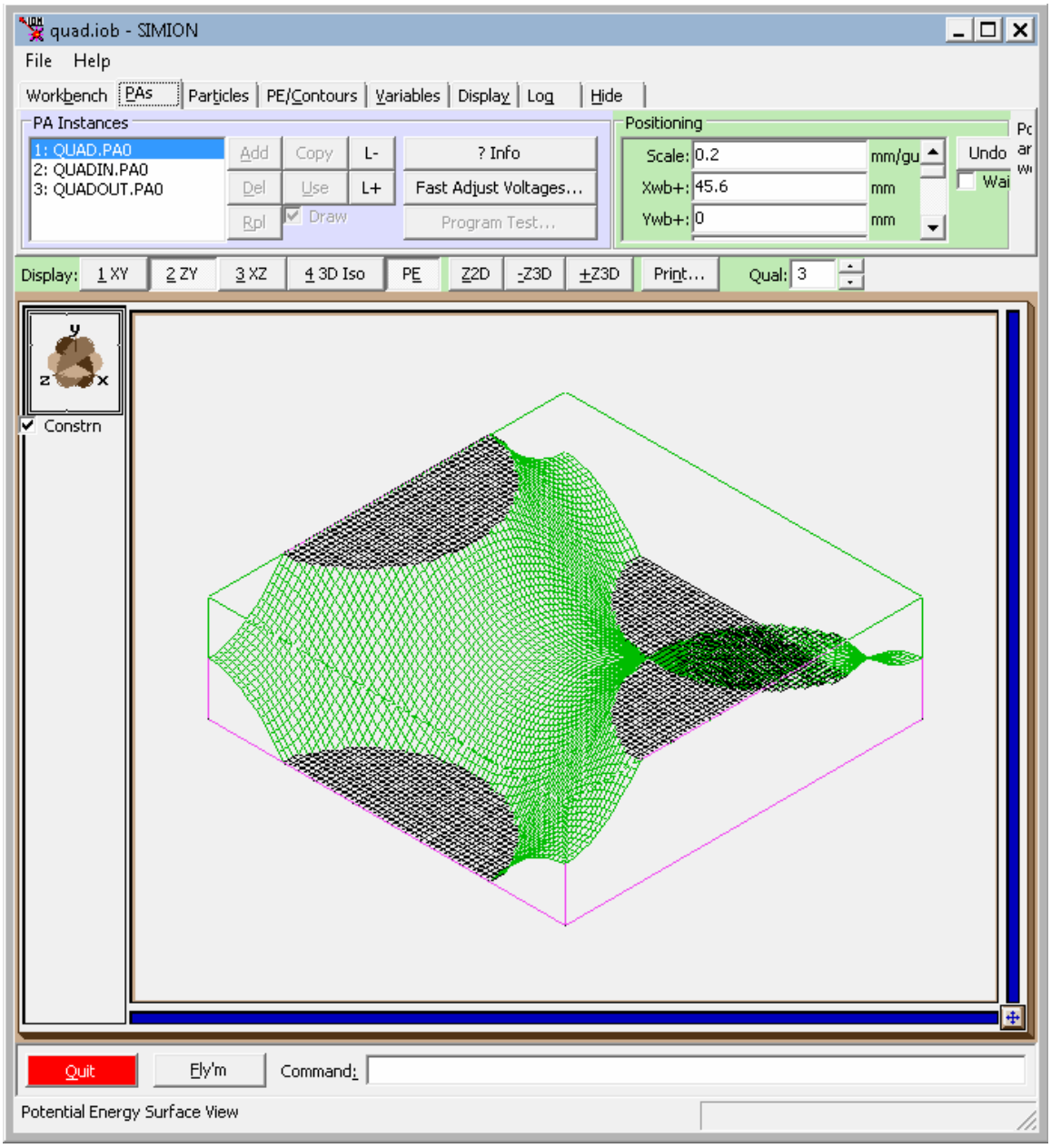


Figure 22: "Saddle-like" potential energy surface of quadrupole.

For example, here's the potential energy map on a cross-section or a quadrupole with poles at constant potential. Note the "saddle"—an ion is not stable on this unless we oscillate the electrodes.

The **ICR cell** traps with a magnetic field:

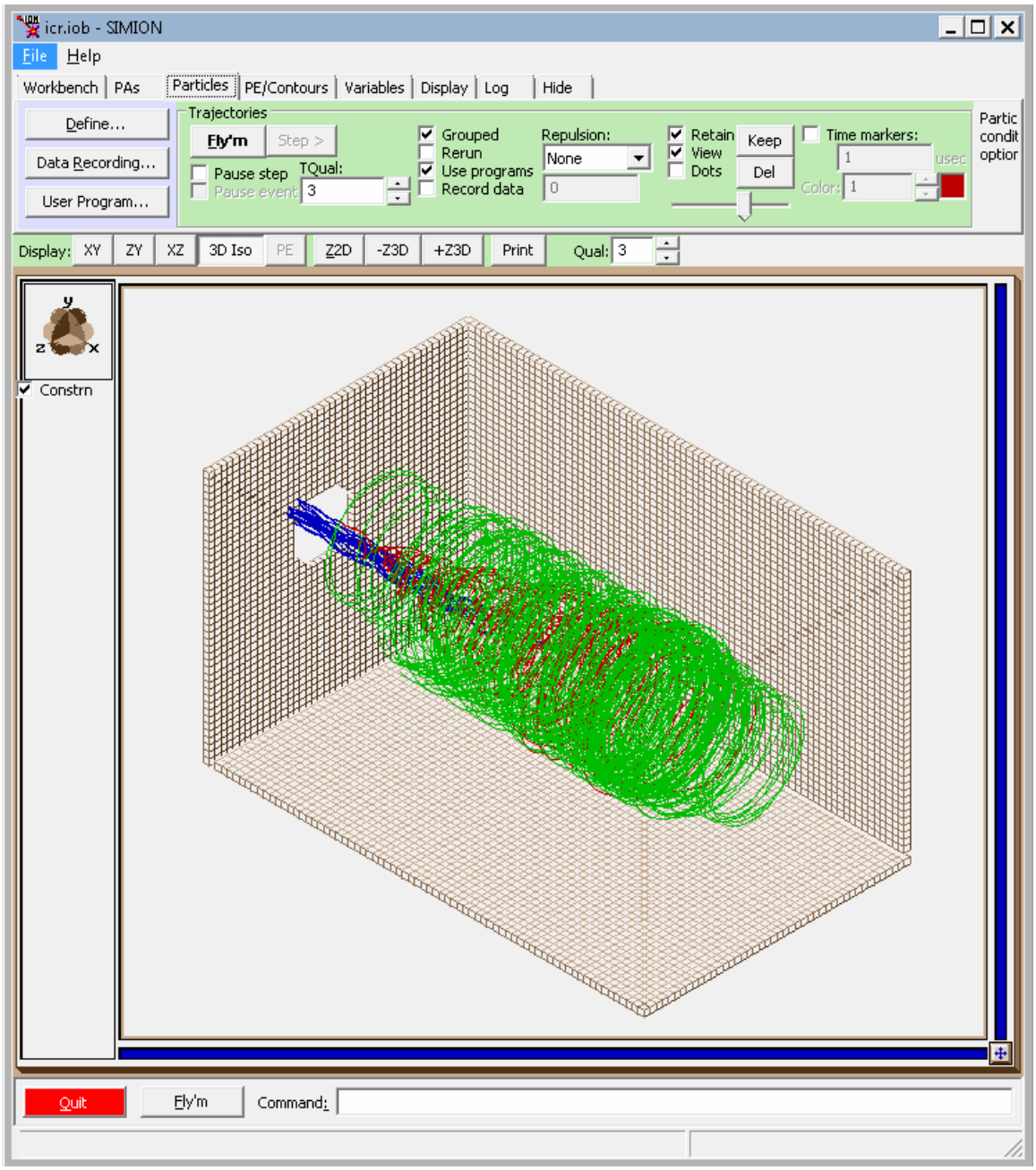


Figure 23: ions moving through an ICR cell.

5.8 First Uniqueness Theorem:

The solution to Laplace's equation in some volume V is uniquely determined if V is specified on the boundary surface S. (Griffiths, 1999)

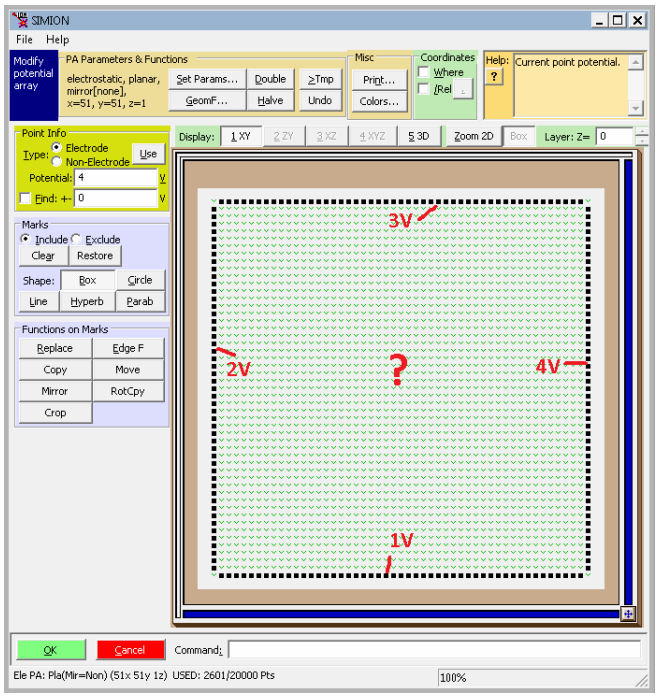


Figure 24: empty space completely surrounded by electrodes surfaces with known potential.

That is what we did—all points we sought to calculate were surrounded by a boundary electrodes.

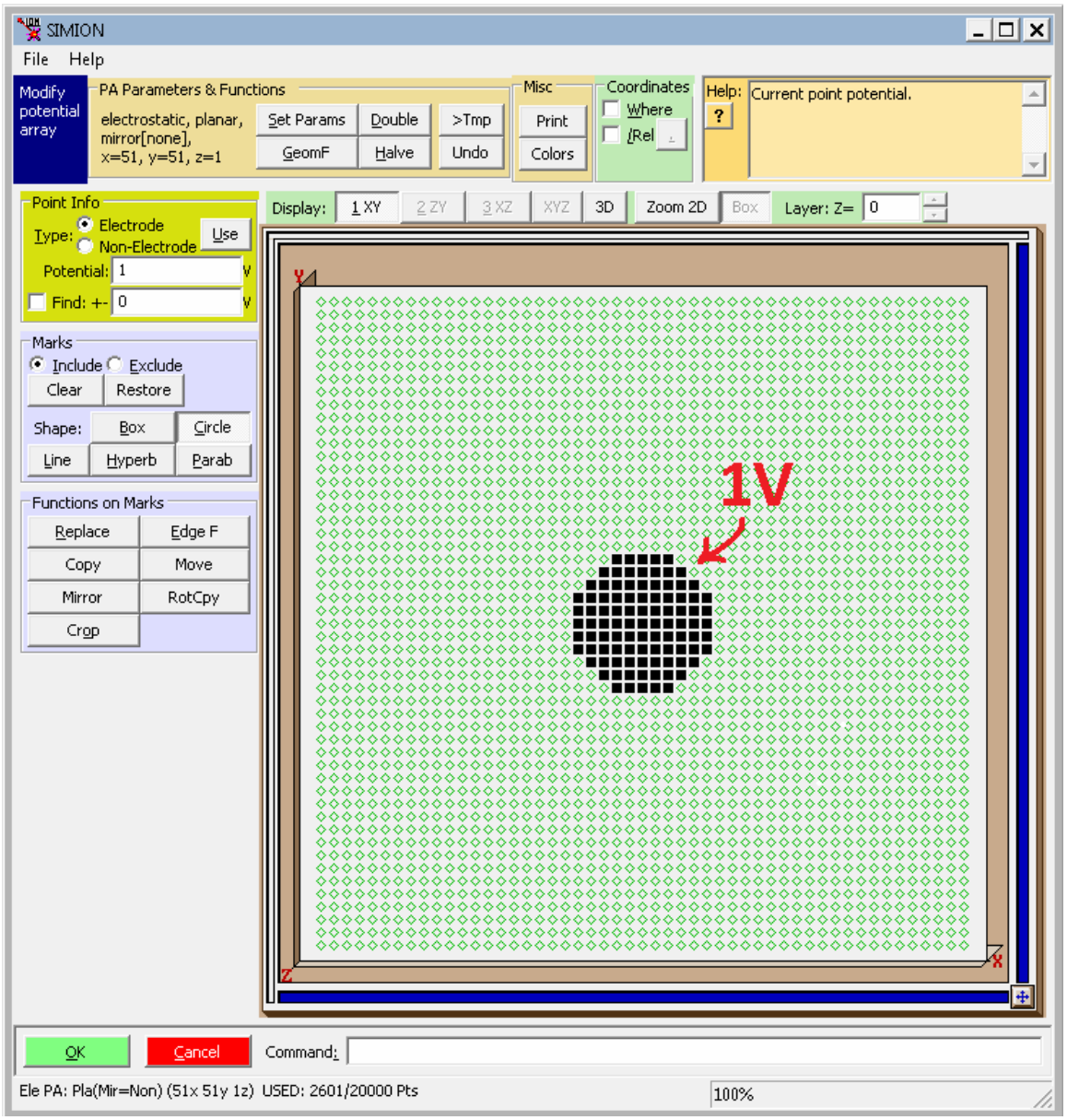


Figure 25: Circle (or sphere) of charge not bounded by any other electrodes.

Can we do the above?

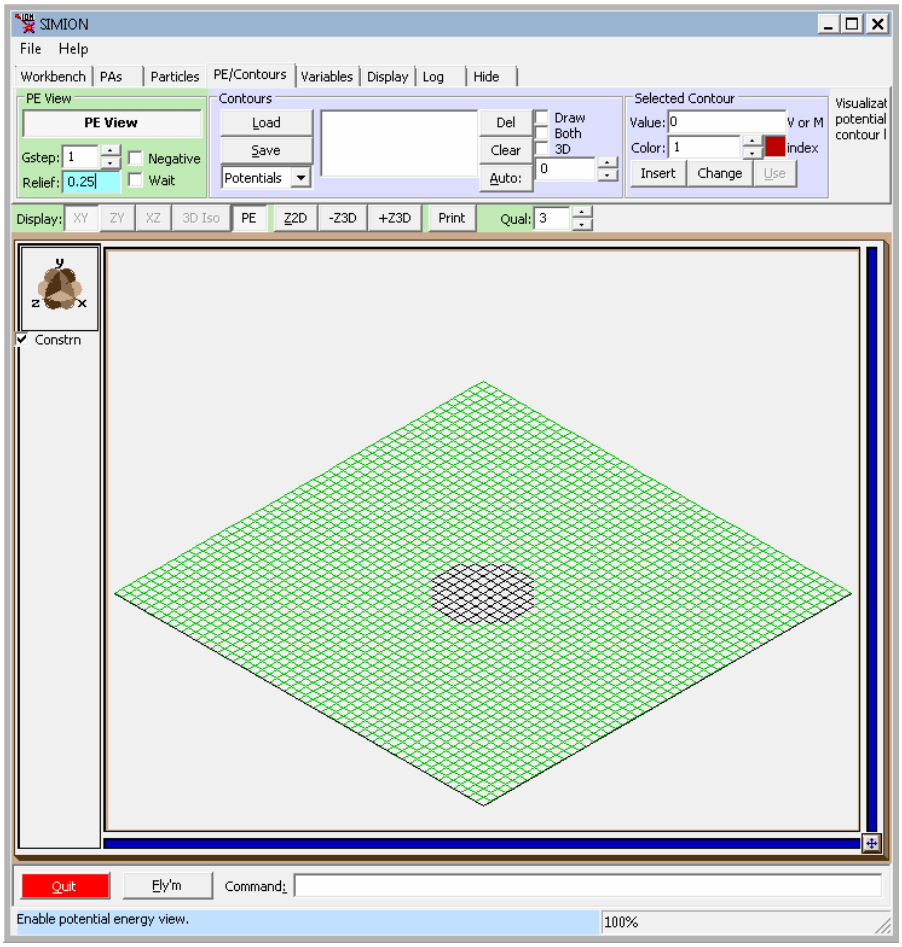


Figure 26: Potential energy surface calculated by SIMION

SIMION calculated 1V over the entire volume. Huh? What's going on? Doesn't Coulomb's Law imply that the potential at a distance r from a charged sphere is inversely proportional to r? That is, we'd expect the potential to decay as one gets further away from the sphere, but the potential is almost **constant**!

The above system does not have a boundary. Laplace will not be solved meaningfully.

Here's a more correct simulation:

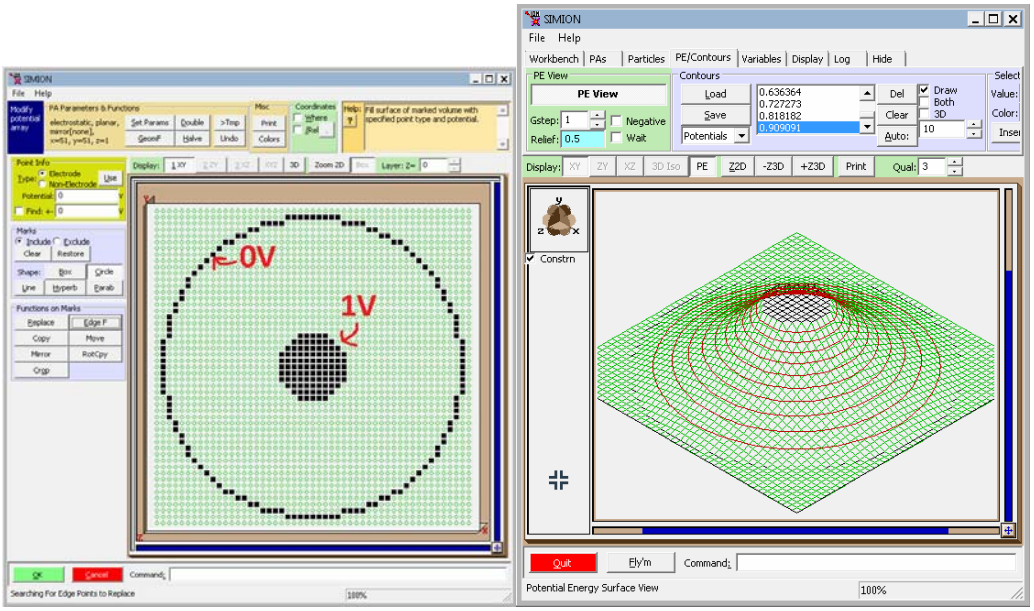


Figure 27: Potential energy surface calculated by SIMION after defining grounded boundary surface.

Actually, SIMION is a tad more lenient than this…

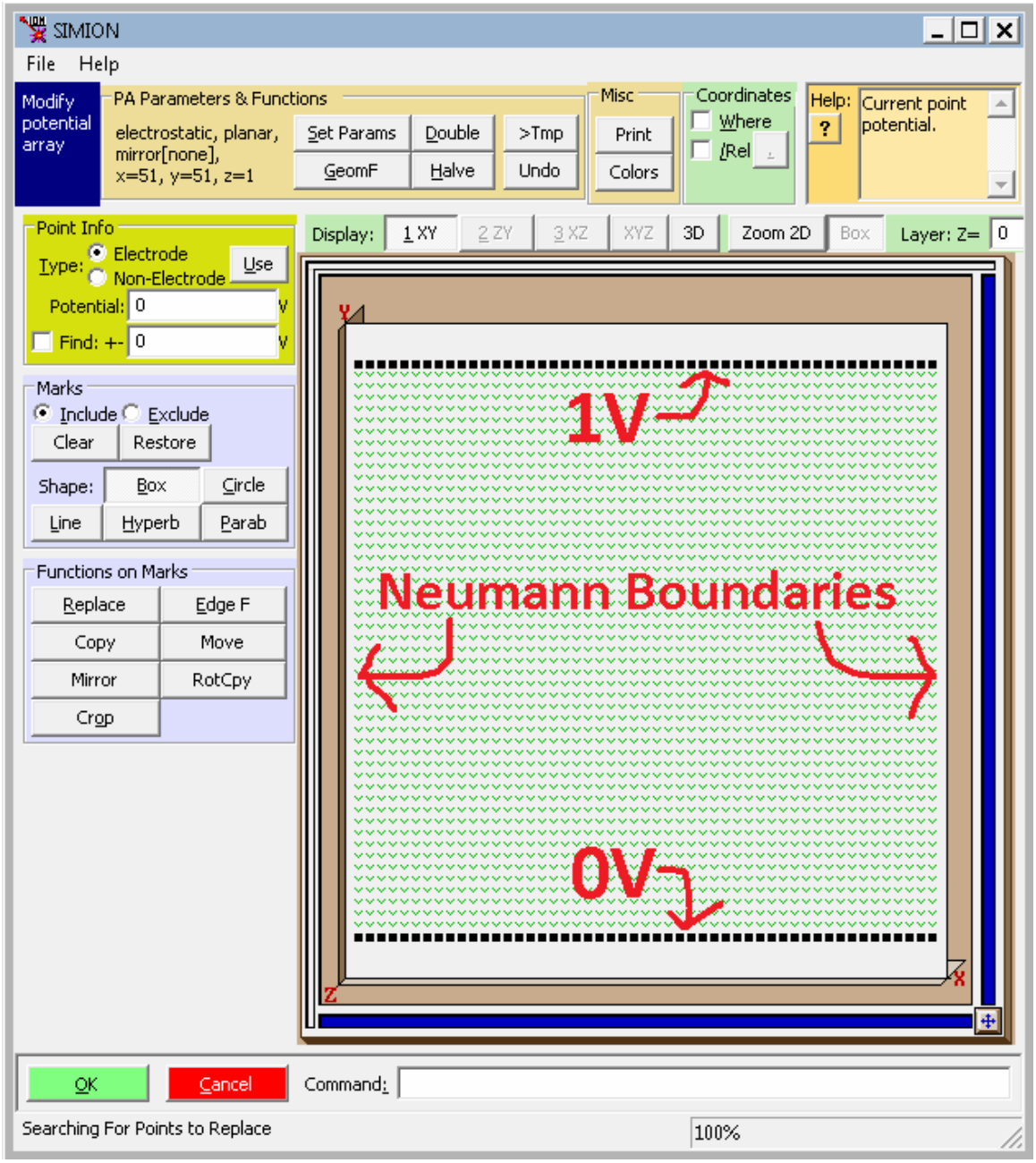


Figure 28: Partially bounded system.

Edges that are open are treated essentially as Neumann boundaries conditions (zero normal derivative perpendicular to the surface). Sometimes this is what we want; sometimes it is not.

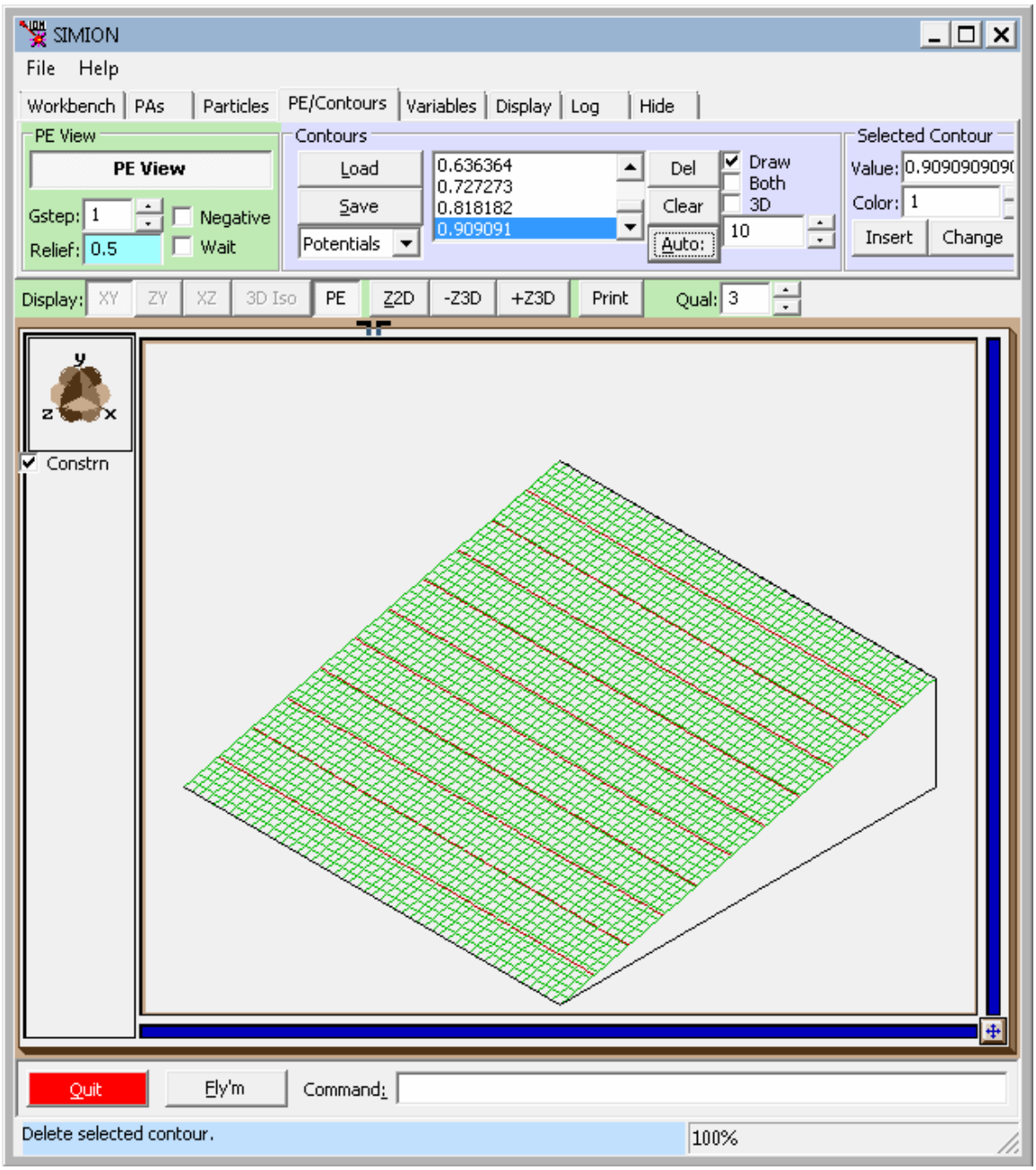


Figure 29: Potential energy map.

Note that this is roughly correct (ignoring fringing effects, i.e. curving near the edges).

We've seen another partially bounded system before:

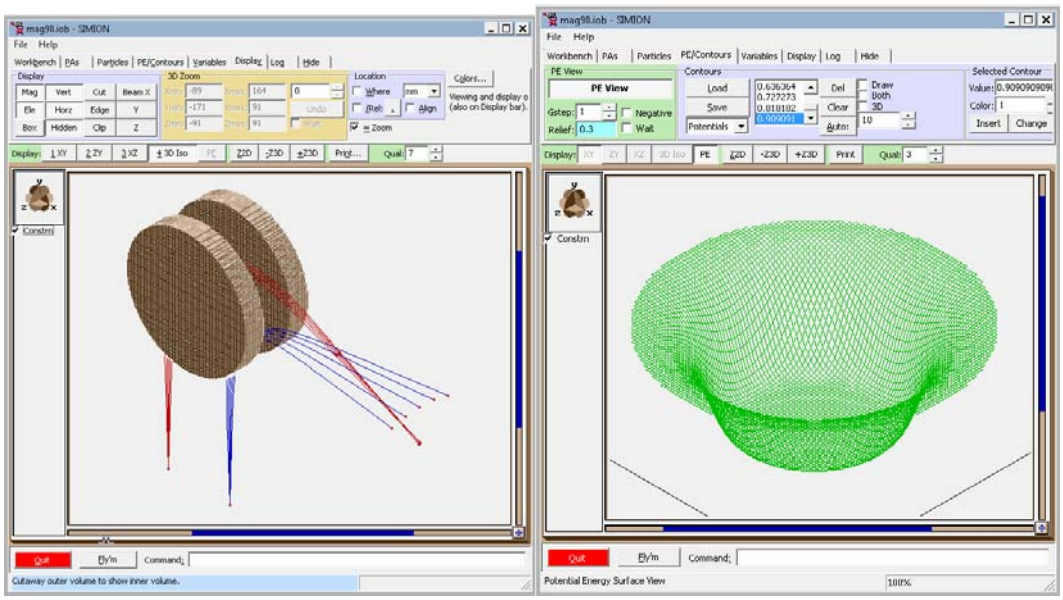


Figure 30: To concentric magnets open in the center. (right) potential energy map along center (note: there are even fringing effects).

5.9 The Lorentz Force Law

The movement of particles in electric and/or magnetic fields is described by the

Lorentz force law:

$$\mathbf{F} = \mathbf{q}[\mathbf{E} + (\mathbf{v} \times \mathbf{B})] \quad (8)$$

And Newton's second law of motion:

$$\mathbf{F} = m\,\mathbf{a} \quad (9)$$

6 Anatomy of a SIMION Project

A SIMION simulation consists of a number of different pieces. These are described here.

6.1 Workbench Concept

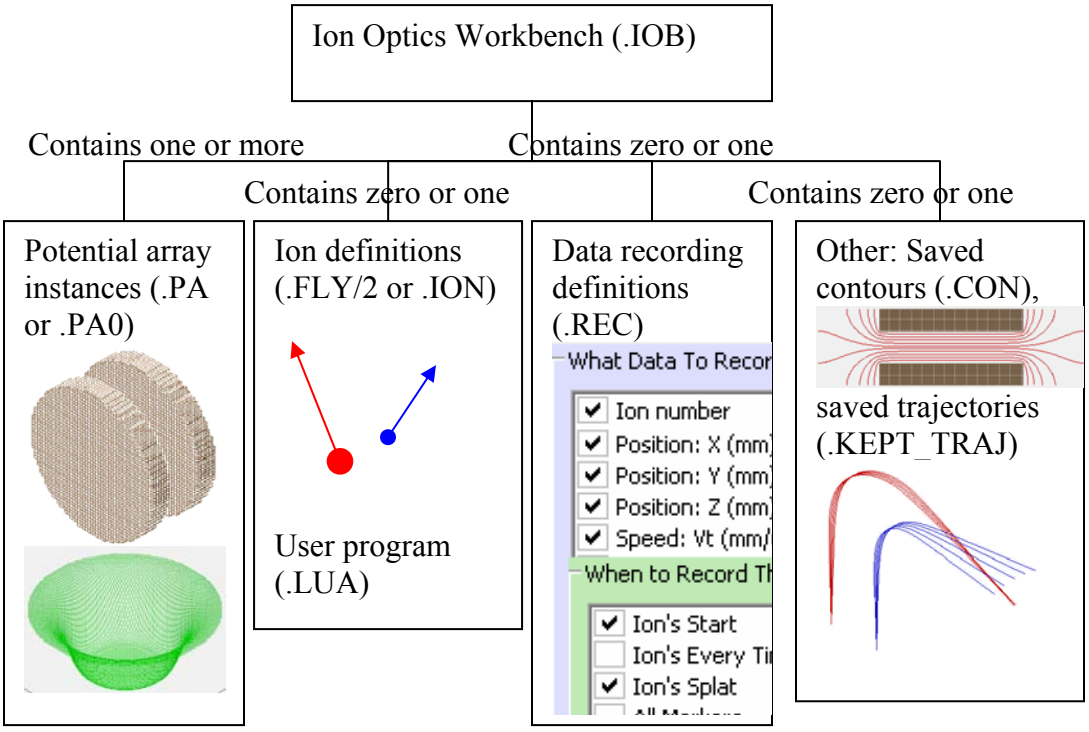


Figure 31: Components of a SIMION workbench

6.2 Ion Optics Workbench (.IOB) File

The .IOB file is the main SIMION project file that ties all other files together, and it's what you "run." The .IOB file is accessed from the "View" screen. It

- Contains other files
- Defines the extent of the space enclosing the system (length, width, height)
- Specifies the scaling, rotation, and translation of the .PA instances.
- Specifies the voltages on any fast adjustable .PA0 instances.
- Specifics any other project-wide settings.

6.3 Particle Definitions: .FLY2 and .ION files

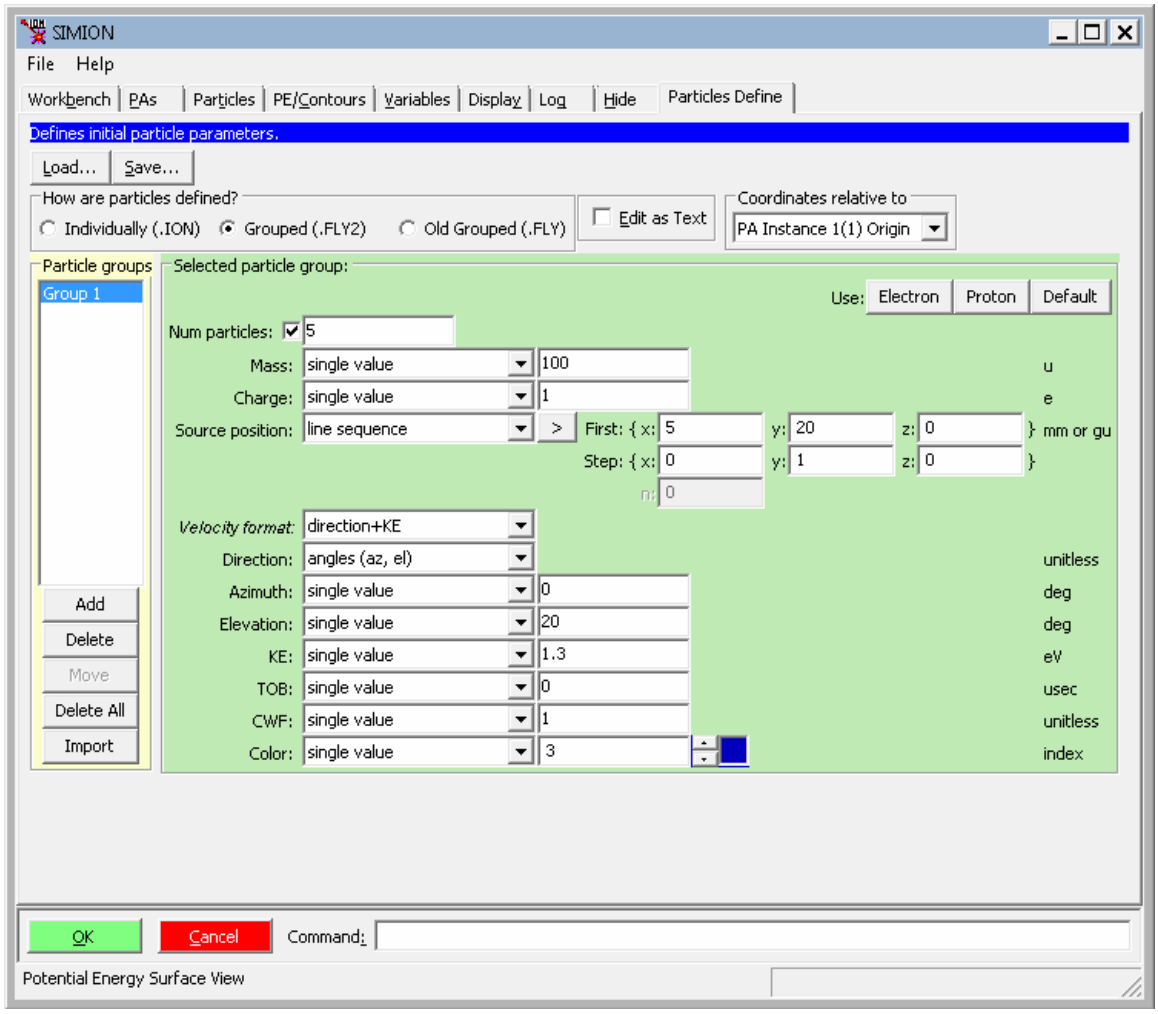


Figure 32: SIMION particle definitions in "FLY2" format.

Specifies:

- One or more "groups" of ion definitions, where each group specifies
  - The number of particles in the group
  - The mass, charge, position, angle, and KE of the first particle in the group.
  - Deltas for subsequent particles in the group

.FLY2 files can make it **easy** to generate a series of ions, where each ion differs from the next by a constant. However, .FLY2 files are not powerful enough to specify arbitrary ions (not related by a constant delta). For this, we need .ION files.

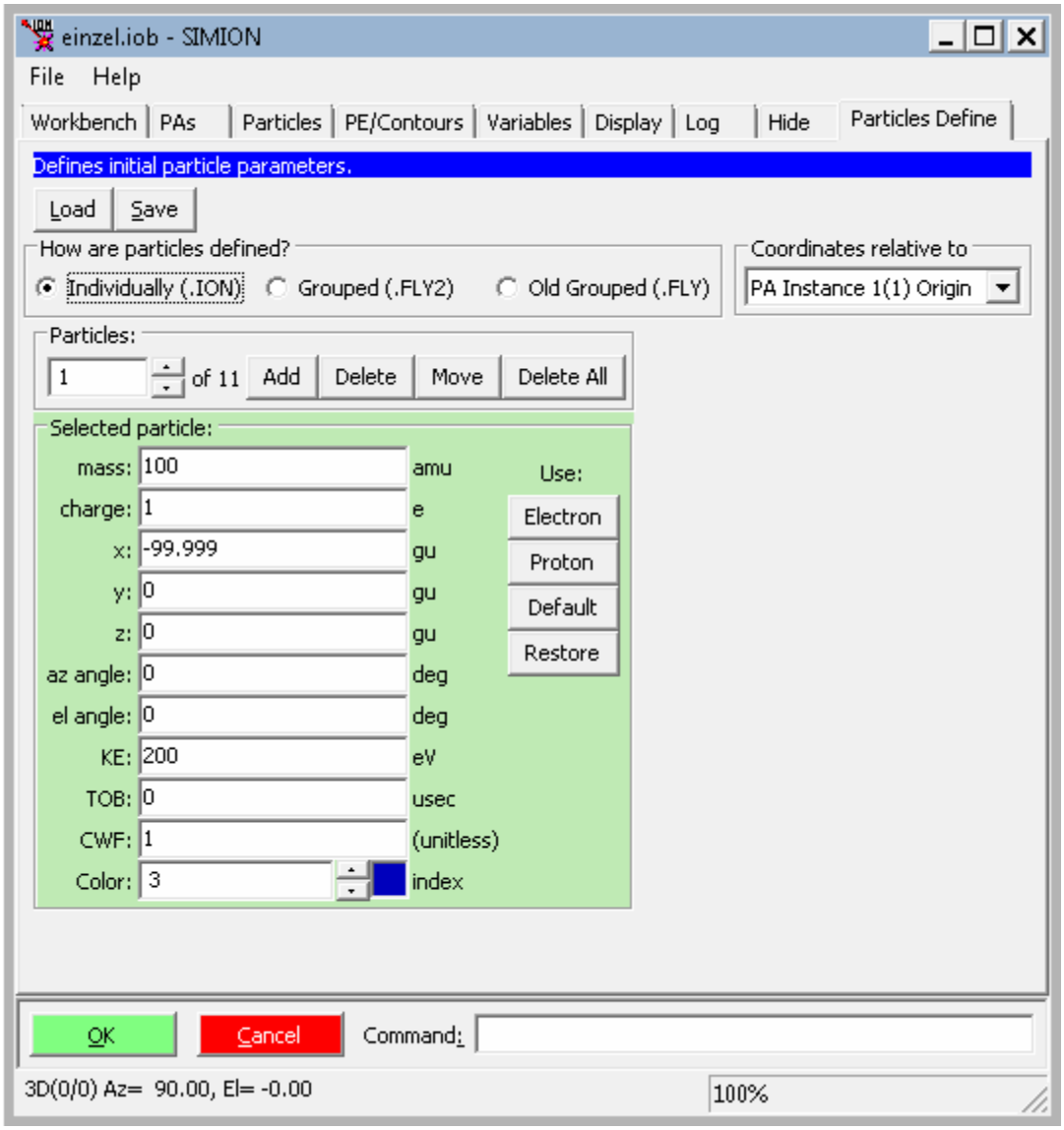


Figure 33: SIMION particle definitions in ".ION" format.

The .ION file defines one or more ions, each individually.

```
;1
 1.00000000e+000, 2.00000000e+000, 3.00000000e+000, 4.00000000e+000,
5.00000000e+000, 6.00000000e+000, 7.00000000e+000, 8.00000000e+000,
9.00000000e+000, 1.00000000e+000,0
 1.00000000e+000, 2.00000000e+000, 3.00000000e+000, 4.00000000e+000,
5.00000000e+000, 6.00000000e+000, 7.00000000e+000, 8.00000000e+000,
9.00000000e+001, 1.00000000e+000,0
 1.00000000e+000, 2.00000000e+000, 3.00000000e+000, 4.00000000e+000,
5.00000000e+000, 6.00000000e+000, 7.00000000e+000, 8.00000000e+000,
9.00000000e+002, 1.00000000e+000,0
```

Figure 34: TEST.ION - Contents of the .ION file (showing three ions, one per line).

Since it's just a delimited text file, you could even read/write the .ION file in Excel.

6.4 Data Recording (.REC) file

The data recording file specifies what data SIMION should record as the particles move through the system.

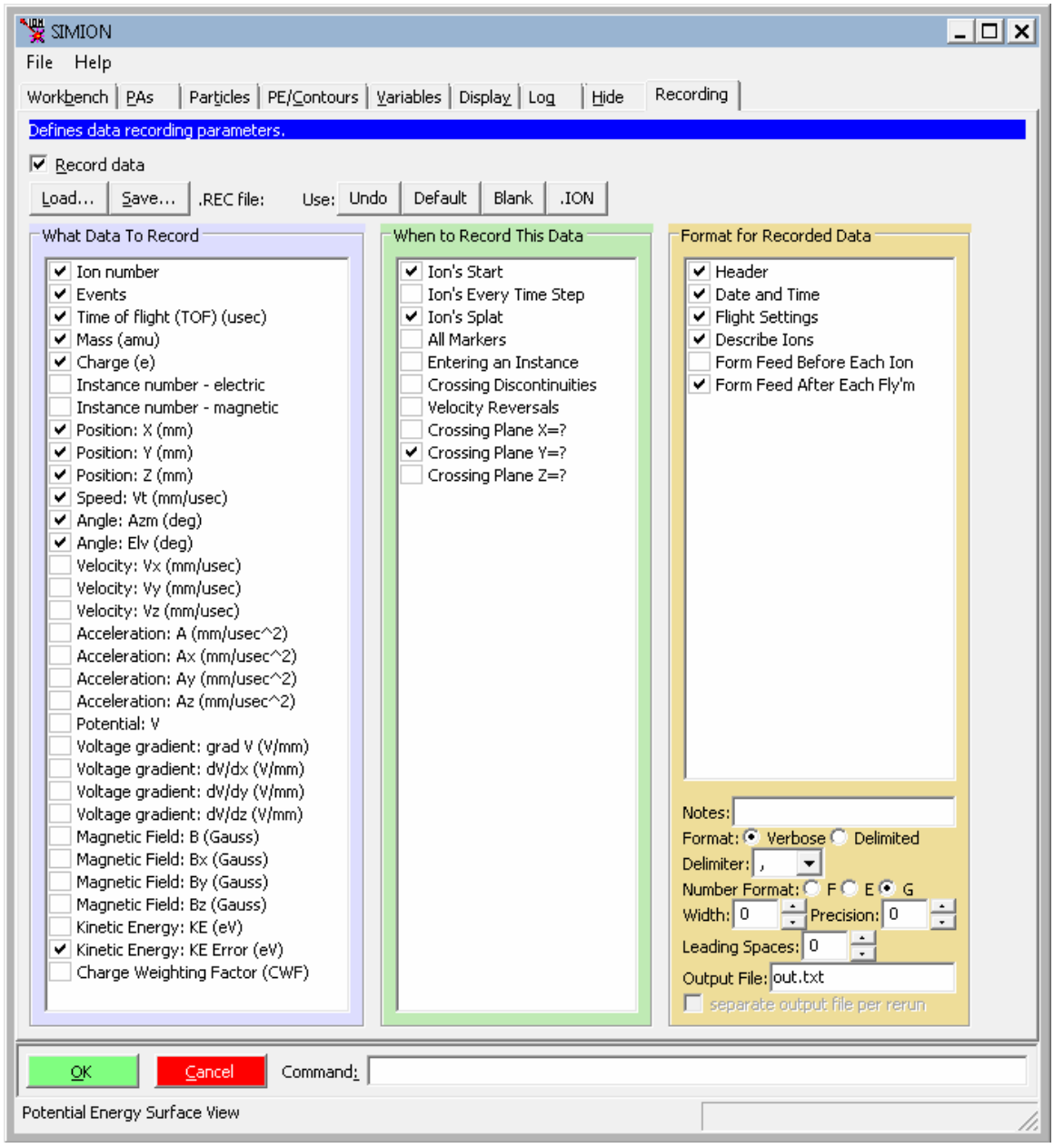


Figure 35: SIMION Data Recording options

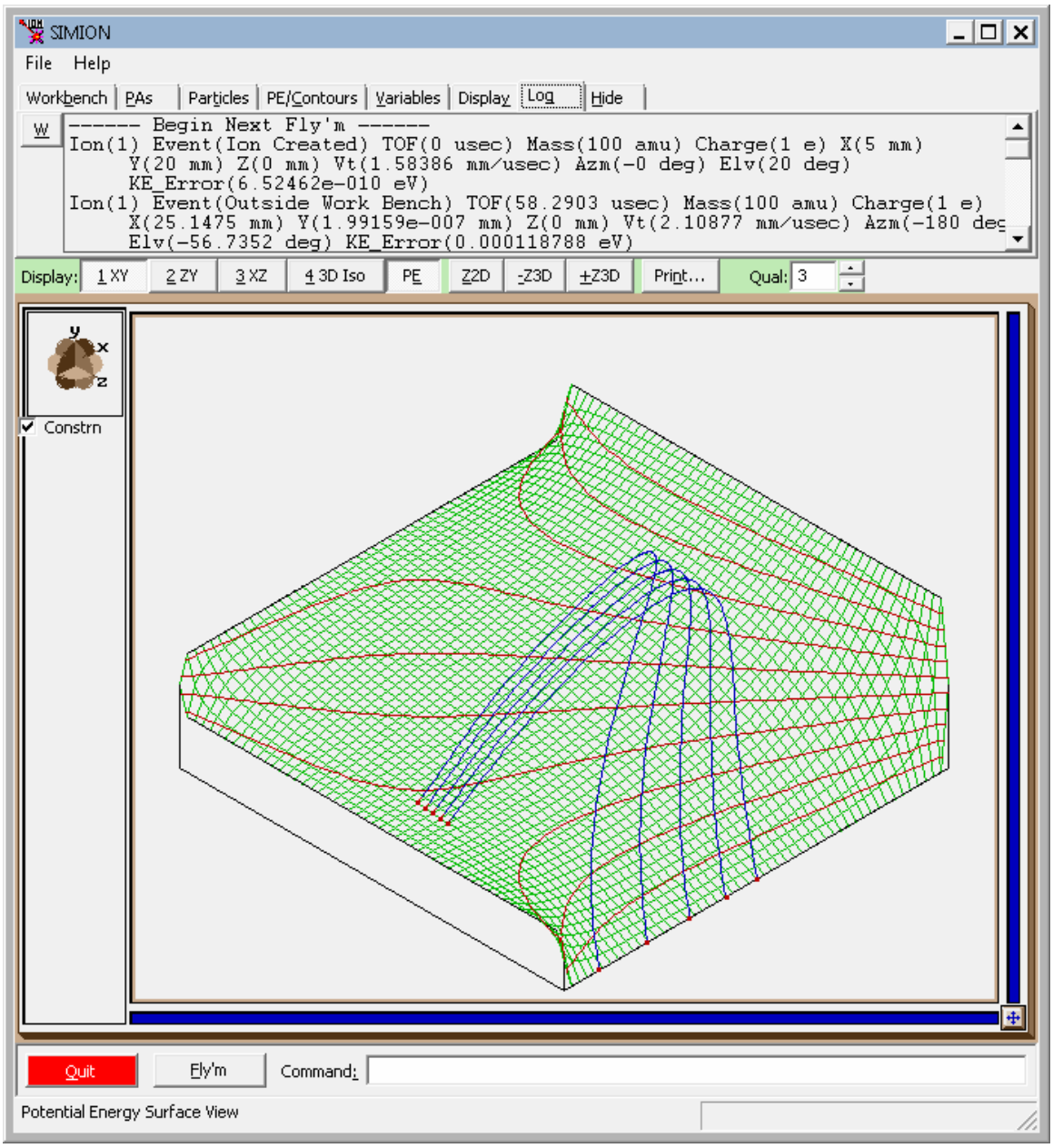


Figure 36

Data can be recording to the screen in real-time or to a text file. The text-file may even be in the delimited format for reading into Excel or another program.

6.5 Potential Array Instances (.PA and .PA0)

We've discussed potential arrays already, but a more detailed description will be given now.

A potential array instance defines a 2D or 3D electrode geometry, as well as electric or magnetic potentials at every point (electrode or non-electrode) in the geometry. Electric or magnetic fields are calculated from these potentials.

- Defines a 2D or 3D uniform **rectangular** grid (**mesh**) of points.
  - Each point is marked either as an electrode **or non-electrode** (space) point.
  - Each point has a **potential**.
- Is marked to indicate whether the point **potentials** represent electric or magnetic potentials.
- Is marked to indicate whether **symmetry** (cylindrical or planar) and/or **mirroring** (x/y/z) is applied to the mesh.

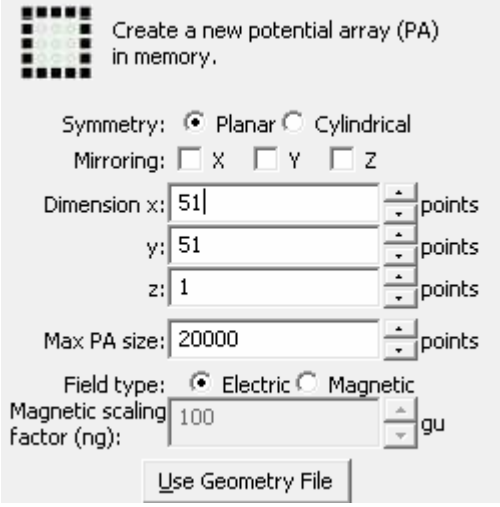


Figure 37: Creating a new potential array in SIMION

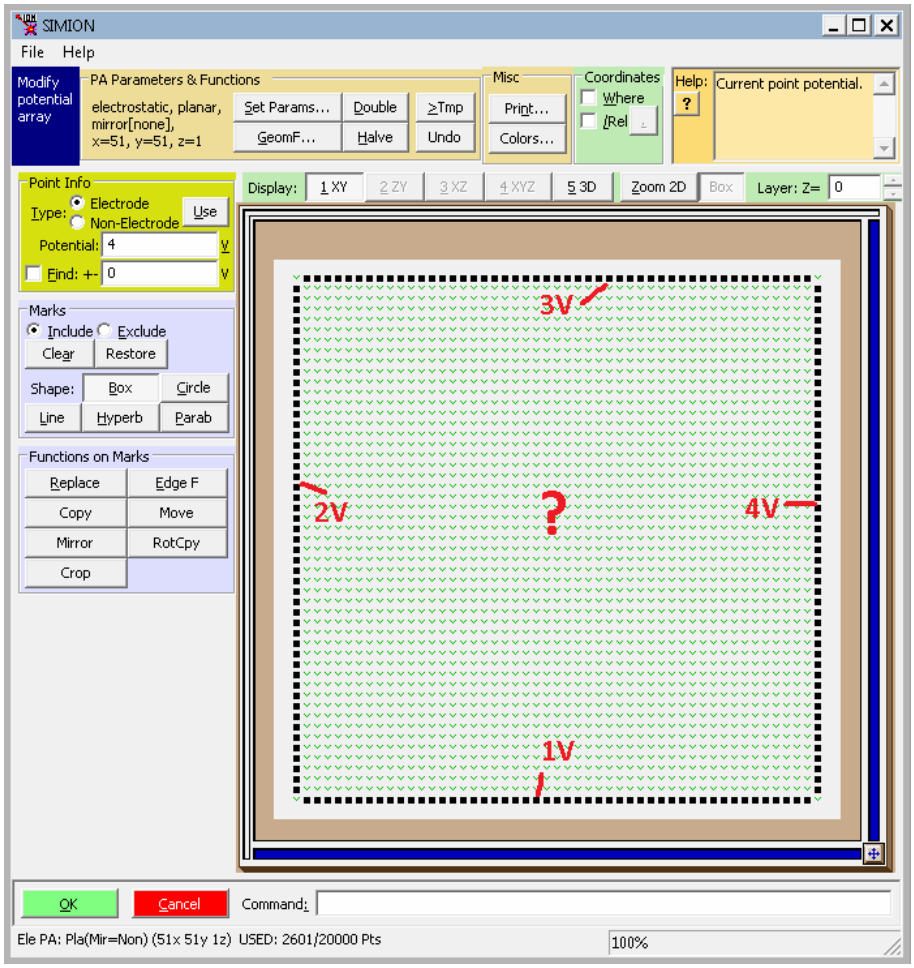


Figure 38: Example of SIMION potential array

The potential array instance is a 2D or 3D uniform rectangular grid (mesh) of points. Each point can be an electrode (black) or non-electrode (clear). Each point (electrodes and non-electrodes) has a potential.

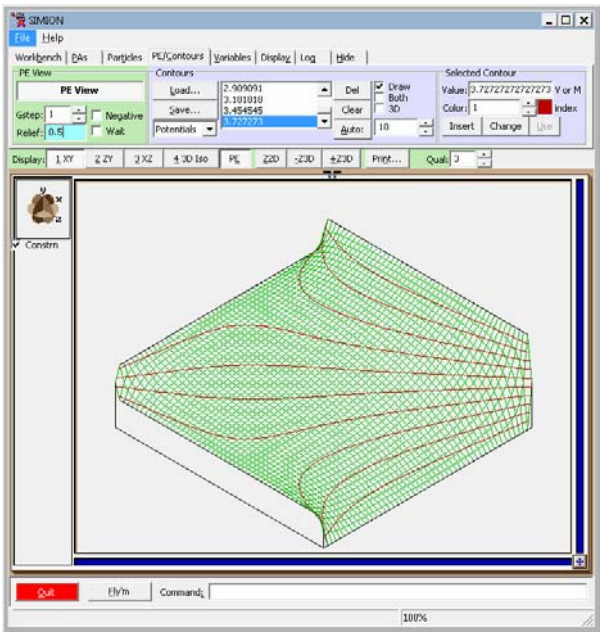


Figure 39

Here is another view of the potential array instance showing the potential (vertical axis) of each point. The electric field (black contour lines) are calculated from it.

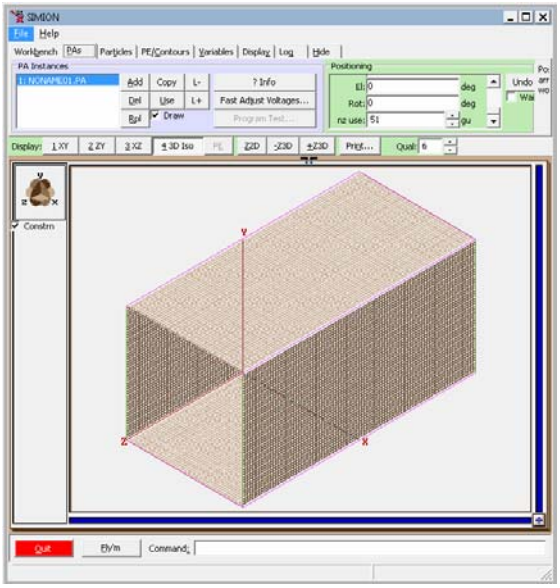


Figure 40

The 2D mesh along with symmetry/mirroring (e.g. extrude along the z-axis) can represent a 3D object in practice.

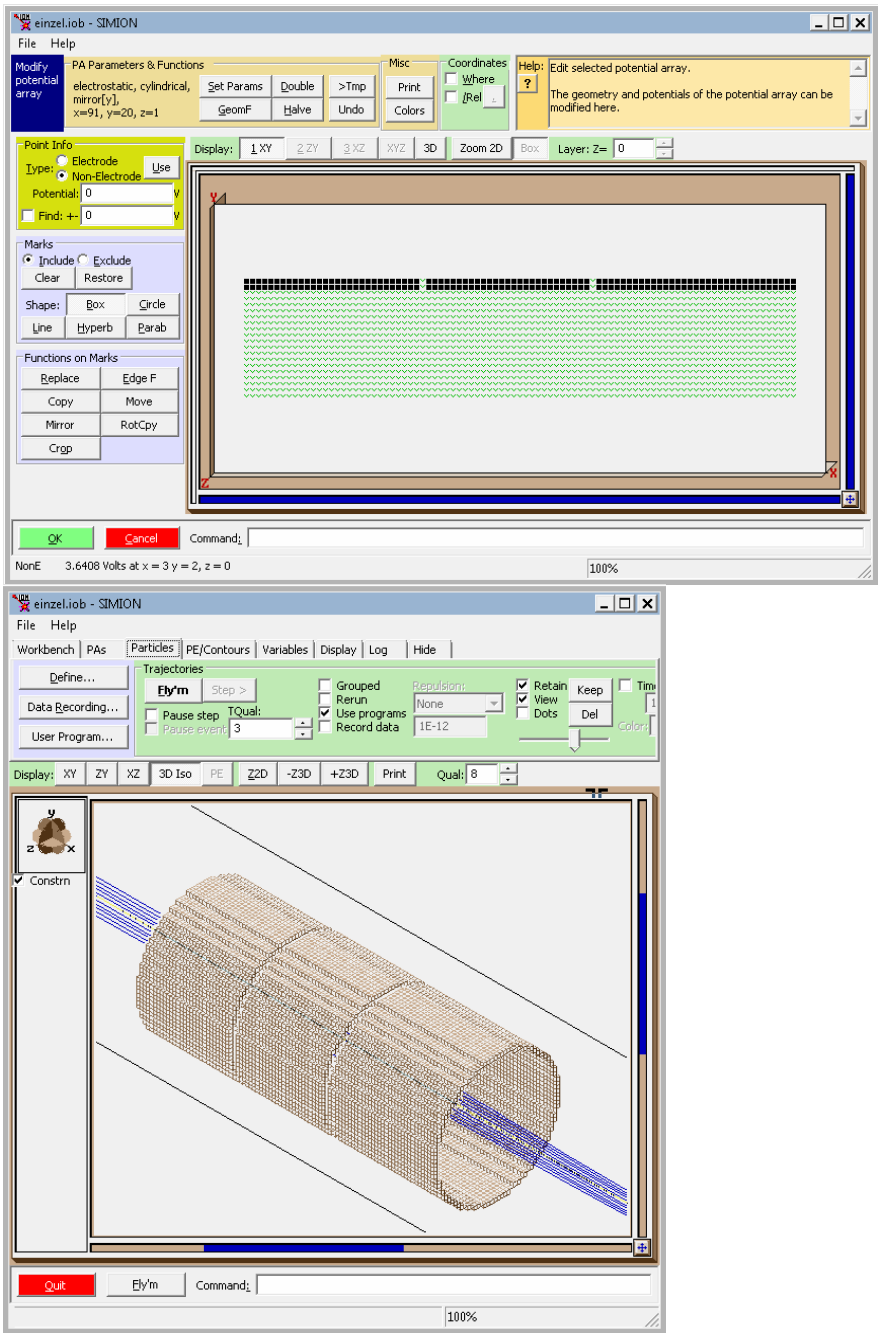


Figure 41 – Original 2D potential array (top) and physical view (bottom) after applying 2D cylindrical symmetry.

Here's a three element "Einzel" lens (bottom) formed by a 2D mesh (top) rotated around the x-axis by cylindrical symmetry.

6.6 Creating Potential Array Instances (.PA and .PA0)

for use in ion optics workbenches (.IOB files)

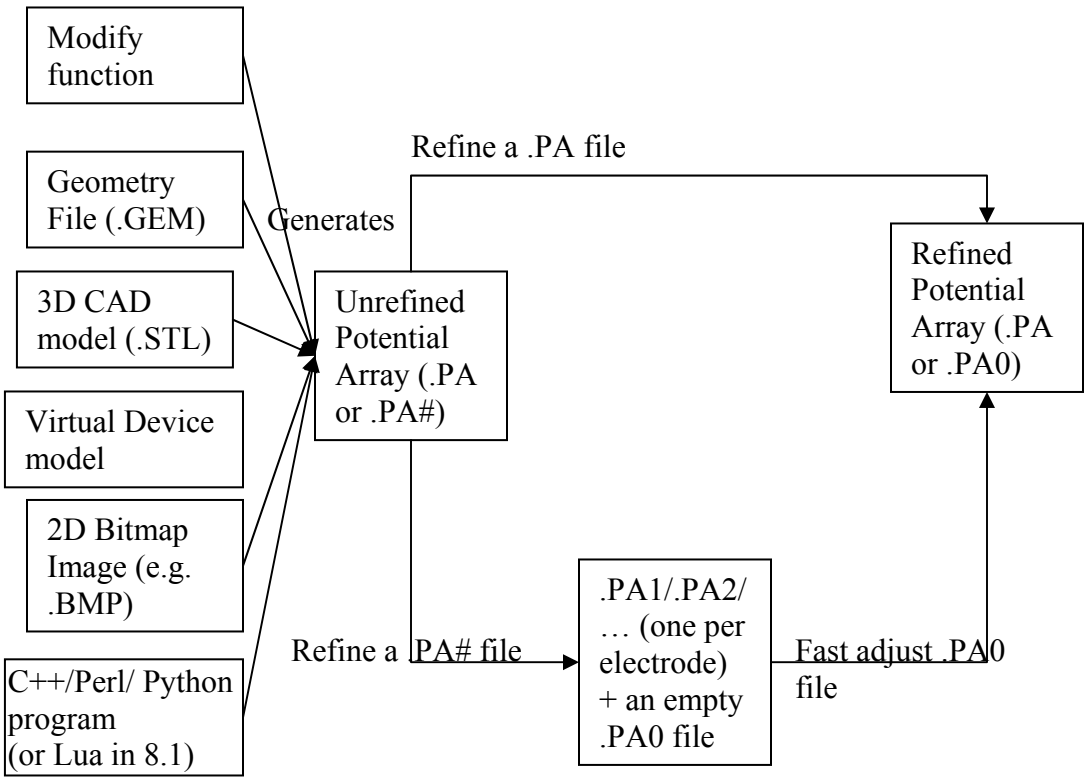


Figure 42: Different ways to create potential arrays.

The **unrefined potential array (.PA or .PA#)** contains the electrode geometry and electrode voltages. It does **not need to specify the non-electrode voltages** (there will be calculated).

The PA1/.PA2/.. files contain the solutions to the non-electrode voltages, one per electrode. There are in separate files so that we can quickly adjust voltages without re-refining).

The PA0 is essentially a linear combination of the PA1/PA2/… files (principle of superposition):

$$PA0 = k_1\, PA1 + k_2\, PA2 + k_3\, PA3 + \ldots$$

Where k_i represent the electrode voltages. This makes voltage changes quick (especially, e.g. oscillating the potentials on a quadrupole).

6.7 Geometry Definition

SIMION provides multiple ways to create geometries in potential arrays:

- 3D pixel paint-like program (Modify)
- Mathematical geometry description in a text file (GEM files)
- CAD import (using the SL Tools utility)
- Programmatically (e.g. using the SL Libraries) from C++, Perl, or Python languages. (or Lua script in 8.1)
- 3D CAD-like drawing in Virtual Device utility (not included in SIMION).

Many of the previous screenshots have shown the Modify screen, but you may prefer to draw your geometries via other methods like CAD software (e.g. SolidWorks, AutoCAD, etc.).

6.8 Geometry (.GEM) File

A GEM files is a text-file containing mathematical **constructive solid geometry (CSG)** operations. It is used to create a PA file.

In constructive solid geometries, shapes are formed by the addition and subtraction of other shapes. It's a bit like machining (form a plate and then cut a hole through it).

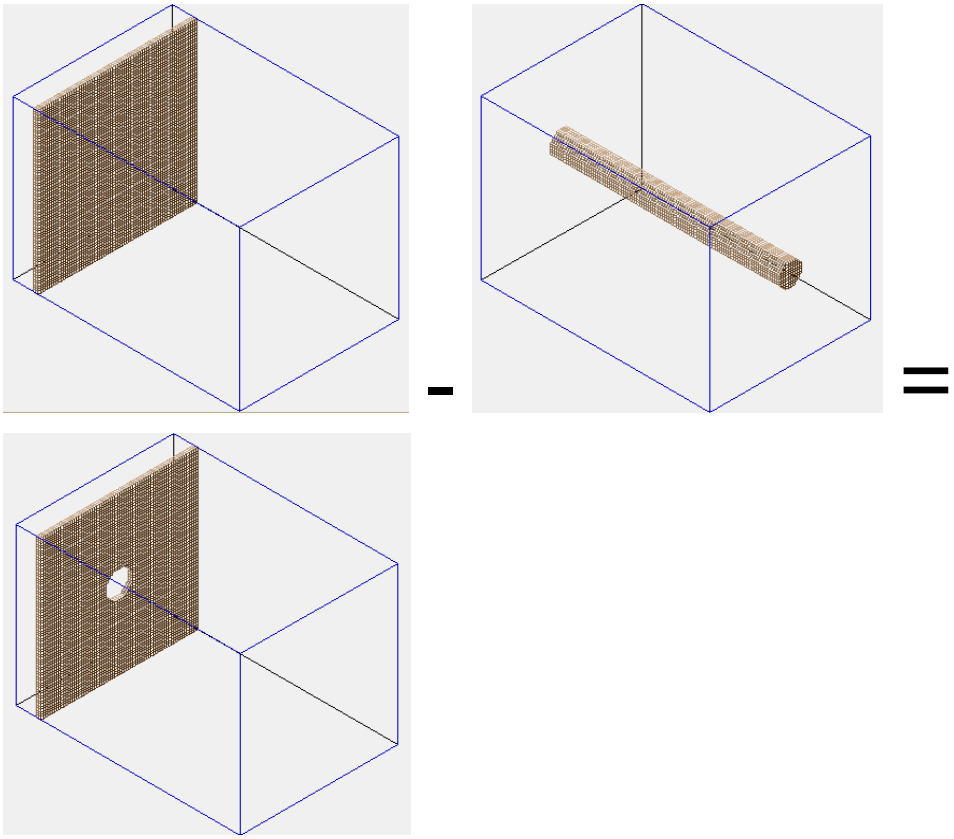


Figure 43: Constructive solid geometry operations.

Here's how to create the previous box geometry in a GEM file:

```
; box.gem
pa_define(50,50,1,planar,non-mirrored)

electrode(1) {
  fill { within { polyline(49,0,  0, 0)  }}
}
electrode(2) {
  fill { within { polyline(0, 0,  0, 49) }}
}
```

```
electrode(3) {
  fill { within { polyline(0, 49, 49,49) }}
}
electrode(4) {
  fill { within { polyline(49,49, 49,0)  }}
}
```

Figure 44: GEM file defining a rectangular mesh of 50 x 50 points with four electrodes of voltage 1, 2, 3, and 4 along each border.

```
; makes simple detector with dynode as fast adjust file
pa_define(101,71,71,planar,non_mirror)

locate(10,35,35) { ; center in 3d array 10 back in x
  electrode(0) { ; zero volt electrode
    ; create entrance plane
    fill{
      within{centered_box(0,0,2,70)}
      notin{ locate(,,,,-90){ circle(0,0,5) }}     ; hole in entrance plane
    }
  }
  electrode(1) { ; use electrode number one for dyode
    locate(5,-8) { ; offset location for dynode
      ;+5 in x and -8 in y
      include(dyn_inc.gem)   ; include dynode
    }
  }
  electrode(2) { ; use electrode number two for detector
    locate(20) { ; shift detector aiming location +20 in x
      locate(,,,,,30) { ;elevate 30 degrees (revolve detector)
        locate(20) { ;shift detector +20 from point of rotation
          include(det_inc.gem) ; include detector
        }
      }
    }
  }
}
```

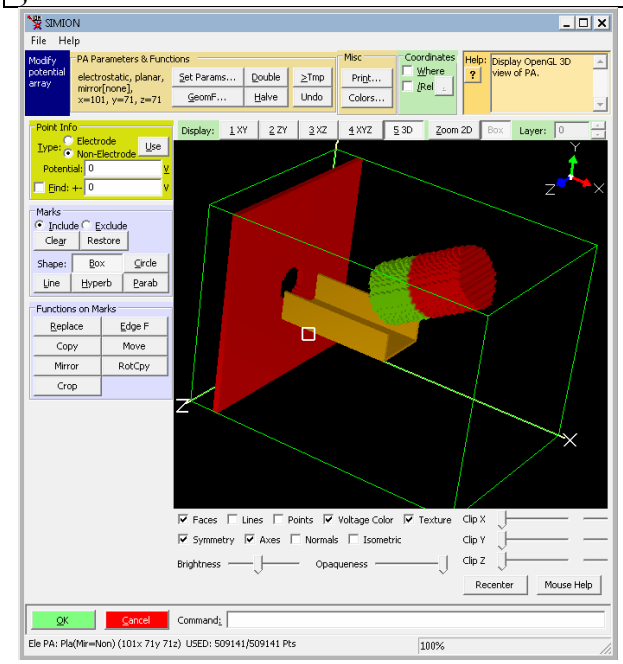


Figure 45: Geometry file example based on DETECT.GEM in SIMION.

6.9 CAD Import (STL Files)

STL is a triangular surface mesh format supported by most CAD packages (including SolidWorks, AutoCAD, …). SIMION can convert it to a potential array.

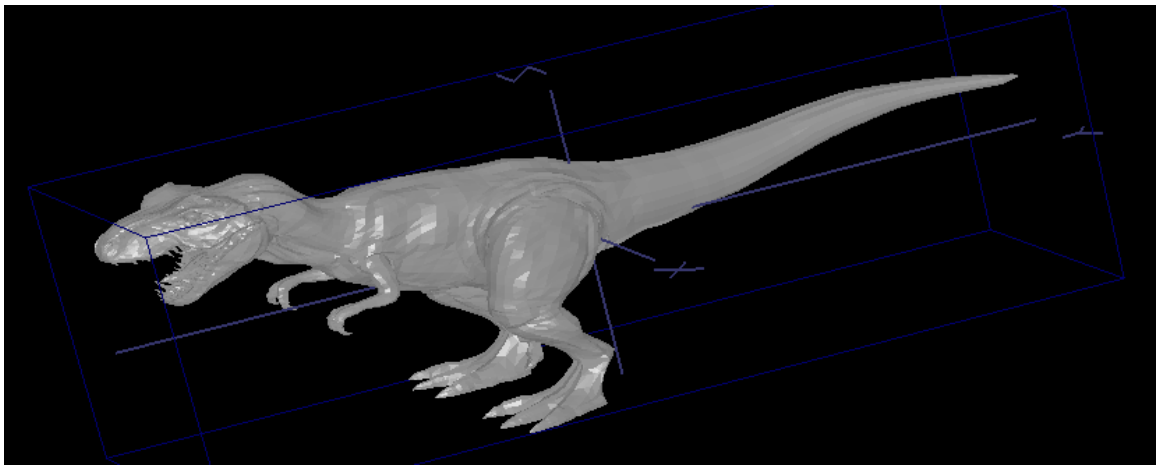


Figure 46: STL file created by a CAD package (Source: (c) 3D CAD Browser (www.3dcadbrowser.com) (2001), by Ross Blackburn). 24723 polygons, 22969 points)

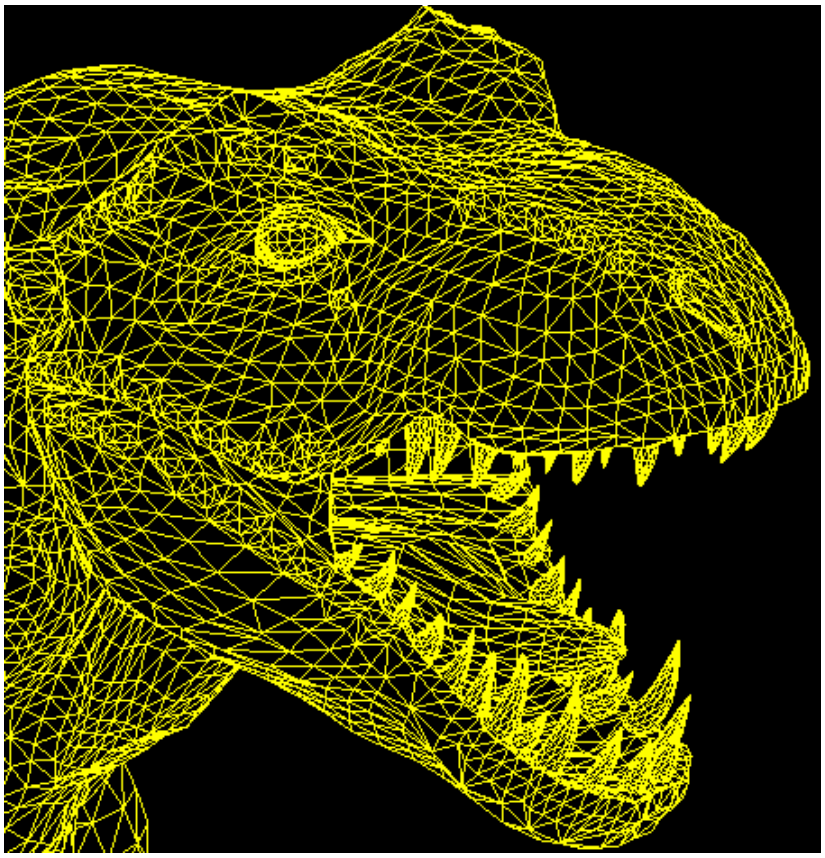


Figure 47: Zoom up of head—tiny triangles form the surface.

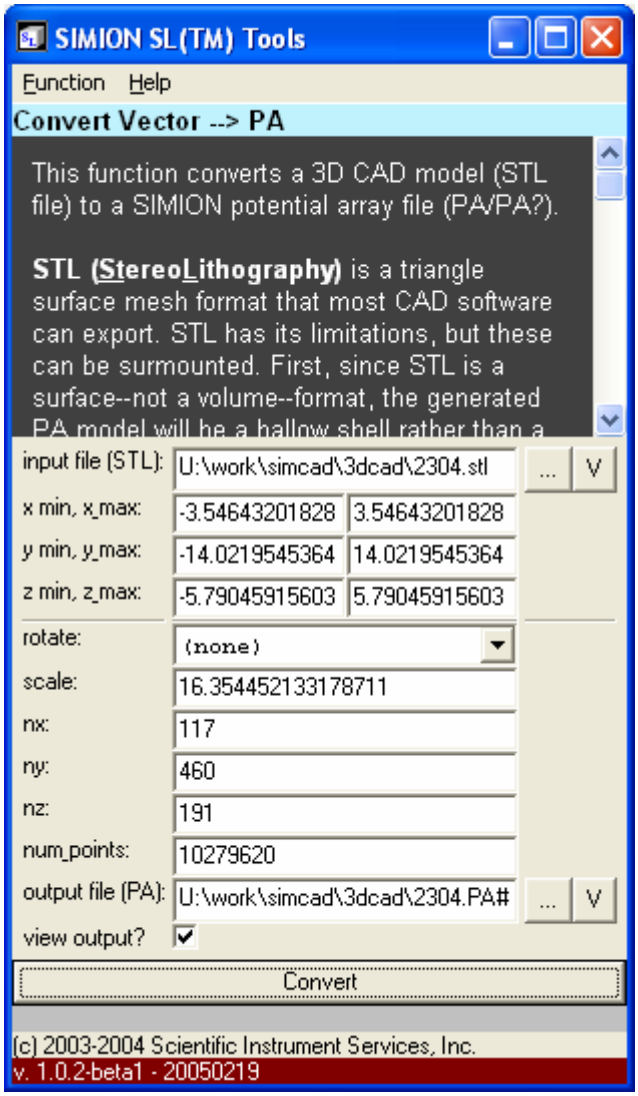


Figure 48: Convert the STL file to a PA file using the SL Tools utility.

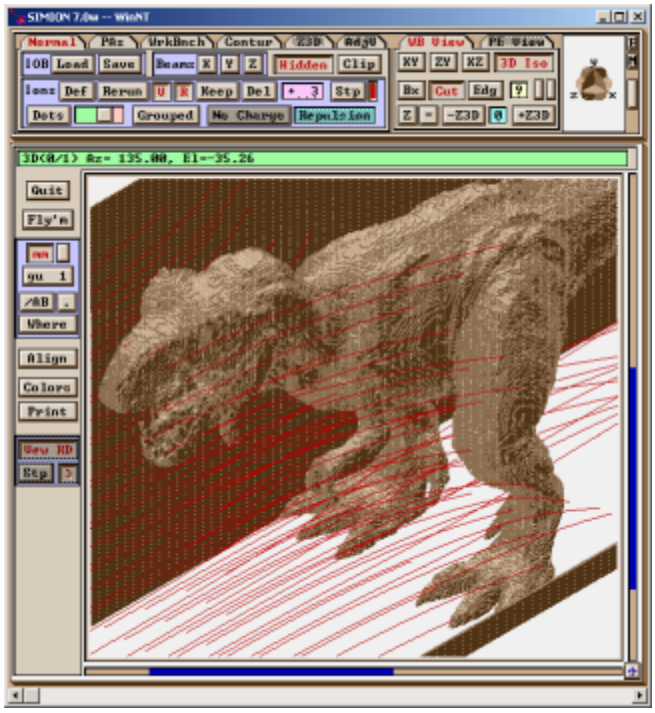


Figure 49 – Generated PA loaded into SIMION (older SIMION 7.0 version)

Above is the simi-o-saurus in SIMION. Note that the STL format is a surface (not volume) format. Here's a CAT-scan of the simi-o-saurus's head:

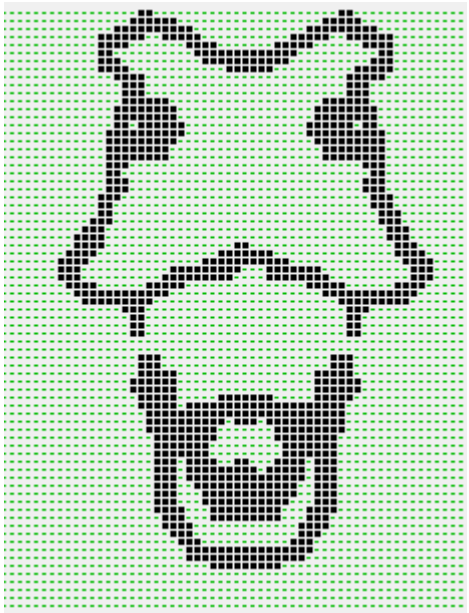


Figure 50

It's hollow! But that's ok. Why? The Laplace equation needs the boundary (not the inside) of electrodes to be defined (first uniqueness theorem). The results are identical with or without a brain.

6.10 2D Bitmap (e.g. .BMP) files

Another option (less frequently used) is to convert 2D bitmaps to potential arrays. This mostly doesn't apply to 3D arrays though.

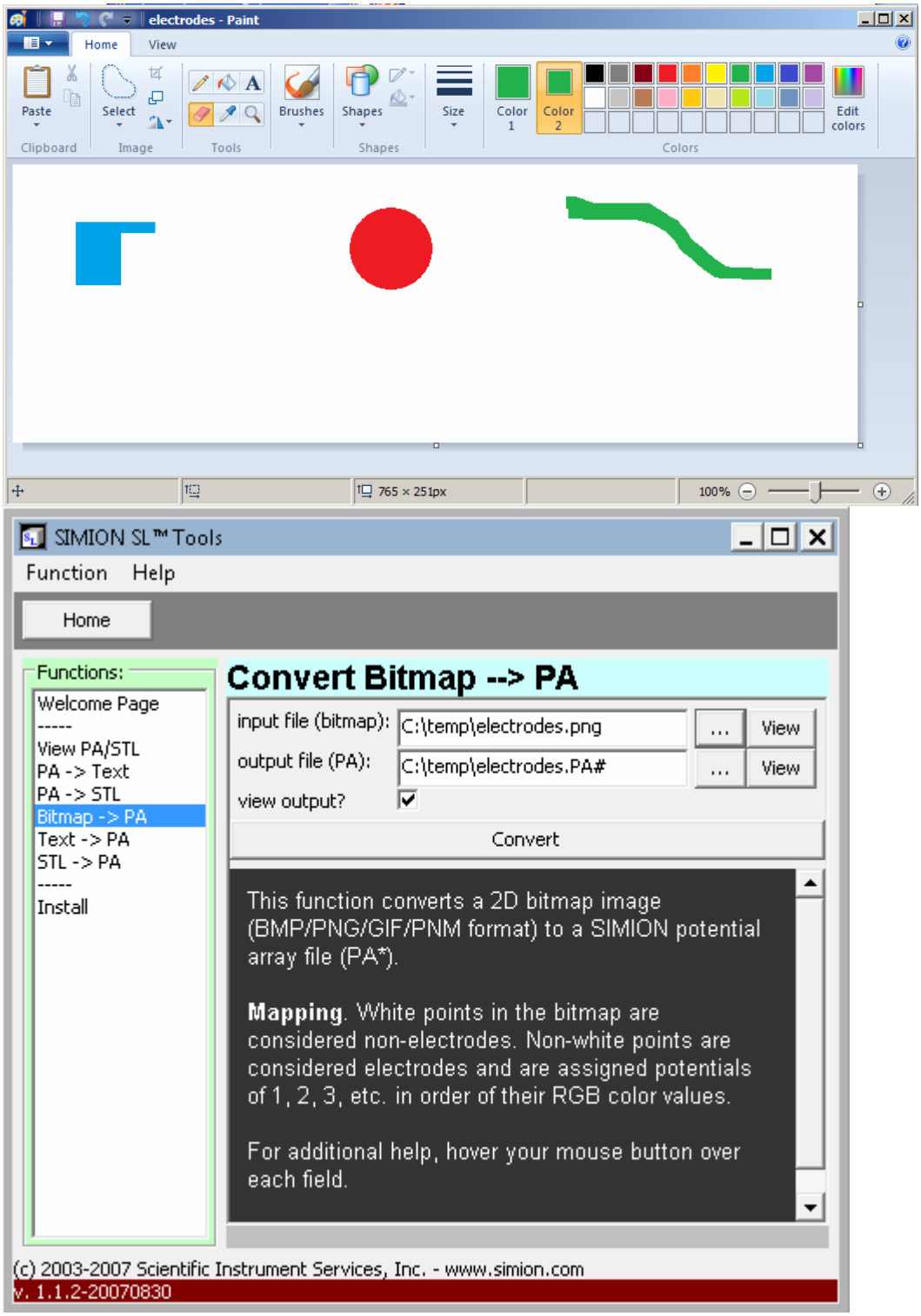


Figure 51: Converting 2D bitmap image into SIMION potential array using SIMION SL Tools utility.

(left) Creating a 2D image using the MS Paint (the colors indicate the voltages, 1V, 2V, 3V). **(right)** Converting the bitmap file to a PA file using the SL Tools utility.

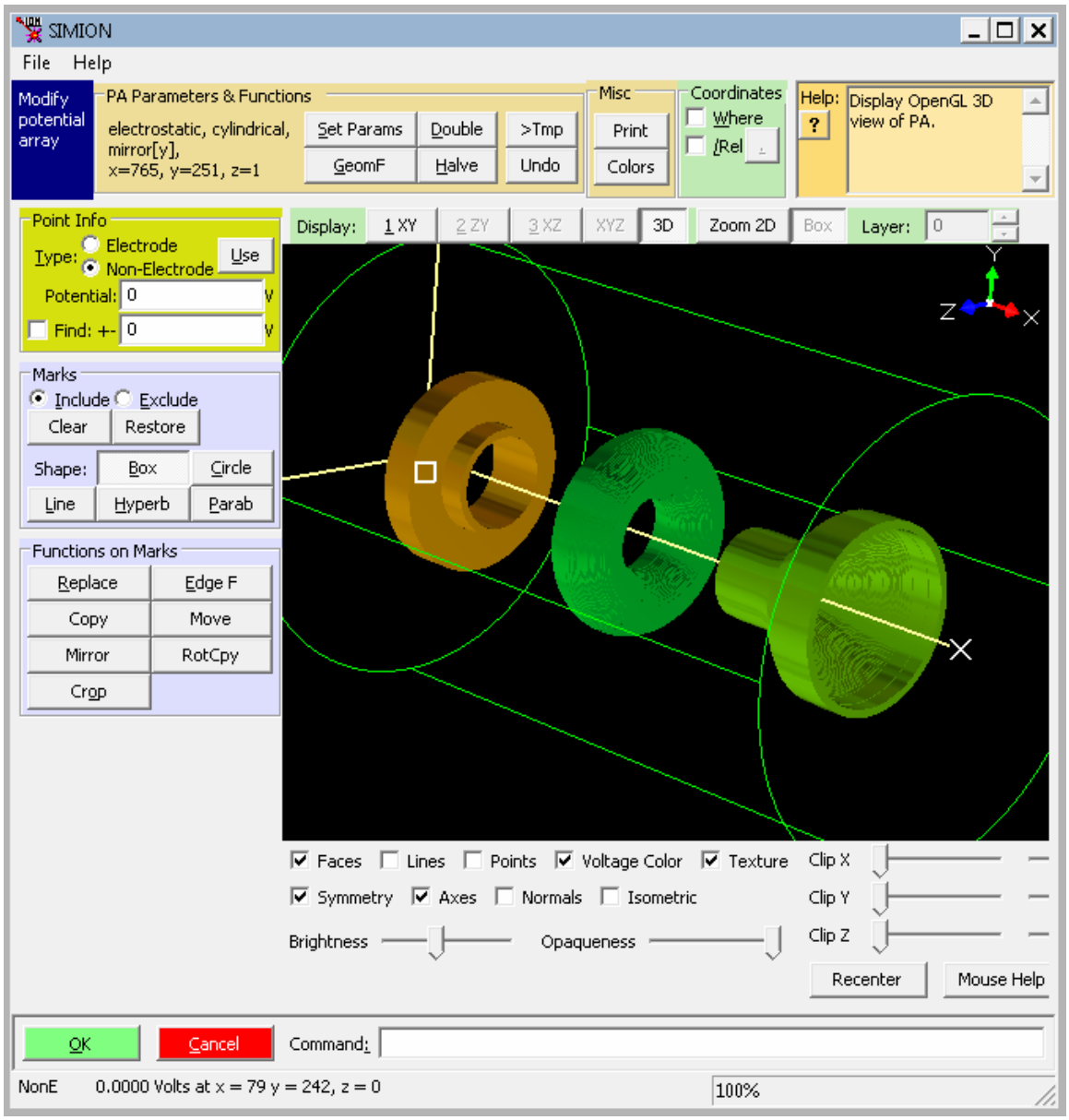


Figure 52

The PA (after applying cylindrical symmetry in Modify) is shown here in SIMION.