Defining and Editing Array Geometry

Introduction

It is assumed that you have read the discussion of potential arrays in Chapter 2 and have scanned Chapter 4. If not, read the material before proceeding further. ***This chapter covers the mechanics of defining and editing of potential array geometry.*** Chapter 6 discusses how to properly define, refine, and adjust the two array definition types (*.PA and .PA#*). Chapter 7 explains how to project arrays via array instances into the workbench volume.

The geometry creating/editing methods discussed in this chapter involve the **Double**, **Halve**, and **Modify** functions as well as a few comments concerning geometry files (*an advanced SIMION feature - Appendix J*). For the sake of this discussion, array geometry means the methods employed to define the shapes of electrode/pole boundaries in potential arrays in order to simulate real-world electrostatic or magnetic fields. ***One potential array can only simulate either electrostatic or magnetic fields.*** Thus combined electrostatic and magnetic fields require ***two*** superimposed potential array instances (*one electrostatic and one magnetic*).

Changed Arrays are Flagged

When you change the type, size, symmetry, mirroring, and/or geometry of a potential array SIMION automatically flags the array as modified. ***This is done by placing a red outline around the PA's button in the PA list window on the Main Menu Screen.*** When you save the changes to disk the red outline will be removed. If PA changes are still unsaved at program exit SIMION ask you to

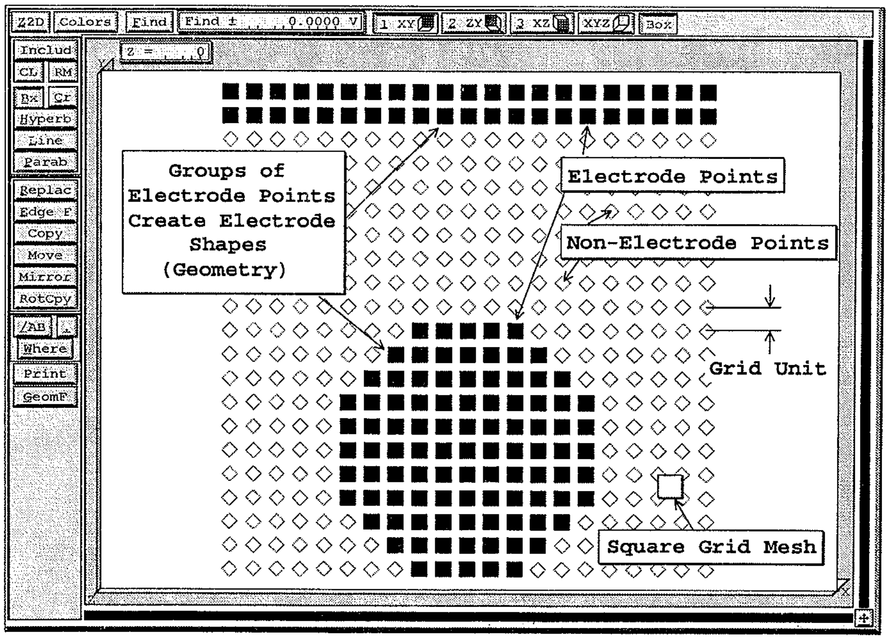


Figure 5-1 Potential arrays use square meshes of points to define electrode/pole geometry

selectively save the desired potential array changes before exiting the program.

Array Geometry Involves Many Things

The use of potential arrays composed of square (*2D*) or cubic (*3D*) mesh grid points to simulate real-world electrostatic or magnetic field devices is at best an approximation of reality (*Figure 5-1*). ***The extent that SIMION simulates the real-world accurately is the result of the care you take in understanding and defining array geometry properly.***

Potential Arrays are Like Pixel Displays

SIMION's potential arrays contain points arranged to form square or cubic point meshes (*grids*). ***The separation distance between two adjacent grid mesh points is called a grid unit.*** Individual points can be flagged as electrode/pole or non-electrode/non-pole. ***Groups of adjacent electrode/pole points having the same potential serve to define (approximate) electrode or pole shapes.***

Potential arrays are thus very similar to pixel displays. The higher the pixel density (*grid density*) the better the view (*e.g. simulation*). However, just like pixel displays, a relatively low point density (*course*) potential array can successfully model reality very well if reality shares the integral spacing of the potential array (*all edges actually occur at array points*) and the shapes to be simulated are planes (*or cylinders and planes in cylindrical symmetry*) that are integrally aligned with the array points. ***The better reality fits the modeling method, the better the modeling method will simulate reality.*** This is a very important concept to remember when you are choosing the spacing and shapes of your electrodes/poles.

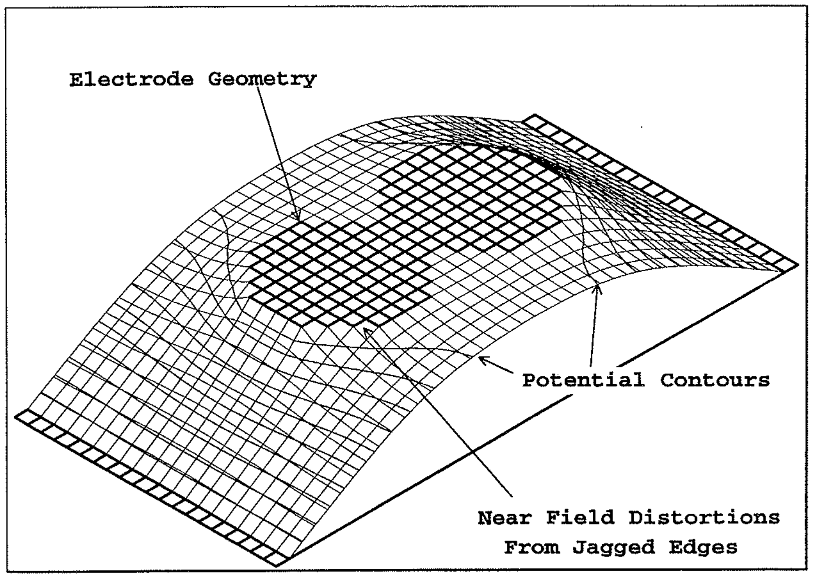


Figure 5-2 Potential fields can become jagged near irregularly shaped electrode boundaries

The Array Density Verses Accuracy Problem

If you are modeling complex shapes with non-integral alignment and spacing (*relative to the potential array point grid pattern*) the problem of accuracy becomes more complex (*Figure 5-2*).

Using points to model something like a hyperbolic shaped electrode introduces a collection of issues that can be mitigated by increasing the potential array size (*e.g. its array point density*).

Surface Jags Near Electrodes/Poles

The accuracy of the fields near slanted or curved electrode/pole surfaces can suffer from the jags (*steps in electrode/pole points to simulate surface shapes*). Ions flying very close to (*or originating from*) these jagged electrodes/poles will definitely ***not*** have realistic trajectories.

Increasing the grid density reduces the size of the jags and allows you to fly ions closer to these surfaces (*in terms of mm*). In this case, the averaging nature of the Laplace equation comes to your rescue by smoothing out the fields in the far field region (*3 or more grid units off the irregular surface*).

This, however, doesn't help ions originating from slanted or curved surfaces. The best trick here is to start the ions out a bit (*around one grid unit away from the surface*) with the energies and directions appropriate to their adjusted starting location.

The Issue of Field Curvature

Another frequently unrecognized issue involves the accuracy with which a given grid density models a particular field's curvature (*Figure 5.2*). Think of potential array grids as square plates that ***must*** be warped a bit to match field shapes. ***The less each grid needs to be warped the better the chance that the potential array accurately models reality. Simply refining an array to a very low error term does not guarantee any particular level of accuracy***. If your array density is too course for your field's curvature you will encounter significant systematic errors in field estimates that high accuracy refining simply will not remove.

Laplace to the Rescue

There are several important issues to keep in mind despite the above gloom and doom. First, the Laplace equation is fundamentally an averaging process. This means that jags and the like tend to be averaged out quite well in the far field (*Figure 5-2*). ***It is surprising how crude the surface of a spherical ESA can be (in terms of jags) and have close to theoretical far field ion trajectories.***

Moreover, field imperfections are only important to the extent that they adversely impact ion trajectories. There are two relatively painless methods to establish whether you have significant problems: First, double the array size and see what impact this has on ion trajectories. If things don't change much your grid density is probably OK. Second, try to back-fly your ions from their termination points (*using reversed directions and termination energies*). The ability to duplicate ion trajectories bi-directionally (*at least approximately*) means that the modeling is reasonably self-consistent. ***However, neither test guarantees that the results are really correct. There is no substitute for testing the as-built.***

Defining Solid or Grid Boundaries

Electrodes/poles are represented in potential arrays as groups of electrode/pole points with the same potential. SIMION can treat a collection of these electrode/pole points as either a solid object or a grid. The difference is that ions ***will*** pass through a grid but ***will not*** pass through a solid object (*Figure 5-3*).

Defining Solid Electrodes/Poles

SIMION kills (*splats*) an ion whenever the ion finds itself within a 2D grid square with electrode/pole points at ***all four corner points*** of the grid or within a 3D grid cube with electrode/pole points at ***all eight corner points*** of the grid cube. ***Thus an electrode/pole must be***

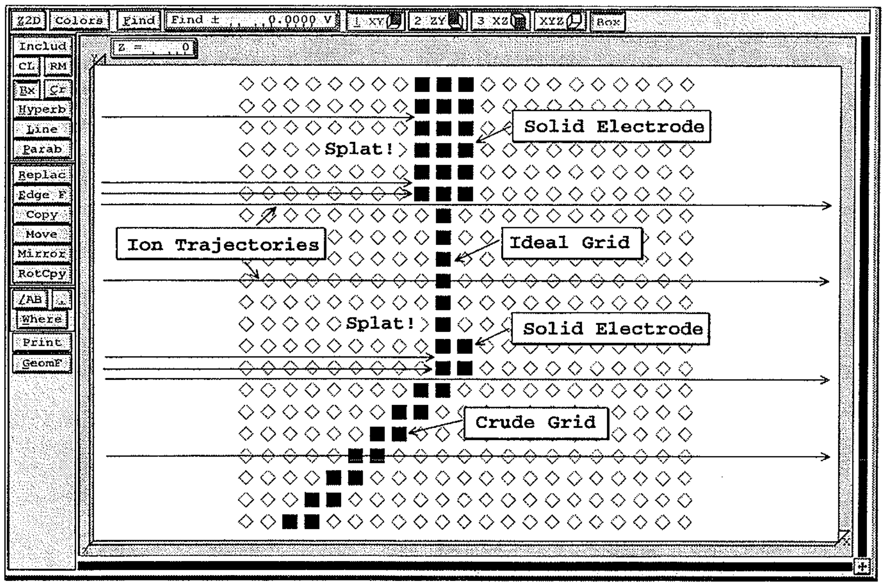


Figure 5-3 Simulated ion trajectories through electrode geometry that demonstrates solid electrodes and grids

defined in terms of grid squares (2D) or grid cubes (3D) for SIMION to consider it solid in any particular region.

Minimum Flying Separation Between Electrodes/Poles

Note: The potentials of the grid square or cube corner electrode/pole points do not have to be the same for the ion to splat. Thus, SIMION requires that at least one non-electrode/non-pole point separates ***solid*** electrodes or poles of differing potentials (*two or more grid units of separation*) for ions to be allowed to fly between them (*use wider grid unit separations for better field simulation purposes*).

Splats When Electrostatic and Magnetic PAs Overlap

When an ion finds itself within the volume of overlap of an electrostatic and a magnetic array instance, ***SIMION checks*** the ion's location against ***both*** array instances for splat conditions. ***If the ion splats in either, it is considered to have died.***

Defining Electrode Grids

SIMION supports the notion of grids for electrostatic arrays (*works for magnetic instances too - though probably not as useful*). ***A grid is any collection of electrostatic grid points that do not fit the solid electrode/pole definition above.*** The important point is that ions ***do not*** splat (*die*) when they pass through grids.

The Definition of Simple Grids

The simplest (*and most accurately modeled*) grids are created by electrode points that create planes (*or planes and cylinders in cylindrical symmetry*) in instances that are integrally aligned relative to the potential array (*x = c or y = c lines in 2D or x = c, y = c, or z = c planes in 3D, where c is an integer*).

Issues with Curved or Slanted Surface Grids

Curved surfaces (*e.g. a hyperbolic grid*) or slanted surfaces (*e.g.* $y = mx + b$) that model grids will introduce the jags. Since ions are probably expected to fly through these grids it is important to realize what to expect. ***The worst case involves decelerating curved/slanted grids that pass ions at a very low energies***. If the ion's energy as it passes through the grid is not a couple of orders of magnitude greater than the energy drop in the jags region (*within plus or minus three grid units of the grid's rough surface*) the ion's trajectory will be highly suspect. Increasing the grid density can reduce this problem. ***If at all possible, try to transverse curved/slanted grids with reasonably energetic ions***.

The Notion of Ideal Grids

The typical grid defined within SIMION involves a connected straight or jagged (*sloped or curved*) line of electrode/pole points. This type grid is treated as an ideal grid. That is, SIMION assumes that the grid holes are infinitely small (*no field penetration through the holes*) and that the grid passes 100% of the ions. ***Would that we could actually build such grids!*** The reason for this is simply that each point on the boundary of the grid is an electrode/pole point (*with fixed potential*). Thus there is no path for fields to penetrate through the grid.

Constructing Non-Ideal Grids

Non-ideal grids are modeled by replacing a percentage of electrode/pole points along the grid boundary with non-electrode/non-pole points. This allows fields to penetrate through the grid. A relatively open grid might have only every 5th point as an electrode/pole point. In the case of 2D arrays these points actually model wires (*wire circles in cylindrical*). ***Non-ideal crossed wire grids must be simulated in 3D arrays***.

Each point that acts as a wire can be thought of as having a diameter of about 0.4 grid units in terms of its field effects. However, single points do not stop ions. ***This means all ions will pass through grids with single point wires***. To create a non-ideal grid that captures ions requires that the wires be solid (*formed by one or more connected four point grid meshes in the manner solid electrodes/poles are created*).

Doubling or Halving Potential Arrays

SIMION provides quick methods to double or halve the size (*and thus density*) of the currently selected potential array (*its button depressed on Main Menu Screen*). These functions ***attempt*** to keep the array's existing geometry intact.

If a workbench instanced (*referenced*) array is doubled or halved, SIMION will automatically detect that the array has been doubled or halved when you re-enter the **View** function. SIMION will ask if you want to have the instance's definition sized and adjusted to fit the doubled or halved PA instance exactly where the previous PA instance fit (*handy*). ***You should then save the adjusted .IOB file***.

The Double Function

The **Double** function can be used to double the size of any potential array. ***The main use for doubling is to increase the grid density of the array to improve its ability to model the potential fields created by its electrodes/poles***. Array doubling eats up more RAM (*4x in 2D and 8x in 3D*). Thus you should avoid overdoing it.

Procedure to Double an Array

The procedure to double a potential array is simple (*starting at Main Menu Screen*):

1. Select the PA to double (*depress its button if not already depressed*).
2. Click the **Double** button or hit the **<d>** key.
3. SIMION will ask you if you're sure you want to double the potential array.
4. You also will be asked if you want to interpolate electrode/pole potentials. Select the default (**NO**) if you don't know what this means (*discussed below*).

Remember to Refine Doubled Arrays

The doubling process inserts new points into the array and the potentials of existing non-electrode/non-pole points become unreliable. ***Thus you must always refine any doubled potential array before you attempt to use it for ion flying.***

Memory Requirements for Doubled Arrays

Each time you double a 2D array you increase its size by approximately a factor of four (*3D arrays increase by about a factor of eight*). The exact size required is:

New Size = (2nx -1)(2ny -1)(2nz -1)
Where nx, ny, and nz are the current array dimensions

SIMION will refuse to double an array in a PA memory region with insufficient memory allocated for the doubled potential array size. Your recourse when this happens is to save the PA, remove all PAs from RAM, and reload the PA into a larger allocated memory region (**Max PA Size** *panel set to needed memory value or above* ***before*** *reloading the PA*).

How SIMION Conserves Geometry

When you double an array, SIMION places rows and columns of non-electrode/non-pole points between existing points in 2D arrays (*and also inserts intervening planes of points in 3D arrays*). It then tries to decide which of these newly inserted points to convert to electrode/pole points. ***The basic rule is if adjacent points in the x, y, or z directions are electrodes/poles of the same potential then the point in question is converted to an electrode/pole of that potential too.*** The process requires multiple passes through the array to fully resolve the fate of all the new points.

How Successful is this Approach?

The advantage of this process is that grids are conserved (*not made solid by doubling*). Electrode/pole surfaces that are ***aligned*** with the potential array are also re-sized faithfully. ***Unfortunately, diagonal and curved surfaces become more jagged.***

One Way to Fix Jagged Surfaces in Doubled Arrays

The **Modify** function can be used to examine the results of doubling. If there are jagged surfaces that need a bit of smoothing, use the various **Modify** filling tools (*single points, lines, circles, and boxes*) to smooth out your array.

Reducing Jagged Surfaces Via Geometry Files

The best way to painlessly reduce jagged surfaces when doubling is to make use of geometry files. If all the geometry for the potential array is defined in a geometry file (*see Appendix J*), smooth doubling becomes a snap! The following procedure shows you how:

1. **Double** the potential array.

2. Enter the **Modify** Function and Click the **GeomF** button to access geometry files.
3. Select the appropriate geometry file (*if not already selected*).
4. Erase the geometry definitions in the potential array (*using the* **erase** *button*).
5. Set the scaling factor to 2.0 to adjust for doubling (*e.g. 4.0 if doubled twice*).
6. Insert geometry into potential array (*using the* **insert** *button*).

The beauty of using geometry files is that SIMION automatically creates the optimal geometry for the grid density of the array (*providing you scaled properly*). All surfaces converge on their ideal shape and locations as the grid density is increased (*through successive doubling*). The only limit is RAM or patience with virtual speeds (*as always*).

Interpolating Electrode/Pole Potentials

The **Double** function also supports interpolation of electrode/pole potentials as an option. When this option is active, SIMION converts any newly inserted point (*due to the doubling process*) to an ***interpolated*** electrode/pole point if it is adjacent to two electrode points of ***differing*** potentials. The actual potential given to this new point is the average (*mean*) of the potentials of the two adjacent electrode/pole points. This option is useful in certain special cases:

Doubling Surfaces with Non-Uniform Potentials

Interpolation is useful when doubling a magnetic array with non-uniform surface potentials (*surface potential gradients*). Doubling with the interpolation option active preserves the existing gradients (*at least in a piece-wise linear sense*) in the doubled array.

Interfacing Fields Between Potential Arrays

The interpolating option is also useful in creating an electrode gradient boundary around a potential array to allow it to interface properly into a larger potential array (*e.g. an emission point inside a larger cavity*).

For example, let's say we want a high point density model of a field emission point inside some other potential array. One approach that will work is to create a model of the outer cavity including the point itself (*somewhat crude*). Now refine this array and save it.

The next step is to create the high detail inner potential array. This could be done by starting with a refined copy of the outer potential array and modifying it. The **Modify** function would be used in the following way to do this. The **Replace** function would be used with **Find** active (*with appropriate boundary marking*) to convert the ***edge*** points bounding a small area around the emission point into electrode points ***without*** changing their potentials. Then the **Move** function would be used to move this small array region ***including*** its boundary points to lower left corner of array, and finally, the array would be downsized to just include this small array region. **Double** would then be used with the electrode interpolation option active (*one or more times*) to obtain a higher density array. **Modify** would then be used to better define the point's shape, followed by **Refine**, and finally by removing the outer boundary of electrode points with **Modify** (*optional*).

At this point, an array instance of the point's detailed array can be superimposed on the array instance of the outer cavity. The use of this approach insures more field consistency across the instance boundaries (*though not necessarily correctness*).

The Halve Function

The **Halve** function can be used to halve the size of any potential array. Halving is the exact reverse of doubling. ***Its main use is for reducing large 2D arrays to a reasonable size so they can be converted to manageable 3D arrays with* Modify.**

Procedure to Halve an Array

The procedure to halve a potential array is simple (*starting at Main Menu Screen*):

1. Select the PA to halve (*depress its button if not already depressed*).
2. Click the **Halve** button or hit the **<h>** key.
3. SIMION will ask you if you're sure you want to halve the potential array.
4. You also will be asked which halving method to use. Select the default method if you don't know what this means (*discussed below*).

Remember to Refine Halved Arrays

The halving process deletes points from the array and the potentials of existing non-electrode/non-pole points become unreliable. ***Thus you must always refine any halved potential array before you attempt to use it for ion flying.***

How SIMION Conserves Geometry

When you halve an array, SIMION uses one of two methods (*selected by you*) to ***attempt*** to conserve array geometry (*emphasizing conserving electrodes/poles or minimizing distortions*).

Copying Alternate Points (Default Method)

The default array reduction method just copies alternate points into the new reduced size array. This approach introduces the minimum amount of geometry distortion. ***However, grids and other thin details can be missed.***

Conserving Electrode/Pole Points

The second option tries to conserve electrode points by looking at the four (*2D*) or eight (*3D*) points that are to be converted into a single point in the halved array. If any of these points is an electrode point the equivalent point in the halved array will be made an electrode point too. ***While this approach conserves grids and thin electrodes it also can introduce more distortion.***

Examine an Array After Halving

If you **halve** an array be sure to use **Modify** to examine/correct for distortions. ***Remember an array can be doubled and then halved without distortion (using the default methods).*** However, the converse can introduce geometry distortions.

Reducing Distortions Via Geometry Files

The best way to painlessly reduce distortions when halving is make use of geometry files. If all the geometry for the potential array is defined in a geometry file (*see Appendix J*), smoother halving becomes a snap! The following procedure shows you how:

1. **Halve** the potential array.
2. Enter the **Modify** Function and Click the **GeomF** button to access geometry files.
3. Select the appropriate geometry file (*if not already selected*).
4. Erase the geometry definitions in the potential array (*using the erase button*).
5. Set the scaling factor to 0.5 to adjust for halving (*e.g. 0.25 if halved twice*).
6. Insert geometry into potential array (*using the insert button*).

The beauty of using geometry files is that SIMION automatically creates the optimal geometry for the grid density of the array (providing you scaled properly). All surfaces retain the best approximation of their shape and location as the grid density is decreased (*through successive halving*). ***Note: When the array gets too small for its geometry, even the optimal can get pretty bad.***

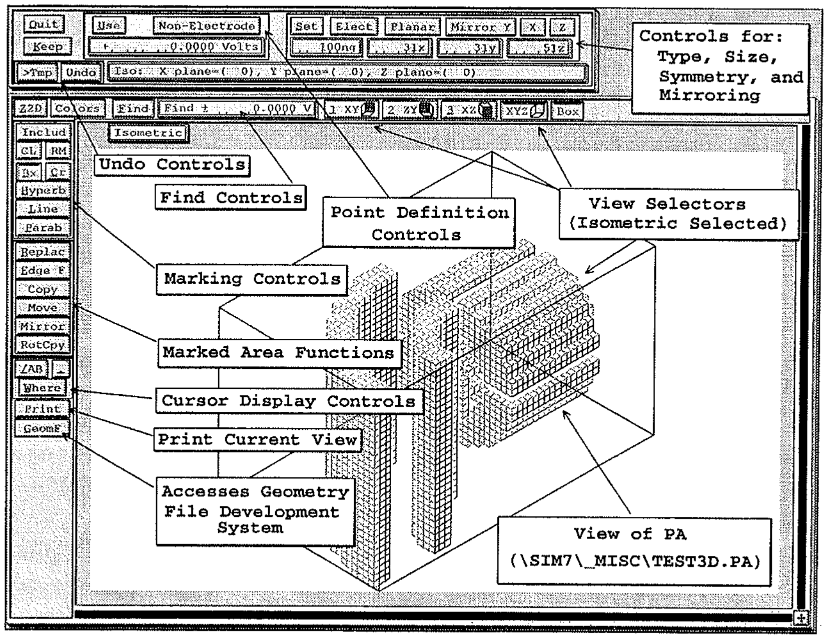


Figure 5-4 Modify function's screen with functional areas highlighted

The Modify Function

SIMION's **Modify** function provides a very direct way to define array geometry via a simple point and click drawing interface (*Figure 5-4*). Features are provided to support quick creation of electrode shapes from rectangles, circles (*or ellipses*), multi-line boundaries, parabolas, and hyperbolas. Editing functions like selective marking, finding, moving, copying, and mirroring help speed the geometry definition process. Moreover, **Modify** can be used to change the type, size, symmetry, and mirroring of potential arrays. It also provides access to the Interactive Geometry File Development System (*an advanced SIMION feature - Appendix J*). **Modify** is most appropriately used for the quick definition of 2 dimensional potential arrays. However, it has tools to support the definition of relatively simple 3 dimensional potential array geometry too (*complex 3D is better handled via geometry files – Appendix J*).

The editing and viewing strategy employed in **Modify** is to view and edit a potential array in terms of planes of points. A 2D potential array is by definition a single plane of points. A 3D potential array is represented by multiple planes of points that can be selectively viewed and edited from the X, Y, or Z directions. Moreover, 3D isometric view capabilities are provided as shown in Figure 5-4 to assist in maintaining ones visual prospective when navigating within a sea of planes.

Entering, Exiting, and Undoing the Modify Function

Entering the Modify Function

The **Modify** function is accessed from the Main Menu Screen by selecting a PA (*depressing its button*) and then clicking the **Modify** button or hitting the **<m>** key. If the selected in-memory potential array has been modified ***without*** having these changes saved to disk (*its button is outlined in red*) SIMION will ask you if you want to save a temporary copy to disk before you enter **Modify**. It is generally a good policy to say ***Yes*** (*array saved to* ***TEMP_0.PA***)**,** because you then have the option of quitting **Modify** (*assuming some editing disaster occurred*) and recovering the unmolested potential array.

Exiting the Modify Function

The **Quit** and **Keep** buttons are used to exit **Modify**. The **Keep** button assumes that you want to keep any changes you made to the potential array while in **Modify**. The **Quit** button (*or the Esc key*) is used to exit **Modify** without keeping any of the changes (*if possible*).

If you elect to quit **Modify**, SIMION will try to recover the array in the following manner. If a temporary copy of the array was saved (***TEMP_0.PA***) prior to entry into **Modify** or via the **>Tmp** button while within **Modify**, it will be reloaded. Otherwise, SIMION will look for a copy of the array (*with the PA's name*) stored in the currently active directory, and reload it if it is found. If no saved array exists in the currently active directory SIMION will warn you that the current changes have been kept because the unchanged array couldn't be found to reload.

Disaster Recovery -- The Temp Save and Undo Buttons

SIMION supports what might be thought of as an ***anticipated*** undo function to help you protect yourself from self-inflicted disasters while within **Modify**. The **>Tmp** button is used to quick save the current PA image to the **TEMP_0.PA** temporary file. The **Undo** button reloads the most recently saved copy of the PA: **TEMP_0.PA** (*>**Tmp** used*) or the currently saved version of PA (*if it exists*). ***Be sure to click the >Tmp button before you do anything really dangerous***. The undo capability is implemented in this manner to provide protection for the cautious without incurring the performance/storage penalties that saving huge amounts of data for an automatic undo would require.

Odds and Ends

Viewing the Current PA Name

Occasionally it may be useful to know the name of the array (*or at least its extension*) that is currently in **Modify**. If you point the cursor to the Modify Status display panel object, (*immediately to the right of the **Undo** button*) the name of the current PA will appear in the display panel. The PA's name will also be displayed when the cursor points to the following buttons: **Quit**, **Keep**, **>Tmp**, and **Undo**.

Stopping and Initiating Modify View Updates

Complex arrays can sometimes cause slow screen updates of **Modify** views. You have the option of hitting the **Esc** key to stop the view updating process. You also can manually initiate a **Modify** view screen update by either moving the cursor into **Modify's** view area and hitting the **Enter** key or by entering either **Ctrl V** or **Alt V** key combinations from anywhere within the client area of SIMION's window (*to redraw the entire client area*).

Changing PA Type, Size, Symmetry, and Mirroring

Controls for viewing and changing potential array type, symmetry, mirroring, and size appear in the upper right corner of the Modify Screen (*Figure 5-4*). These controls normally display the characteristics of the current PA. However, you can change any of these parameters and then click the **Set** button to actually change the potential array.

If you change any of these parameters (*without clicking the **Set** button afterwards*), their original values will automatically be restored when you move the cursor ***out*** of the region. This can accidentally happen when you are trying to point to the **Set** button. ***Thus it is recommended that you change the desired parameters and then hit the <s> key to activate the* Set *function to avoid the chance of accidental resetting.***

Changing Array Type

Modify can be used to switch potential arrays between electrostatic and magnetic types. For example, if an electrostatic the array is to be made magnetic, click the **Elect** button (*the title will switch to* **Magnt**).

When a magnetic potential array is selected access to the **ng** panel will be unblocked. The default value is 100. You can set ng to the number of grid units of pole gap so that the magnetic potentials (*Mags*) are reasonably direct indications of the magnetic field in gauss between the pole pieces (*discussed in Chapter 2*).

Changing Array Symmetry

SIMION allows 2D arrays to have either planar or cylindrical (*surface of revolution*) symmetry. ***All 3D arrays must be planar (SIMION enforced).*** For example, if you want planar click the **Cylind** button (*title will switch to* **Planar**).

Changing Array Mirroring

You have the option of mirroring the potential array for negative x, y, and z values. This can be used to reduce array sizes if the modeled real-world device has the appropriate mirroring symmetries. You can select the desired mirroring. If the combination is illegal, SIMION will beep and enforce a legal mirroring button status. The possible array mirroring combinations are: ***None, X, Y, Z, XY, YZ, XZ, XZY***. The legal mirroring combinations by array category are:

All 3D arrays:	All mirroring combinations are legal
Planar 2D arrays:	All mirroring except z are legal
Cylindrical 2D arrays:	***y mirroring is required***, x is legal, z is illegal

Changing Array Size

The x, y, and z panels are used to specify array dimensions. Note: ***2D*** arrays are specified when *z = **1***. ***3D*** arrays are specified when *z > **1***.

Array Size Bounded by PA's Memory Region

The x, y, and z array dimension panels will not permit array dimensions that would exceed the points allocated to the PA's memory region. Your recourse when this happens is to save the PA, remove all PAs from RAM, and reload the PA into a larger allocated memory region (**Max PA Size** *panel set to needed memory value or above* ***before*** *reloading the PA*).

How SIMION Changes Array Size in Modify

When array dimensions are reduced (*x, y, or z*) within **Modify**, SIMION simply removes the excess points ***and*** whatever geometry they contained from the new reduced size array.

When an array is expanded SIMION ***normally*** inserts zero potential non-electrode/non-pole points for all new points in the expanded array. The exception involves converting 2D arrays to 3D arrays (*discussed immediately below*).

Converting a 2D Planar Array to a 3D Array

A 2D planar array can be converted into a 3D array (*assuming enough memory is allocated to the PA's memory region*) by setting the z dimension to a value greater than one. SIMION will try to help you by asking if you want to copy points from the z = 0 plane into the newly created z planes. ***This duplicates the z = 0 geometry in all the newly created z planes.***

Converting a 2D Cylindrical Array to a 3D Array

A 2D cylindrical array can be converted into a 3D array as with 2D planar above. SIMION will try to help you by asking if you want to ***rotate copy*** points from the z = 0 plane into the newly created z planes. The points are ***normally*** copied into the new z planes assuming a ***surface of revolution*** about the x-axis.

SIMION will ask if you want r boundary electrode/poles extended into the corner regions. ***If you elect this option, r boundary electrode/pole points will be extended to the 3D boundary limits (forming rectangles instead of circles).*** This is normally desirable (*if not visually*) to assure that refining retains more of the infinite r extent assumption for r boundary electrodes/poles that the cylindrical symmetry assumption implied.

Trick: If we have a cylindrical symmetry 2D array of x = 100, y = 20, z = 1 and change the size to x = 100, y = 39, z = 39. SIMION will recognize that it has exactly enough room to convert the 2D array into its 3D equivalent and will ask you if that is what you want (SIMION will also recognize half cylinders too e.g. x = 100, y = 20, z = 39 or x = 100, y = 39, z = 20). This is a very quick way to convert cylindrically symmetrical elements into a 3D array.

Converting a 3D Array to a 2D Array

A 3D array can be converted to a 2D array (*with obvious loss of data*) by setting the z dimension to a value of one. Note: Only the z = 0 layer will remain. ***If something important exists on some other z layer be sure to move or copy it to the z = 0 layer before converting from 3D to 2D.***

Selecting From the Available Array Views

The viewing strategy employed in **Modify** is to view a potential array in terms of planes of points (*see Figure 5-6*). A 2D potential array is by definition a single plane of points. A 3D potential array is represented by multiple planes of points that can be selectively viewed from the X, Y, Z, or isometric directions.

Thus there are four available views: **XY**, **ZY**, **XZ**, and **XYZ** (*isometric*). These views are selected via view selection buttons (*see Figure 5-5*). The first three (**XY**, **ZY**, *and* **XZ**) are 2D views (*planes of points*). The fourth (**XYZ**) is an isometric 3D view. ***Only the* XY *view is allowed with 2D arrays.*** To select a view, click on its button (*if not blocked*).

Layer Displays in 2D Views

The three 2D views (**XY**, **ZY**, *and* **XZ**) display a layer (*or plane*) of array points (*an isometric view of layering is shown in Figure 5-6*). ***In the case of 2D arrays, the view is always the z = 0 layer of the* XY *view.*** This means when you are viewing a 3D array in the **XY** view you are looking at a specific z layer or plane of points (**x** *layer in* **ZY** *view and* **y** *layer in* **XZ** *view*). In Figure 5-5, the view is of the **XY** view of the z = 22 layer. This example uses the **\SIM7_MISC\TEST3D.PA** potential array. ***You might want to load this 3D array yourself and follow along with the manual's examples.***

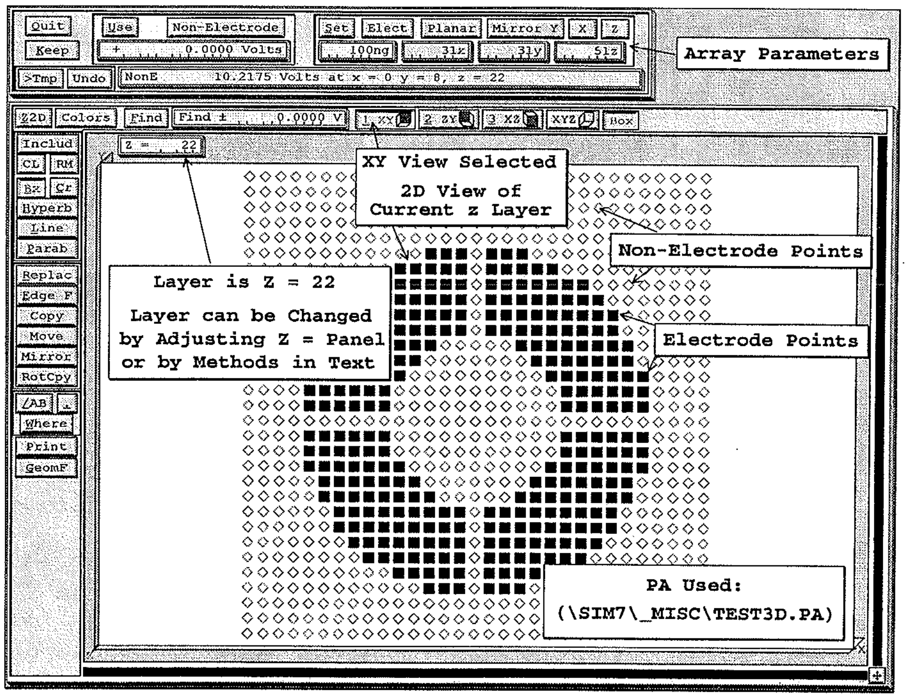


Figure 5-5 Display of XY view of layer 22 in a 3D array

Changing the Displayed Layer in a 2D View

There are many occasions you will want to look at the layers of a 3D array with one or more 2D views. SIMION provides several methods for selecting the layer to display. ***The recommended way is to use 3D layered view layer switching*** (*discussed below*):

Using the Layer Panel

The most direct method involves using the layer panel itself (*upper left corner of 2D view screen*). The value of the desired panel can be entered directly into the panel or the mouse buttons can be used to click the desired digit up or down (***right** mouse button for **up**, **left** mouse button for **down***).

Using the Window Button

The window button can (*lower right corner*) be used in two ways to change the displayed layer:

Normal View Layer Switching

This method changes the layers of the normal 2D view as you move the mouse.

1. Hold down the ***left*** mouse button while on the window button (*lower right corner of window*).
2. Also hold down the **<Alt>** key.
3. Move the mouse about and the window's layer will change. ***Holding down the right mouse button <u>too</u> will speed up re-displays.***

3D Layered View Layer Switching

This method changes the layers by using a stacked 3D isometric view as you move the mouse. Each layer appears in its location (*much like in a deck of cards*). The screen drawing is quite fast and you get a better feeling about where you are. ***This is the recommended way to change layers.***

1. Hold down the ***right*** mouse button while on the window button (*lower right corner of window*).
2. Also hold down the **<Alt>** key.
3. Move the mouse about and the window's layer will change. ***Holding down the left mouse button too will speed up re-displays.***
4. Release the mouse button(s).

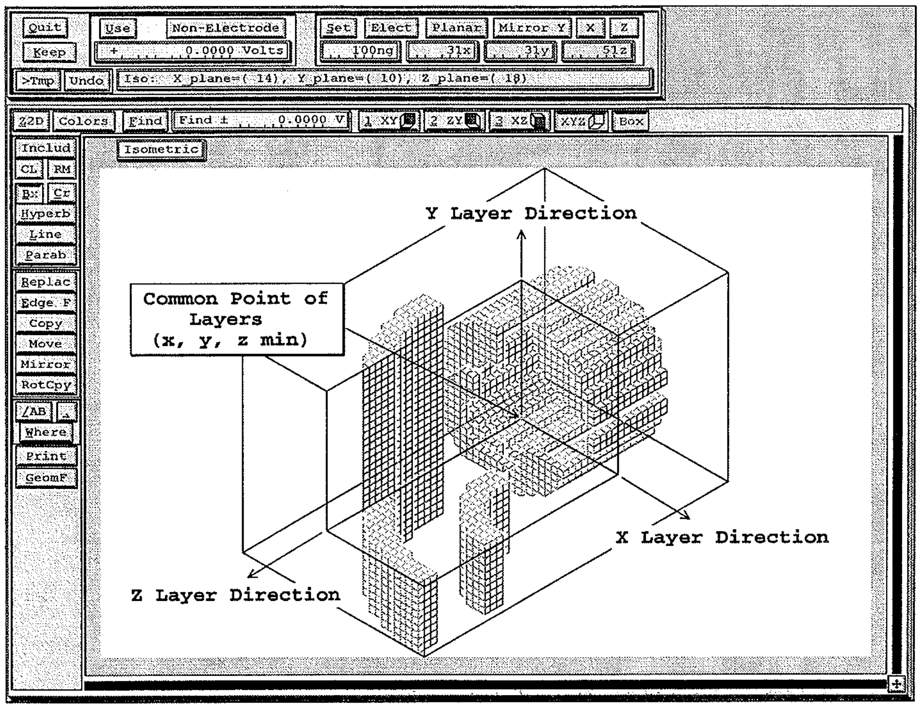


Figure 5-6 Isometric view showing interaction between the three layers: XY, ZY, and XZ

Lower Left Corner Connects All Three Layer Views

When you adjust layers and switch between views SIMION tries to keep the lower left corner point (*min. x, y, and z*) as the min. in all 2D layer views (*Figure 5-6*). This can result in apparent 2D zooming (*reduced view size*). ***You can always restore the full view of the layer by clicking the right mouse button while the cursor is in the view area*** (*or clicking the **Z2D** button or hitting the <z> key*).

Changing Layers From an Isometric View

SIMION also supports changing all three layers (*x, y, and z*) from within an isometric view (***XYZ** button depressed*) via the notion of 3D isometric pointing (*Figure 5-6*). ***This is also useful for cutaway views of 3D arrays:***

Normal View Layer Switching

This method changes the layers of an ***isometric*** view as you move the mouse (**XYZ** *button depressed*).

1. Hold down the ***left*** mouse button while on the window button (*lower right corner of window*).
2. Also hold down the **<Alt>** key too.
3. Move the mouse in the desired isometric direction(*s*) and the x, y or z layer(*s*) will change. ***Holding down the right mouse button too will speed up re-displays.***

Page View Layer Switching

This method changes the layers of an isometric view as you move the mouse (**XYZ** *button depressed*). This method makes use of page view drawing (*faster*).

1. Hold down the ***right*** mouse button while on the window button (*lower right corner of window*).
2. Also hold down the **<Alt>** key.
3. Move the mouse in the desired isometric direction(*s*) and the x, y or z layer(*s*) will change. ***Holding down the left mouse button too will speed up re-displays.***

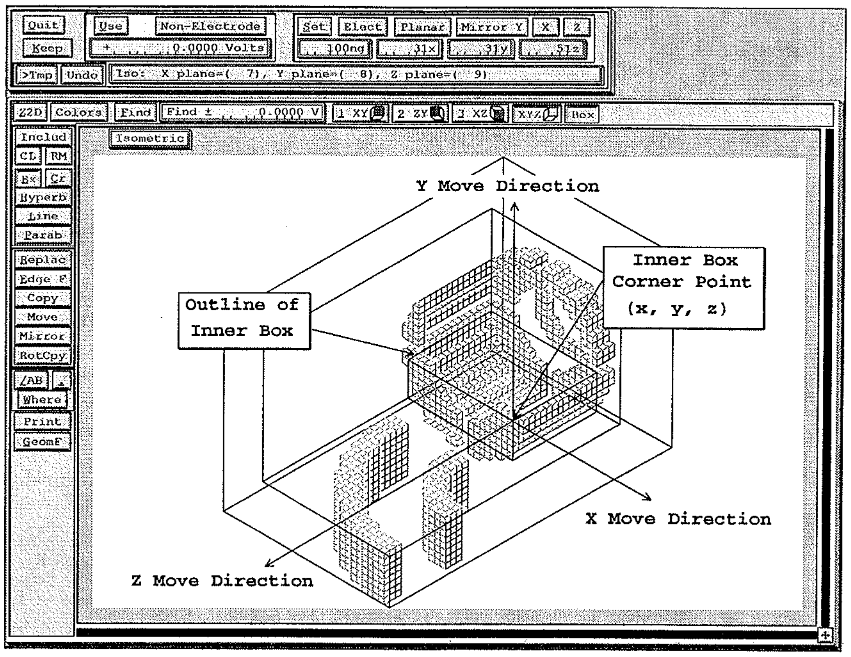


Figure 5-7 Creating interior volumes in an isometric view by isometric 3D mouse pointing

Showing Interior Volumes From an Isometric View

The ability to create cutaway views as shown in Figure 5-6 above can be very useful. SIMION also adds the ability to define an inner box to be viewed within this cutaway (*Figure 5-7*). This can be very helpful to see geometry within the inner box that would have been blocked by the outer cutaway region.

1. Create cutaway view using methods described above.
2. Move cursor into view area.

3. Hold **<Alt>** key depressed.
4. Depress ***either*** mouse button and drag mouse in isometric directions to define the inner volume (*box*). ***Holding down the other mouse button too will speed up re-displays***.

Zooming and Scrolling within the Current View

Zooming the Viewed Area

There are several ways to zoom in and out in the currently selected view (*whether 2D or isometric*):

Zooming Via Area Marking

The easiest way to zoom in is to mark the desired area (*by box, circle, or line marking described below*), and click the ***right*** mouse button (*any prior marks will be restored after zooming*). Ten levels of zoom are remembered. To zoom out one level just click the ***right*** mouse button ***without*** making a mark first. To zoom back in a level hold a **shift** key depressed and click the ***right*** mouse button without making a mark first (*prior marks are conserved*).

Zooming in Page View

You can also zoom using the window button in page view. If you hold down the ***right*** mouse button while the cursor is on the window's button (*page view*) you will see a blue rectangular outline around the currently displayed area. If you hold down the **Ctrl** key while keeping the ***right*** mouse button depressed and move the mouse you can change the size of this rectangular outline (*zooming is fast*).

Zooming in Normal View

You can zoom using the window button in normal view. Hold down the ***left*** mouse button while the cursor is on the window's button (*normal view*). If you hold down the **Ctrl** key while keeping the ***left*** mouse button depressed and move the mouse you can change the area viewed (*zooming is slower than page view*).

Scrolling within the Current View

The current view can be scrolled (*horizontally shifted*) by either using the window's button or by what is called demand scrolling.

Scrolling the View with the Window's Button

To scroll a zoomed view horizontally or vertically move the cursor to the window's button. Hold down the ***left*** (*normal view*) or ***right*** (*page view - faster*) mouse button. Now move the mouse in the desired direction (*keeping the mouse button depressed*) and the view will scroll (*horizontal and vertical 2D scrolling supported*).

Demand Scrolling the View

You can also demand scroll when the cursor is in the view area. Hold the **Ctrl** key depressed and move the cursor. When you try to push the cursor outside the view area, the view will automatically scroll in that direction.

Displaying the Cursor Coordinates

Cursor coordinates are displayed on the Modify Status Display ioline and optionally with the screen cursor itself. The coordinate values displayed are either absolute array grid unit coordinates (*the default*) or relative array grid unit coordinates (*optional*) to some reference array point.

Viewing Cursor Array Coordinates on the Status Ioline

Cursor Coordinates (*2D Views only*) are ***always*** displayed on the Modify Status Display ioline (*just to the right of the **Undo** button*). Move the cursor to the desired location and ***pause*** a moment and the status ioline will be automatically updated to display the current cursor location.

Displaying the Cursor Array Coordinates at the Cursor

If you depress the **Where** button (*Figure 5-7 near bottom of buttons on left*) or hit the <w> key the cursor will the display the cursor's coordinates near the crosshairs cursor's intersection point (*2D views only*). ***This is handy for making sure that you are pointing at the desired location when marking volumes.*** The status of the **Where** button is ***remembered*** between **Modify** sessions.

Selecting Absolute or Relative Cursor Coordinates

By default, absolute cursor array coordinates are displayed. Absolute array coordinates assume that the origin is the lower left corner of the potential array ($x = 0, y = 0, z = 0$). You also have the option of designating the coordinates of an array point that is to be used for relative coordinates. The two buttons immediately above the **Where** button are used to control these features.

The /AB /RL Button

The **/AB /RL** button is used to toggle the coordinates displayed between **ab**solute (*/AB*) and **rel**ative (*/RL*). The </> key can be used to toggle this button from the keyboard. ***A construction line connects the cursor to the reference point when relative positions are active*** (*Whenever the button is depressed - /RL is displayed*).

The . Button

The **'.'** button is used to record/change the reference point for relative positions. Use the mouse to mark the location of a reference point (*point to the desired location and click the left mouse button*). Now click the **'.'** button or hit the **'.'** key. A screen will appear with the coordinates of the marked reference point. You can edit these coordinates as desired and then accept them. If no point is marked when the **'.'** button is clicked or **'.'** key is pressed, the current reference point's screen will be displayed for you to view and/or edit.

An Introduction to Marking Areas and Volumes in Modify

Generally you must first mark an area or volume in **Modify** before you can access functions like **Replace** or **Move**. Thus, the ability to mark a region of points properly is crucial to using **Modify**. SIMION treats marks made in 3D isometric views differently from marks made in 2D views. Marks in isometric views are ***only*** used for zooming. While marks in 2D views are used ***either*** for zooming ***or*** area/volume marking.

Marking a Region in an Isometric 3D View

SIMION uses the simplest notion of marking for 3D isometric views (***XYZ** view button depressed*). Marks on an isometric view only support the zooming function. To mark a rectangular region for zooming move the cursor to one corner of the desired region and then drag the cursor to the diagonally opposite corner with the ***left*** mouse button depressed. SIMION will draw a rectangle around the marked screen area. Now click the ***right*** mouse button to zoom into the marked region (*or click the **Z2D** button or hit the <z> key*).

Marking an Area or Volume in 2D View(s)

SIMION's notion of marking changes when you mark from within a 2D view (**XY**, **ZY**, *or* **XZ**). Marks in 2D views are used to define areas/volumes as well as zoom areas. We will focus our discussion on how areas/volumes are marked.

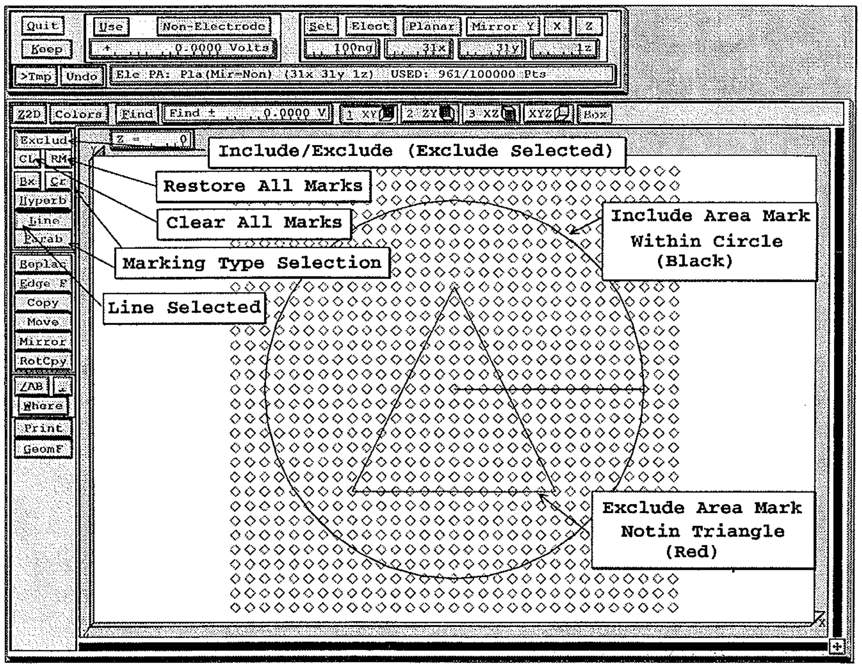

Include/Exclude (Exclude Selected)
Restore All Marks
Clear All Marks
Marking Type Selection
Line Selected
Include Area Mark Within Circle (Black)
Exclude Area Mark Notin Triangle (Red)

Figure 5-8 Example of using include (a circle) and exclude (a triangle) area marking

The Strategy for Marking Volumes

Whenever you mark an area within any layer in a 2D view, you are actually marking all layers with the same mark. ***Thus an area mark is <u>really</u> a volume mark.*** For example, if a circle is marked in any layer in an **XY** view, SIMION assumes a cylinder has been marked (*for a 3D array*).

Constraining the Marked Volume Further

In most cases we do not want to mark volumes that span all layers of the view. SIMION solves this problem by allowing you to mark areas (*volumes*) in the other 2D views to further constrain the marked volume. Thus if we switched to the **ZY** view and marked a rectangle that intersected the cylinder (*above*), SIMION would automatically limit the marked volume to the intersection volume of the two marked volumes (*a cylindrical disk*).

This means that the marked volume is always the volume of intersection of area marks in one <u>to</u> three of the 2D views (XY, ZY, and/or XZ). If you define area marked volumes that don't have a volume of intersection SIMION will object when you try to do something (*e.g.* **Move** *points*).

Types of Marks Supported

Modify allows you to mark areas with: Rectangles, circles (*or ellipses*), connected line segments, parabolas, or hyperbolas. You have the option of using a different type of mark for each area marked. This provides considerable flexibility in defining the shape of marked volumes (*Figure 5-8*).

The Notion of Include and Exclude Volumes

To further complicate your life, SIMION allows both include (*within*) and exclude (*notin*) volume marking. Thus you have the option of defining up to six area marks (*an include and exclude mark in each of the three 2D views*).

SIMION will consider a point to be within the marked volume if it is within the intersection volume of the include marks and notin the intersection volume of the exclude marks. Note: If you define include area marks that do not have a volume of intersection or exclude marks that do not have a volume of intersection, SIMION will object when you try to do something (*e.g.* **Replace** *points*).

The Modify Screen has a button in the marking control area to control whether a mark is considered to be include (*within*) or exclude (*notin*). ***Click the button to the desired marking type before starting the marking process*** (*Figure 5-8*).

In the example above, we could have also marked a triangular ***exclude*** mark within the circle mark in the **XY** view to create a circular disk with a triangular hole in it (*Figure 5-8 above*)..

How to Select and Use the Various Area Marks

The recommended steps when creating a 2D view mark are to switch to the desired view, select include or exclude, select the type of mark (*e.g.* **Box**), and draw the mark on the view area. The following gives specific instructions about each marking type:

How to Use Box Marking

To mark a rectangle make sure the* Box *button is depressed (*Figure 5-9*). Now move the cursor to a corner of the box. Hold the ***left*** mouse button depressed and drag the cursor to the diagonally opposite box corner. Now release the ***left*** mouse button.

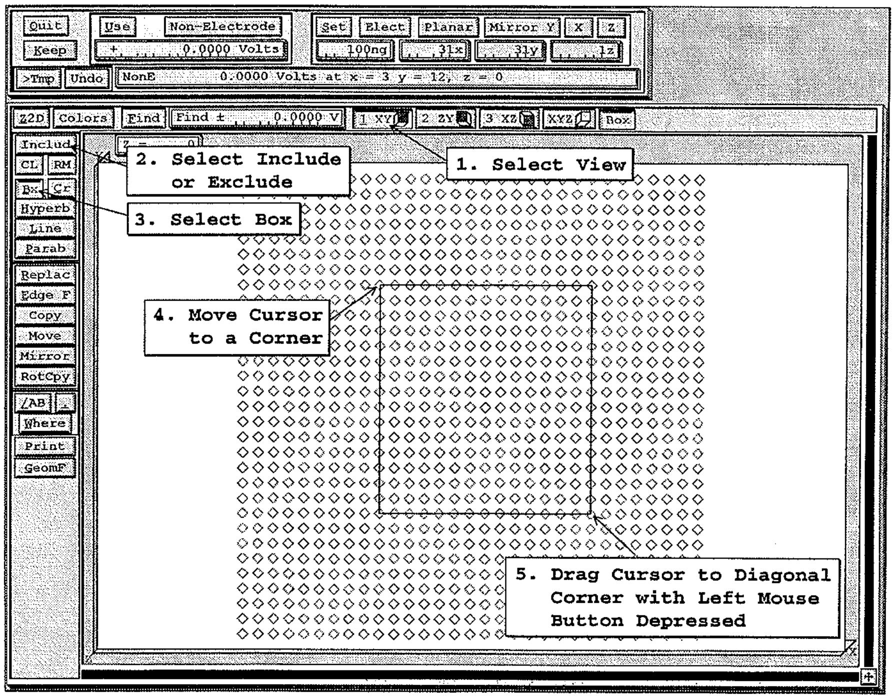


Figure 5-9 Steps involved in using box marking

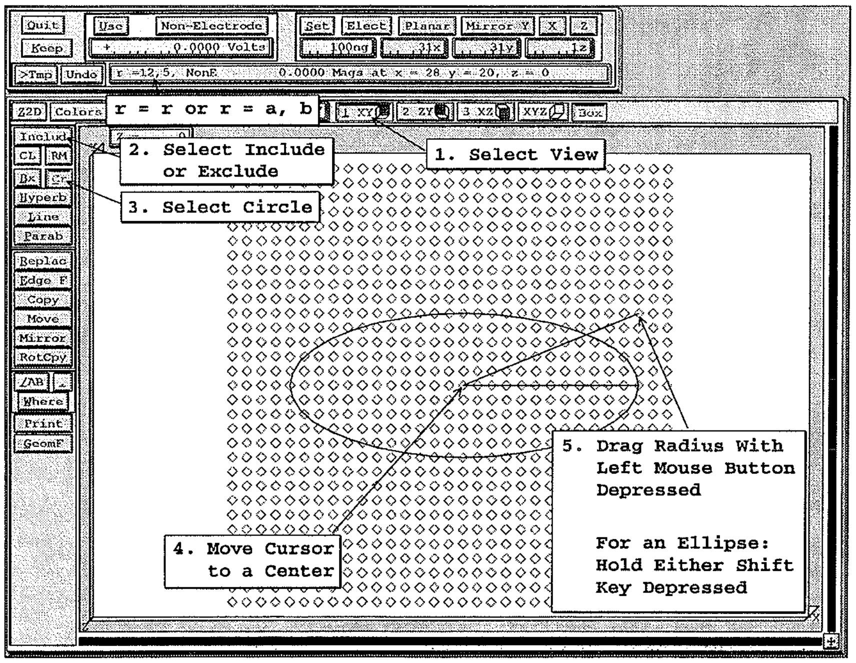


Figure 5-10 Steps involved in using circle marking

How to Use Circle (or Ellipse) Marking

To mark a circle make sure the* Circle *button is depressed (*Figure 5-10*). Now move the cursor to the center of the desire circle. Hold the ***left*** mouse button depressed and drag the cursor to the desired circle radius (*the status line will display the radius value when you pause*). Now release the ***left*** mouse button.

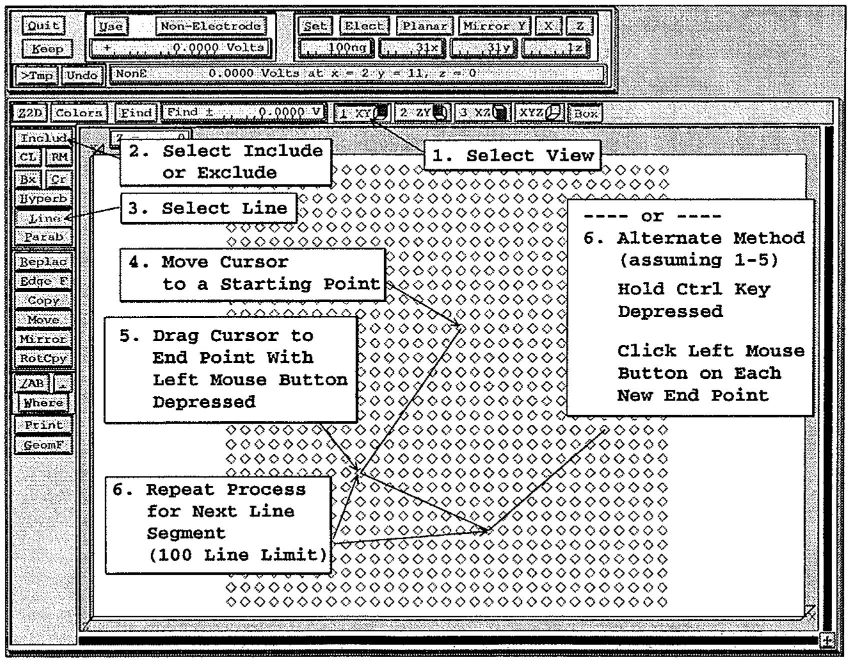

Quit
Keep
>Tmp
Undo
Use
Non-Electrode
0.0000 Volts
Set
Elect
Planar
Mirror Y
X
Z
100ng
31x
31y
1z
NonE 0.0000 Volts at x = 2 y = 11, z = 0
Z2D
Colors
Find
Find ± 0.0000 V
1 XY
2 ZY
3 XZ
XYZ
Box
Includ
CL
RM
Bx
Cr
Hyperb
Line
Parab
Replac
Edge F
Copy
Move
Mirror
RotCpy
Where
Print
GeomF
1. Select View
2. Select Include or Exclude
3. Select Line
4. Move Cursor to a Starting Point
5. Drag Cursor to End Point With Left Mouse Button Depressed
6. Repeat Process for Next Line Segment (100 Line Limit)
---- or ----
6. Alternate Method (assuming 1-5)
Hold Ctrl Key Depressed
Click Left Mouse Button on Each New End Point

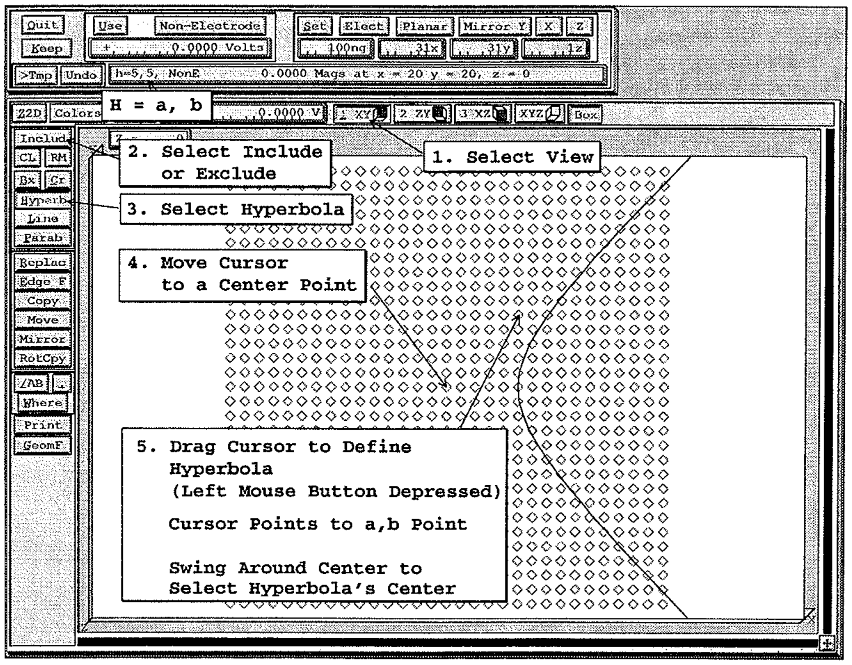
Figure 5-11 Steps involved in using line marking

To mark an ellipse make sure the **Circle** ***button is depressed*** (*Figure 5-10*). Now move the cursor to the center of the desired ellipse. Hold the ***either*** **Shift** key ***and*** the ***left*** mouse button depressed and drag the cursor to the desired ellipse shape (*the status line will display the r = a, b radius values when you pause*). Now release the **Shift** key and ***left*** mouse button.

How to Use Line Segment Marking

To mark by line segments make sure the **Line** ***button is depressed*** (*Figure 5-11*). Now move the cursor to the beginning of the first line segment. Hold the ***left*** mouse button depressed and drag the cursor to the end of the first line segment. Now release the ***left*** mouse button. To add a line segment move the cursor to the end of the most recently defined line segment and drag an additional line segment as above. ***Up to 100 connected line segments can be used***. SIMION will automatically add a closing line segment to areas marked by line segments (*last point same as first point*) if required. ***Hollow regions can be made by drawing them linked with the outer line boundary***.

The shortcut to drawing lines is to move the cursor to the beginning of the first line and click the ***left*** mouse button. Now hold the **Ctrl** key depressed. Move to each successive line connected point and click the ***left*** mouse button. Release the **Ctrl** key when you're done.



1. Select View

2. Select Include or Exclude

3. Select Hyperbola

4. Move Cursor to a Center Point

5. Drag Cursor to Define Hyperbola (Left Mouse Button Depressed)
Cursor Points to a,b Point
Swing Around Center to Select Hyperbola's Center

Figure 5-12 Steps involved in using hyperbola marking

How to Use Hyperbola Marking

To mark with a hyperbola make sure the **Hyperb** ***button is depressed*** (*Figure 5-12*). Now position the cursor at the hyperbola center point and drag the desired hyperbola (***left*** *mouse button depressed*). ***Note: Swing the cursor around the center point to select the desired hyperbola direction***. The following equations are used:

$$y^2/b^2 - x^2/a^2 = 1$$
$$x^2/b^2 - y^2/a^2 = 1$$

Moving the cursor about changes the values of a and b. The status line displays these values (*e.g. h = a,b*).

How to Use Parabola Marking

To mark with a parabola make sure the* Parab *button is depressed (*Figure 5-13*). Now position the cursor at the parabola vertex point and drag to the focus of the desired parabola (**left** *mouse button depressed*). ***Note: Swing the cursor around the center point to select the desired parabola direction.*** The following equations are used:

$$\mathbf{y = x^2/(4p)}$$
$$\mathbf{x = y^2/(4p)}$$

Moving the cursor about changes the values of p (*focus offset*). The status line displays this value (*e.g. p = p*).

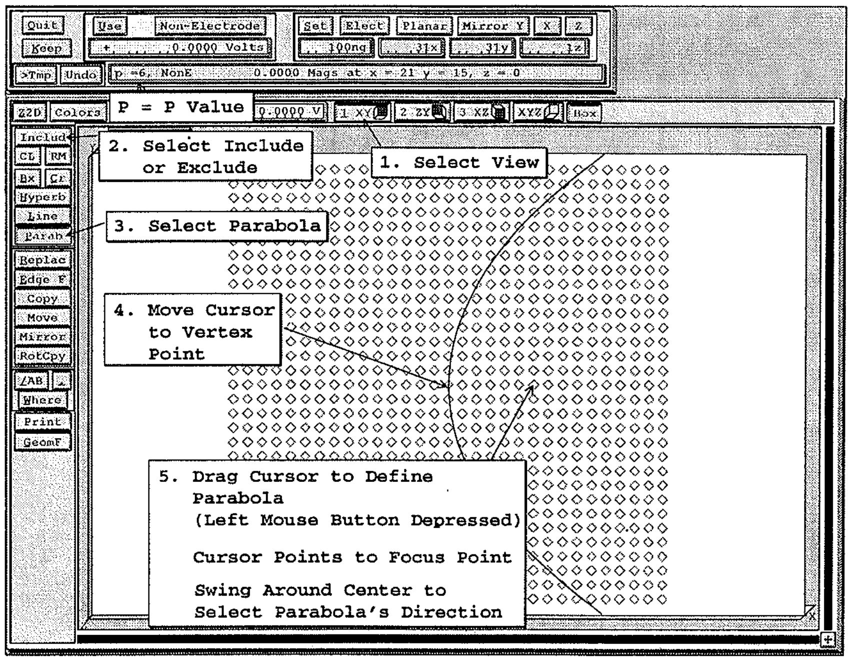

Quit
Keep
>Tmp
Undo
Use
Non-Electrode
0.0000 Volts
Set
Elect
Planar
Mirror Y
X
Z
p =6, NonE 0.0000 Mags at x = 21 y = 15, z = 0
Z2D
Colors
P = P Value
1 XY
2 ZY
3 XZ
XYZ
Box
Includ
CL
RM
Bx
Cr
Hyperb
Line
Parab
Replac
Edge F
Copy
Move
Mirror
RotCpy
Where
Print
GeomF
1. Select View
2. Select Include or Exclude
3. Select Parabola
4. Move Cursor to Vertex Point
5. Drag Cursor to Define Parabola (Left Mouse Button Depressed)
Cursor Points to Focus Point
Swing Around Center to Select Parabola's Direction

Figure 5-13 Steps involved in using parabola marking

Clearing and Restoring Marks

Clearing All Marks

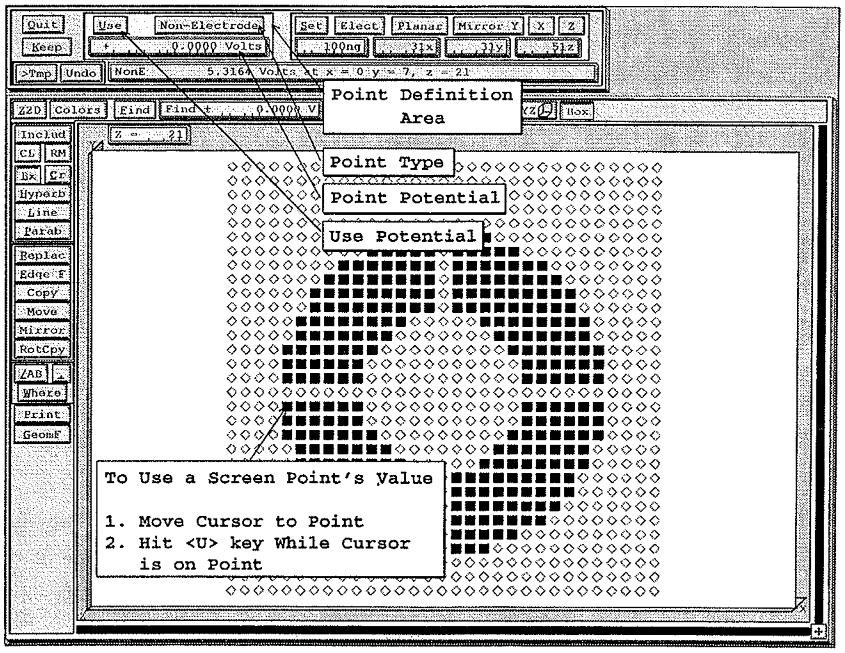
To clear ***all*** marks click the **CL** button (*just below the* **Include/Exclude** *button*). A copy of the current marks (*if any*) will be saved in mark memory (*Figure 5-8*).

Clearing a Specific Mark

A specific mark (*include or exclude*) in any 2D view can be deleted by switching to the desired 2D view, selecting the desired include/exclude status (*with the* **Include/Exclude** *button*), moving the cursor into the view object, and pressing the **Delete** key.

Restoring Marks From Mark Memory

Functions like **Replace**, **Move**, **Copy**, and the **CL** button automatically save any current marks in the mark memory and then clear all marks. If you want to restore these cleared marks click the **RM** (*Restore Marks*) button (*just below the* **Include/Exclude** *button*).



```
To Use a Screen Point's Value

1. Move Cursor to Point
2. Hit <U> key While Cursor
   is on Point
```

Figure 5-14 Point definition area and its control objects

Point Definition Area

The area immediately to the left of the array definition area is the point definition area (*Figure 5-14*). The point values in the point definition area are used as the replacement values for marked points that are acted on by the **Replace** or **Edge** functions (*when* **Find** *is inactive*) as well as defining the points to search for when the **Find** function is active.

Viewing and Setting the Currently Defined point.

Control objects allow you to view/set the type (*e.g. electrode or non-electrode*) and potential (*e.g. Volts*) of the currently defined point.

The Function of the Use Button

The **Use** button is provided to allow you to quickly transfer the type *and* potential of any point in the view to the currently defined point. ***To transfer the values from an array point in the current view, point to it with the cursor and hit the <U> key.***

Find Function

The **Find** function (*Figure 5-15*) allows finding the currently defined point type and potential (*in the point definition area*) ***plus or minus a delta potential***. The plus or minus delta potential allows more flexibility in point selection.

Moreover, when **Find** is active functions like **Replace**, **Move**, and **Copy** automatically use it to further limit the points selected (*e.g. points moved must be inside include volume, outside exclude volume, and be flagged as found*).

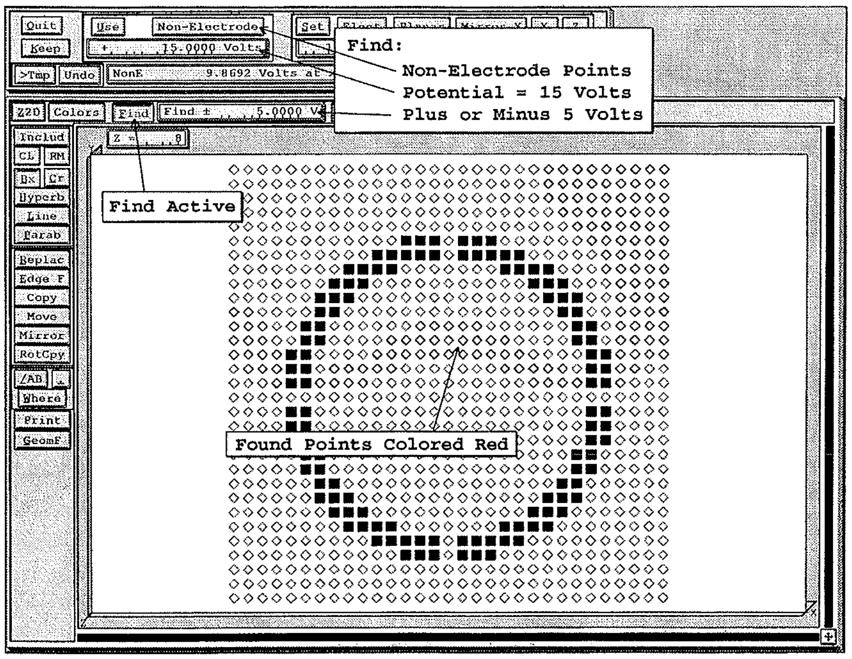


Figure 5-15 Example of using the Find function

Functions Using Marked Volumes

Modify has a collection of functions that operate on marked volumes (*e.g.* **Replace**, **Edge**, **Copy**, **Move**, **Mirror**, *and* **Rotate Copy**). Each of these functions is accessed via a button on the left edge of the Modify Screen (*Figure 5-15*). ***Note: A volume (area for 2D arrays) must be marked <u>before</u> accessing any of these functions.*** A point is selected for processing via these functions if it

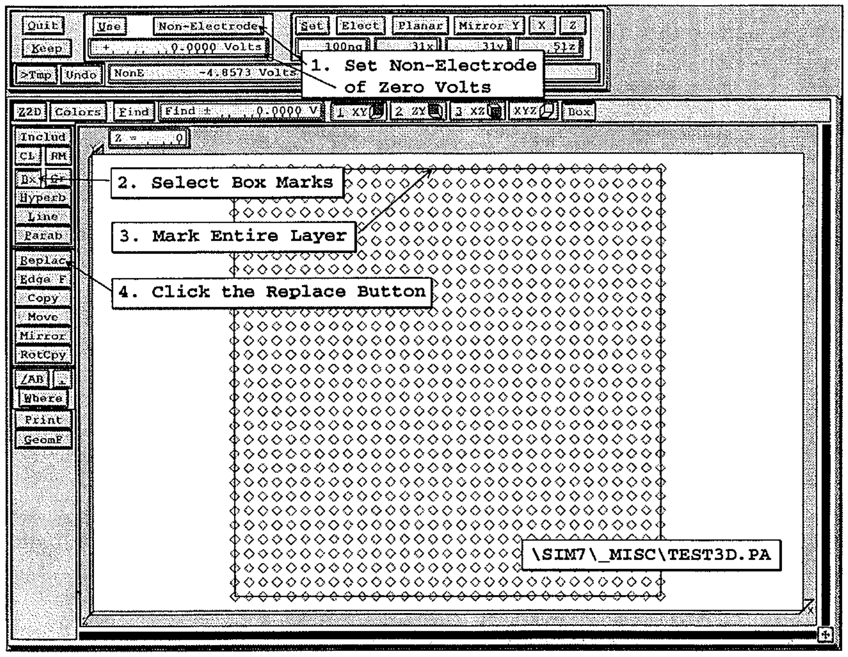

1. Set Non-Electrode of Zero Volts
2. Select Box Marks
3. Mark Entire Layer
4. Click the Replace Button
\SIM7_MISC\TEST3D.PA

Figure 5-16 Using the Replace function to erase the entire array

is within the include volume and notin the exclude volume.

Using Find to Augment Selection

SIMION also allows **Find** (*if active - buttion depressed*) to further augment point selection with all of these functions. When **Find** is active a point is selected for processing via the function if it is within the include volume, notin the exclude volume, and is a found point based on the current find criteria (*point type and within potential range*).

The Replace Function

The **Replace** function replaces the selected points with points of the specified type and potential. There are three basic types of point replacement:

Simple Single Point Replacement

This is the simplest form of **Replace**. ***A volume must NOT be marked (use CL button if necessary to clear all marks), and Find status is ignored.*** Set the point type and potential to the desired ***new*** value. Place the cursor at the point you want to replace, hold the **<Ctrl>** key depressed, and click the ***right*** Mouse button. This ***only*** changes the point under the cursor in the ***current*** layer (*not all layers*). It is very useful for quick point editing on a layer by layer basis. ***Find, if active, is ignored.***

Point Replacement (*Find NOT Active*)

This is probably the most frequently used method of point replacement. The steps are: Mark a volume; set the point type and potential to the desired new value; and click the **Replace** button, hit the **<R>** key, or **<Ctrl>** ***right*** mouse button. SIMION will ask you if you're sure before replacing any points.

The Replace function can be used to erase all points in the potential array. The steps are: Mark an entire 2D layer with a box mark; set the point type to non-electrode/non-pole and potential to zero; and click the **Replace** button, hit the **<R>** key, or **<Ctrl> right** mouse button. Figure 5-16 (*above*) demonstrates this process on a memory copy of

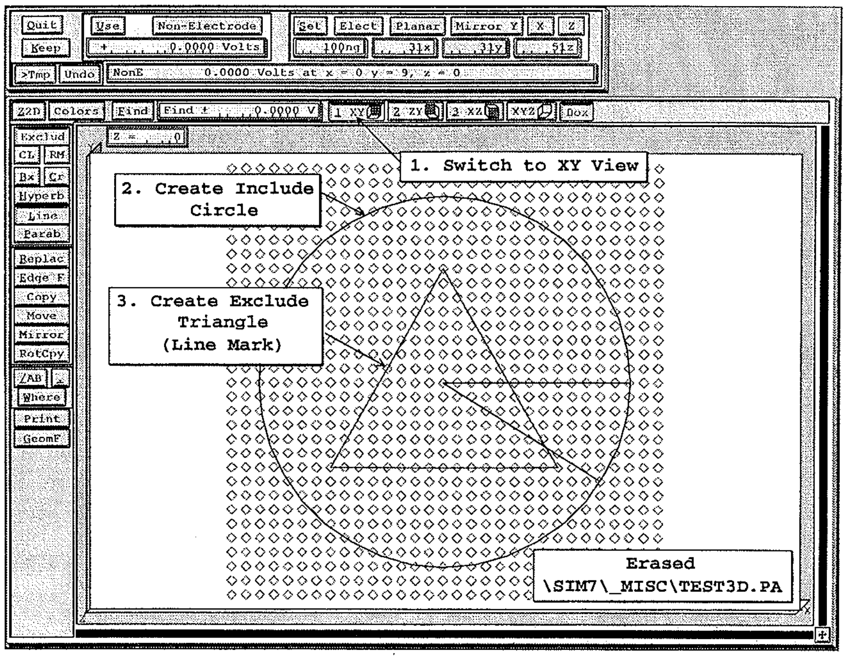


Figure 5-17 First three steps in creating a 3D disk with hole in it

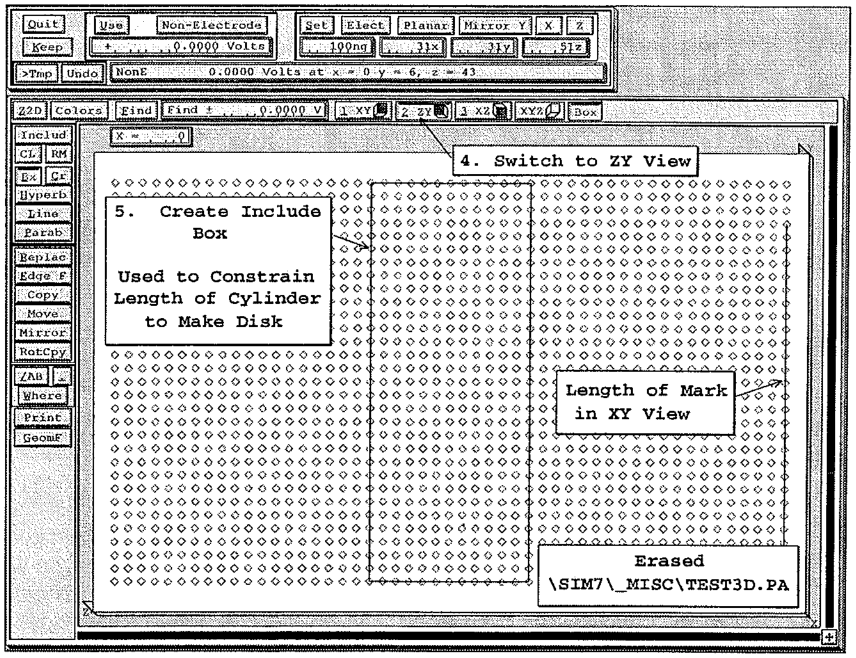


Figure 5-18 Next two steps mark disk length in alternate 2D view

\SIM7_MISC\TEST3D.PA (*you might want to try this yourself*).

Another example of the Replace function involves creating a cylindrical disk with a triangular hole in it. Figures 5-17 through 5-19 demonstrate how this is done. This

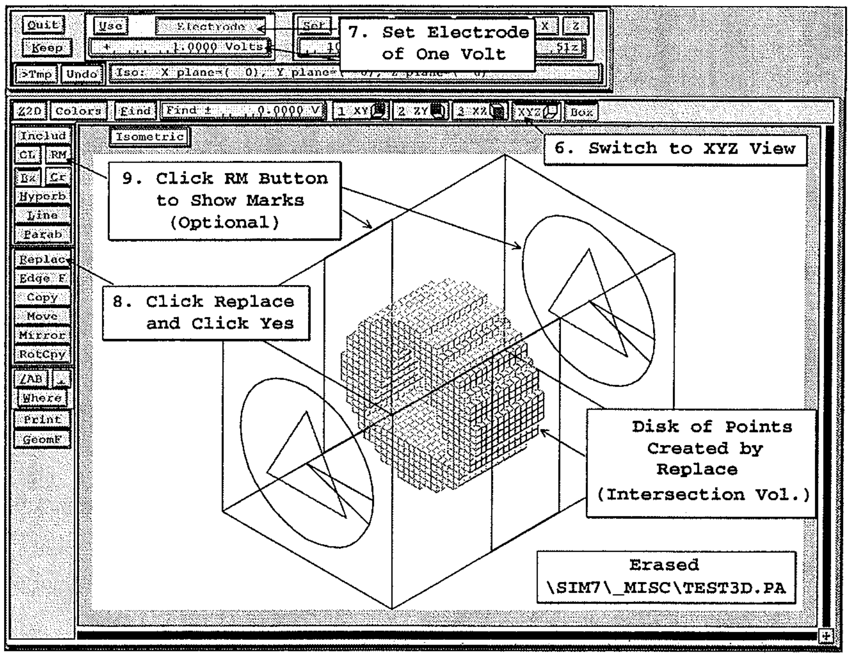

Quit
Keep
>Tmp
Undo
Use
Electrode
+ 1.0000 Volts
Set
X
Z
51z
Iso: X plane=(0), Y plane=(0), Z plane=(0)
7. Set Electrode of One Volt
Z2D
Colors
Find
Find ± 0.0000 V
1 XY
2 ZY
3 XZ
XYZ
Box
Includ
CL
RM
Bx
Cr
Hyperb
Line
Parab
Replac
Edge F
Copy
Move
Mirror
RotCpy
ZAB
Where
Print
GeomF
Isometric
6. Switch to XYZ View
9. Click RM Button to Show Marks (Optional)
8. Click Replace and Click Yes
Disk of Points Created by Replace (Intersection Vol.)
Erased \SIM7_MISC\TEST3D.PA

Figure 5-19 Next four steps to create and view disk and its marks

example starts with the erased PA above (*try to duplicate this yourself*).

Point Replacement (*Find is Active*)

The steps are: Find desired points (*Set the point type, potential, and range. Click* **Find**

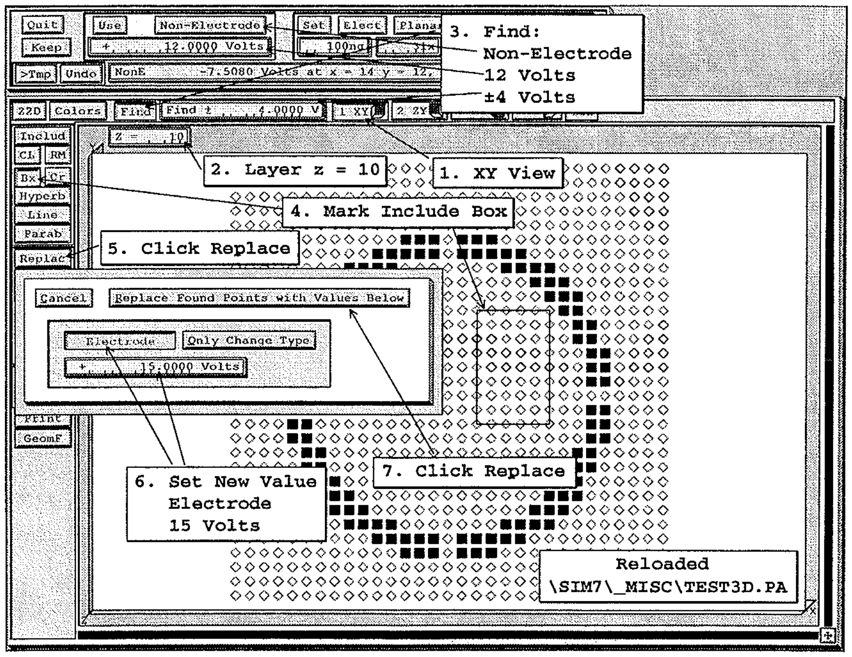


Figure 5-20 Example of using Find to replace selected points

on); mark a volume; and click the **Replace** button, hit the **<R>** key, or **<Ctrl> right** mouse button. SIMION will display a new point definition screen. Select what you want the selected points to be and click the **Replace Found Points** button on this screen.

Note: You have the option of changing the selected points' type and potential or just changing the selected points' type. The latter is useful for creating fixed boundary

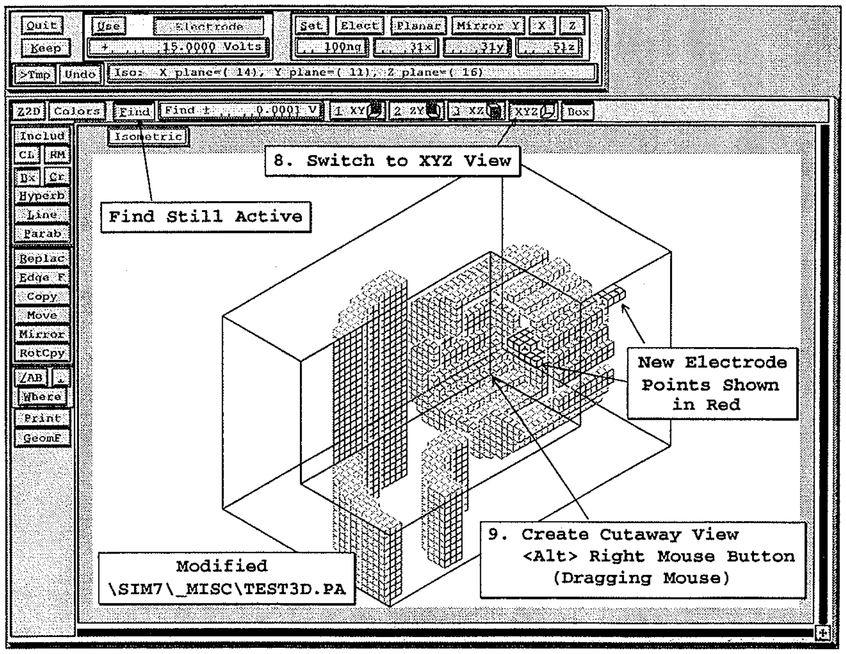

8. Switch to XYZ View
Find Still Active
New Electrode Points Shown in Red
9. Create Cutaway View <Alt> Right Mouse Button (Dragging Mouse)
Modified
\SIM7_MISC\TEST3D.PA

Figure 5-21 Using cutaway view to examine Figure 5-20 results

conditions after refining (*by converting boundary points to electrode/pole*). Figures 5-20 to 5-21 show the creation of an electrode surface from found non-electrode points using a reloaded copy of **\SIM7_MISC\TEST3D.PA** (*try it yourself - good practice*).

The Edge Function

This function works just like the **Replace** function except that ***only*** the ***boundary points*** are ***replaced***. Thus while circular mark would create a solid cylinder, with replace the edge function creates a one point thick cylindrical shell. This function is useful for creating shaped grids.

Note: The **Edge** ***function has no equivalent to*** **Replace's** ***simple single point replace.***

The Copy Function

This function allows you to copy selected ***electrode/pole*** points (*non-electrode/non-pole points are <u>not</u> copied*). You are allowed full include and exclude volume capabilities as well as having **Find** active to further limit point selection. To copy a group of ***electrode*** points make the appropriate volume marks, activate **Find** if desired, and click the **Copy** button. Now draw (*drag with the* ***left*** *mouse button depressed*) a line from the source to the destination to indicate relative location of the copy.

You can click the* RM *button to remark the initial volume if you need to make more than one copy.

The Move Function

This function allows you to move a collection of ***electrode/pole*** points <u>*only*</u> (*as with* **Copy** *above*). You are allowed full include and exclude volume capabilities as well as having **Find** active to further limit point selection. To move a group of ***electrode*** points make the appropriate volume marks, activate **Find** if desired, and click the **Move** button. Now draw

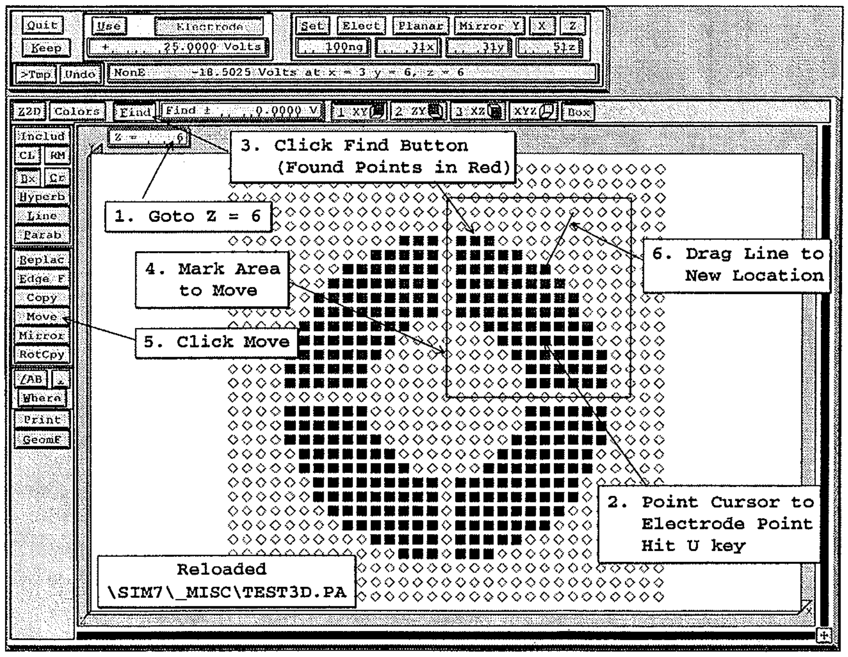
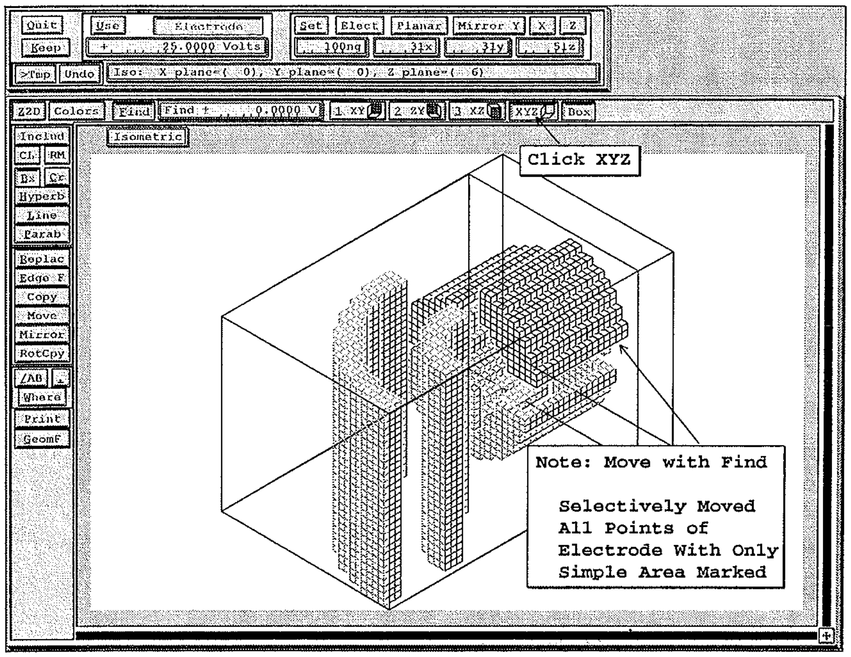


Figure 5-22 Example of moving points with Find active

(*drag with the **left** mouse button depressed*) a line from the source to the destination to indicate relative location of the move. ***Figures 5-22 and 5-23 below, show an example of using*** **Find** ***to selectively move a complex electrode.***



Iso: X plane=(0), Y plane=(0), Z plane=(6)

Note: Move with Find

Selectively Moved
All Points of
Electrode With Only
Simple Area Marked

Figure 5-23 Isometric view of moving points in Figure 5-22

You can click the RM button to remark the volume (*Note: The marks have moved with the electrode points*) ***if you need to make more than one move*** (*e.g. 3D move using another 2D view*).

The Mirror Function

This function allows you to mirror a collection of ***electrode/pole*** points ***only*** (*see Copy above*) across a mirroring line (*plane in 3D*). You are allowed full include and exclude volume capabilities as well as having **Find** active to further limit point selection. To mirror a group of ***electrode*** points make the appropriate volume marks, activate **Find** if desired, and click the **Mirror** button. Now draw (*drag with the **left** mouse button depressed*) a ***horizontal or vertical mirroring line (defines mirroring plane in 3D).***

Note: The mirroring line must be outside the marked volume.

The Rotate Copy Function

This function allows you to create a volume of revolution via a rotating plane copy process. You are allowed full include and exclude volume capabilities as well as having **Find** active to further limit point selection.

To rotate copy a group of ***electrode/pole*** points ***only*** (*see Copy above*), mark an include volume that is a single layer (*one point thick*) ***and extends to the pivot point*** (*of rotation*), activate **Find** if desired, switch to the edge 2D view of the marked volume (*marked region appears as a line*), and click the **RtCpy** button. Now start at the pivot point of revolution and draw (*drag with the **left** mouse button depressed*) the desired arc of revolution (*angle will be displayed on status line*).

The geometry files capability introduced below provides <u>much</u> more powerful rotate copy capabilities.

Geometry Files

SIMION also allows the creation of geometry definitions via an advanced feature called geometry files. The **GeomF** button on the left edge of the Modify Screen is used to access the geometry file development system.

This development system can be used to write geometry instructions in a geometry file (*an ASCII file with a* **.GEM** *extension*). This geometry file can then be compiled with the SIMION's geometry compiler (*accessible from within the geometry development system*) and the resulting geometry definitions inserted in the current potential array.

The process can be highly interactive. The user can erase the potential array from within the development system. Edit the geometry file. Then re-insert the newly re-defined geometry into the array, and immediately examine the results with **Modify**.

Geometry files are most suitable for complex geometry definitions (*2D or 3D*). They are also useful for defining geometry in arrays that may be doubled or halved. The geometry can be re-inserted in a doubled array by doubling the geometry insertion scale factor. This automatically gives the optimum geometry definition for the array's grid density.

See Appendix J for detailed information on the geometry language, defining geometry files, and using the geometry development system.