Refining and Fast Adjusting Arrays - Solving For Fields

Introduction

This chapter covers refining of potential arrays as well as how to make use of SIMION's **Fast Adjust** features. It is assumed that you have read the discussions of potential arrays in Chapter 2, 4, and 5. If not, read the material before proceeding further.

Refining Potential Arrays

Refining, in SIMION, means using the potentials of electrode/pole array points to ***estimate*** the potentials of non-electrode/non-pole array points. The refining process makes use of finite difference techniques to solve the Laplace equation numerically. ***Appendix E*** (*computational methods*) ***provides specific details of the refining algorithms used in SIMION.***

A Glimpse of How Finite Difference Works

The finite difference technique used in SIMION uses the average potential of the nearest four (*2D arrays*) or six (*3D arrays*) neighbor points as the basis for estimating the potential of each non-electrode/non-pole point in the array. ***Note: Cylindrical symmetry 2D arrays have a radius correction applied to this averaging scheme.*** The entire array is scanned sequentially (*point by point*) and the estimated potential of each non-electrode/non-pole point is updated using the average of its nearest four or six neighbor points.

Each scan through the array is called an iteration. Each successive iteration (*scan through the array*) propagates the potentials of the electrode/pole points further throughout the potential array. This successive iteration process (*refining*) continues until the array comes into an equilibrium where no point in the array changes by more than a certain potential (*0.005 volts/Mags*) in a single iteration (*the convergence objective*).

At this point we proclaim that the convergence objective has been reached and the potentials of all non-electrode/non-pole points have been determined (*estimated*). ***Even if we iterate to a convergence objective approaching zero, there is no reason to believe that the potentials of all non-electrode/non-pole points have been determined precisely.*** The only exception is electrode/pole geometry that results in linear field gradients. All other fields are approximate. The quality of the estimate relates to how much the field curvature warps the grid cells (*distorts them away from being small flat plates*). ***The less warped the grid cells are*** (*warping is reduced by higher grid densities*), ***the better the potential estimates are for a given convergence limit.***

Using Various Tricks to Speed up the Process

Refining (*by successive iterations*) can involve many iterations. Naturally, the objective is to reach the convergence objective in the minimum number of iterations (*shortest time*). Unfortunately, while the approach described above is direct, it is generally pretty slow. SIMION makes use of various tricks (*techniques*) to speed up convergence.

Over-Relaxation

The first trick SIMION uses is over-relaxation. The over-relaxation technique determines how much a point's potential is to be changed in an iteration and multiplies this potential change by a percentage of over-relaxation:

Used_Delta_Potential = Normal_Delta_Potential (1.0 + Over_Relaxation_Factor)

The value of the over-relaxation factor varies from 0.0 (*none*) to 1.0 (*unstable*). The default value of 0.9 used by SIMION appears to work best for most potential arrays.

Over-relaxation works much like an elevator. ***Things work best if there are no sudden starts and stops*** (*everything is smooth and steady*). Thus SIMION starts refining and ends refining by automatically reducing the over-relaxation factor to help reduce solution kinks. The current convergence error term is also used to automatically adjust the over-relaxation factor. To avoid sudden changes in over-relaxation factor, SIMION averages the error term using a exponential moving average technique (*smoothes out changes in over-relaxation factor*). An historical memory factor is used to define the amount of exponential averaging (*default value is 0.7*).

Skipped Point Refining

The use of over-relaxation can reduce the number of required iterations by factors of 10 or more. ***However, even with over-relaxation, the number of iterations required to refine an array is generally proportional to n^2 where n is the number of points in the potential array***. This can make refining large arrays of a million or more points take hours, days, or more on a 486 class PC.

The skipped point refining technique, developed by the author, yields refine times roughly proportional to n (*an enormous improvement for large arrays*). The technique involves dividing the array dimensions by powers of two into an initially small array (*containing perhaps one in every 16 points in each dimension*). This array is refined. The array is expanded by two. The new array points are estimated using points from the last refine. The array is refined again. This process continues until the array is finally refined without skipping any array points.

The problem with this approach occurs when electrode/pole points are skipped (*unseen*), their effects will have to be propagated when they at last become visible. ***This delay in potential propagation can slow refining later, destroying all the time savings***. SIMION solves this problem by looking around for skipped electrode/pole points at every level of skipped point refining. If a skipped electrode/pole point is found, a flag is set and SIMION automatically takes the effects of the electrode/pole point into account even though it is not yet visible. The bookkeeping is complex, but the refining is very fast.

Note: Skipped point refining will refined square or cubic arrays the fastest, because there is the most opportunity to skip the most point. ***Conversely a long narrow array (e.g. 1000 x by 10 y) will refine at non-skipped rates, because little if any point skipping is possible***.

How SIMION Handles Array Boundaries

The refining process uses the average of the nearest four (*2D*) or six (*3D*) neighbor points to estimate the new potential for each non-electrode/non-pole point in the array. All this works just fine for points within the interior of the array. ***The problem is how to handle points on the array boundary***.

Points on Non-Mirrored Boundaries

The points on non-mirrored boundaries are handled according to their location:

Corner Points

Corner points use the average of the nearest ***two*** (*2D*) or ***three*** (*3D*) neighbor points to estimate their potentials.

Edge Line Points

Points on an edge line use the average of the nearest ***three*** (*2D*) or ***four*** (*3D*) neighbor points to estimate their potentials.

Edge Plane Points (3D only)

Points on an edge plane use the average of the nearest ***five*** neighbor points to estimate their potentials.

Points on Mirrored Boundaries

The points on mirrored boundaries have the appropriate mirrored points also added into the averaging process. For example, in the case of points along the interior x-axis in a 2D planar array with y mirroring, the average would involve ***four*** points: ***The three neighbor array points plus the point above in y***. Thus the ***plus one*** y point is assumed to also exist as a ***negative one*** y point (*y mirrored*).

This thought process can be extended to logically determine how mirrored averaging works for any point along a mirrored boundary.

What Does All This Mean?

The result of this approach is that SIMION extrapolates at the array boundary. ***If you do not bound the array boundary with electrode/pole points SIMION will struggle on***. Whether the result is what you intended or not depends on how you defined the points near and on the array boundary.

When Electrode/Pole Shapes Touch the Boundary

When electrode/pole shapes actually touch the array's boundary, SIMION assumes (*using the above averaging scheme*) ***that the electrode/pole extends to infinity***. If this is not a valid approximating assumption, there is a good chance that the refined fields may not be particularly valid either.

When Electrode/Pole Shapes Do Not Touch the Boundary

When electrode/pole shapes do not actually touch the array's boundary, SIMION assumes (*using the above averaging scheme*) that the electrode/pole shapes are bounded. Be sure to leave enough space (*non-electrode/non-pole*) to the boundary so that field propagation around the boundary can be approximated.

Remembering the Ground Can

Most optics devices live in grounded vacuum chambers. ***You ignore their existence at your peril.*** I have witnessed many examples of *"it doesn't model my problem properly"* that were the result of not including the ground can in the potential array. ***The better your problem is bounded the better your results are likely to be!***

Comparing Skipped Point Verses Non-Skipped Point Refining

It is useful to compare the results of refining with or without skipped point refining active. Figures 6-1 and 6-2 are potential energy surface views of a simple square 2D planar array (*301x by 301y*) refined by each technique. The boundary electrode points were set to 1,000 volts and the interior non-electrode points were set to zero (*0*) volts.

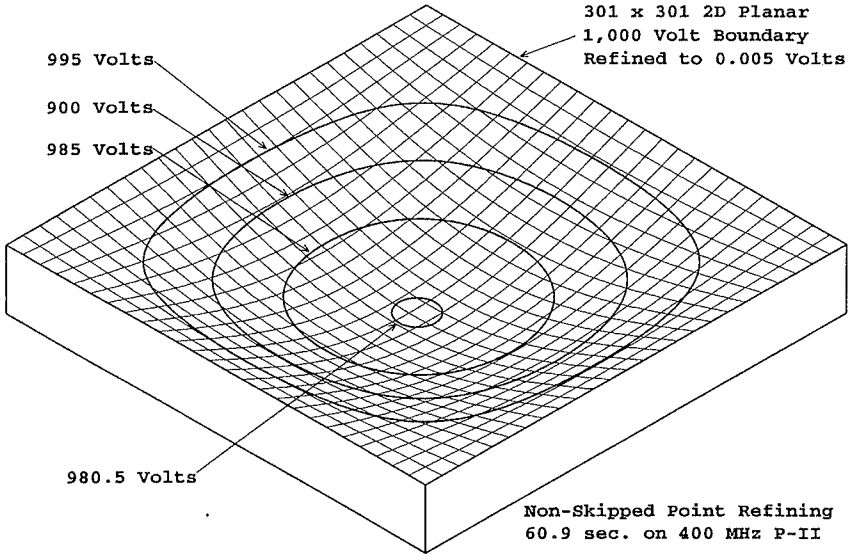

Figure 6-1 Potential energy surface of non-skipped point refined 301 by 301 2D planar array

In theory, after refining is completed, all points in the array should have potentials of precisely 1,000 volts. This is definitely ***not*** the case with ***non-skipped point refining*** (*Figure 6-1*). As Figure 6-1 shows, the center points of the array are only around 980.5 volts (*using default refining parameters - skipped refining turned off*). This is almost ***20 volts*** of systematic error when we have refined to an objective of ***0.005 volts***. ***How can this be? Remember, the convergence objective is based on change and not accuracy***.

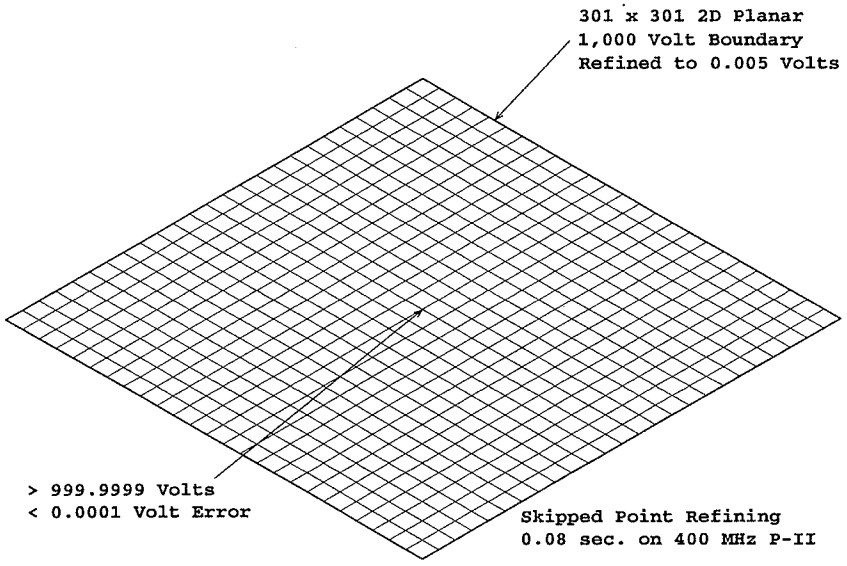

Figure 6-2 Potential energy surface of skipped point refined 301 by 301 2D planar array

The refining process terminated because no point ***changed*** more than 0.005 volts in the last refine iteration. Because it takes the most iterations (*time*) to communicate potentials to the points furthest away from the boundary points, the center points have not settled out. However, since this becomes almost an exponential approach process for distant points the convergence really slows down as you get close. Thus the voltage changes per iteration become very small and the refine terminates. This process required about ***61 seconds on a 400 MHz Pentium II machine***. ***If you***

***set the convergence objective to 0.5 microvolts** (the minimum), **the time required to refine increases to about 137 seconds and the center points are now within 0.4 millivolts of 1,000 volts**.*

Figure 6-2 shows the results with the default refining parameters (*skipped point refining active*). ***All points are closer than 0.1 millivolts of 1,000 volts. The refine time was only 0.08 sec***. Thus, in this contrived example, skipped point refining gave a much better estimate than non-skipped point refining and was ***about 750 times faster***.

One of the strengths of skipped point refining is that it quickly refines linear gradient field areas in potential arrays. Since Laplace's natural propensity is toward the linear gradient, this is a very valuable trait. ***Skipped point refining has eliminated the need for presetting linear gradients*** (*a manual trick users employed in pre 6.0 SIMION versions*).

The only thing you must keep in mind is the sandpaper effect. Skipped point refining creates local disturbances by estimating the values of new points each time the skip factor is reduced. ***These disturbances need to be refined out too!*** While skipped point refining converges more rapidly to the final solution, it also introduces a certain amount of local surface roughness (*the sandpaper effect*). ***Refining to a lower limit reduces this local surface roughness but will never eliminate it completely***.

The Two Types of Refineable Potential Arrays (.PA and .PA#)

Two different types of potential arrays can be refined: **.PA** and **.PA#**. It is important that you recognize how to define each type of array and when to use it.

The Basic Potential Array (.PA)

The **.PA** array is the basic potential array. Its electrode/pole potentials are formally defined in **Modify** or by geometry files. The only way its potentials can be changed quickly is via fast scaling - ***proportional re-scaling*** of ***all*** potentials via **Fast Adjust** (*discussed below*). If you want to change the potential of a single electrode or pole you must change the potentials of ***all*** of the electrode's or pole's ***points*** with either **Modify** (*Chapter 5*) or by changing a geometry file (*Appendix J*), and then re-refine the array.

The **.PA** array is most useful for magnetic pole definitions with non-uniform pole potentials or electrostatic elements like simple single-stage reflectrons. Here proportional scaling can be quite useful. Another good use for **.PA** arrays would be for simple solid electrode beam stops. ***The Fast Adjust Definition arrays, discussed below, now also support proportional re-scaling that is more flexible than available for the .PA arrays***.

The Fast Adjust Definition Array (.PA#)

You should normally use **.PA#** arrays for all your geometry definitions. The **.PA#** arrays are Fast Adjust Definition arrays. Their individual electrodes can be fast adjusted either by you (*discussed below*) or by user programs as the ions fly (*Appendix I*). Moreover, these arrays are easy to define (*e.g. all points in electrode number one are set to **exactly** 1.0 volts*). Up to 30 fast adjustable electrodes/poles can be defined (*the potentials of 1-30 are reserved*). ***SIMION also supports fast scaling with .PA# potential arrays***. All ***non-zero*** potential electrode points that are ***not*** fast adjustable electrode definitions are automatically refined (*solved*) together as fast scaling electrode number 0. These fast-scaling electrode 0 points can be also fast-scaled via user programs as the ions fly (Appendix I).

It is important to remember that while **.PA** arrays are refined, fast adjusted, and used for ion flying, **.PA#** arrays are ***only*** refined. They are ***never*** fast adjusted or used for ion flying. The refining process creates a Fast Adjust Array with the **.PA0** extension. ***This*** is the array that you fast adjust and fly ions through.

You can also create more than one Fast Adjust Array for the same **.PA#** array. This allows you to have ***multiple*** instances of the ***same*** potential array that have ***different*** potentials assigned to their adjustable electrodes/poles. These additional arrays have the **.P?0** extension where **?** is **B - Z**. They are created by loading the desired refine created Fast Adjust Array (.PA0) and then saving it as an array with a **.P?0** extension (*e.g.* **.PB0**). Chapters 4, 5, and 7 further discuss the creation and use for these arrays.

Running Refine

Refine is accessed via the Main Menu Screen. To refine a PA (*potential array*), make sure it is the currently selected PA (*its button is depressed*), and then click the **Refine** button or hit the **<R>** key. Note: SIMION only allows two types of potential arrays to be refined: **.PA** (*basic potential array*) and **.PA#** (*Fast Adjust Definition arrays*). ***SIMION refines an array according to its file name extension.*** Thus you must ***always*** save a Fast Adjust Definition array using a **.PA#** extension or SIMION will consider it to be just a basic **.PA** array.

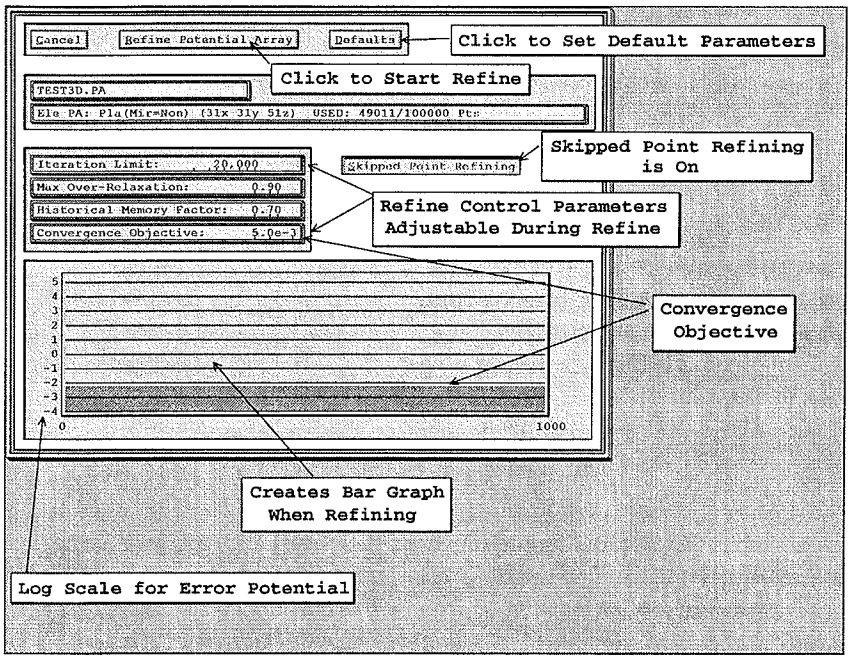

Figure 6-3 Refine Screen for basic potential arrays (.PA)

Refining Basic Potential Arrays (.PA Extension)

The Refine Screen for basic potential arrays (*e.g.* **.PA**) is shown in Figure 6-3. The example below uses the **\SIM7_MISC\TEST3D.PA** file. ***You might want to load it and try refining it yourself for practice.***

Refine Control Parameters

Several refine control parameters can be set before (*and during*) the refining process:

Iteration Limit

The **Iteration Limit** panel controls the maximum number of iterations SIMION will allow before calling a halt to the refining process. This limit exists to prevent SIMION

from going on forever trying to refine an array that is approaching its convergence limit very slowly. The default value of 20,000 is more than enough for almost any refine. ***The Iteration Limit panel is also accessible during the refining process with either .PA or .PA# arrays.***

Max Over-Relaxation

The **Max Over-Relaxation** panel sets the maximum upper limit to the automatic over-relaxation that SIMION is allowed to apply. This value is normally set to 0.90 (*a good value for most arrays*). ***The Max Over-Relaxation panel is also accessible during the refining process for .PA arrays only.***

Historical Memory Factor

The **Historical Memory Factor** panel (*default = 0.7*) sets the time constant for an exponential moving averaging algorithm to calculate an average error term (*iteration to iteration*) that is used to control automatic over-relaxation. The higher the factor the smoother the average error term becomes and the slower the changes in over-relaxation factor. ***The Historical Memory Factor panel is also accessible during the refining process for .PA arrays only.***

The Convergence Objective

The **Convergence Objective** panel (*default = 0.005 volts/Mags*) sets the convergence criteria for terminating the iterative refining process. As discussed above, this parameter does not guarantee any particular level of accuracy. ***The Convergence Objective panel is also accessible during the refining process for .PA arrays only.***

Skipped Point Refining

The **Skipped Point Refining** button is used to select between skipped point and non-skipped point refining. Skipped point refining is selected when the button ***is depressed*** (*default is skipped point refining*). ***The Skipped Point Refining button is NOT accessible when refining any array.***

Selecting Defaults

The **Default** button is used to select the default values for all of the above parameters.

Viewing the Refining Process

When you click the **Refine Potential Array** button, SIMION first performs ***one*** refine iteration with point skipping turned off to see if the array is already refined to the convergence objective (*and stops if it has been*).

The progress of the refining process is displayed in the semi-log display at the bottom of the Refine Screen. Every ten iterations SIMION displays a bar showing the error value at the end of the most current iteration. As the refining process continues the error bars will indicate progress toward the convergence objective (*green region at the bottom*).

If skipped point refining is active a refine will appear to be a series of sawtooths representing successive refines at lower point skipping factors. When the skip number displayed is zero, all array points are being refined (*no points are being skipped*). ***Note: The first two skips are refined to values much lower than the convergence objective.*** This helps resolve areas with linear gradients better, so that the refining process will be faster and more accurate.

What is Actually Refined

SIMION only refines the ***in-memory*** copy of the potential array when it refines a **.PA** array. This means that you must ***save*** the refined array if you want this solution to retained between SIMION sessions.

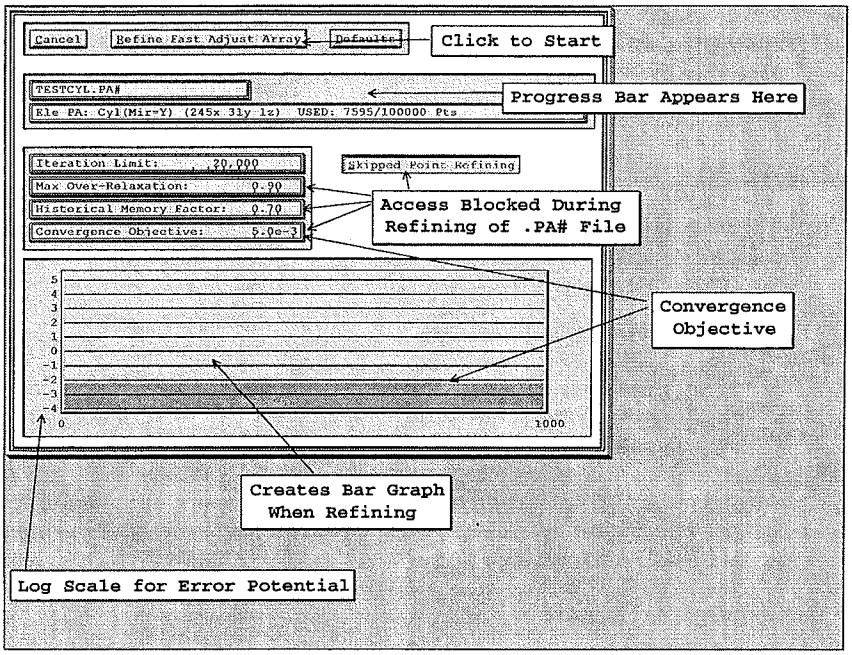

Figure 6-4 Refine screen for Fast Adjust Definition files .PA#

Refining Fast Adjust Definition Potential Arrays (.PA# Extension)

The Refine Screen for fast adjust definition potential arrays (*e.g.* **.PA#**) is shown in Figure 6-4. The example above uses the **\SIM7_MISC\TESTCYL.PA#** file. ***You might want to load it and try refining it yourself for practice.***

Refine Control Parameters

There are several refine control parameters that you can set ***before*** starting the refining process (*as described above in* **.PA** *refining*). ***The only parameter that you can adjust during the refining of* .PA#** *files is the* **Iteration Limit** *panel.*

The Process Involved in Refining a .PA# File

***The first and most important thing to understand is that the* .PA#** *file is not changed in any way by the refining process.* It merely acts as the **template** for creating and refining all the individual electrode/pole solution arrays required to support the fast adjust process. The following is a step-by-step description of what all the activity on your screen is about:

Save a Copy of the .PA# File

You should make sure the a current copy of the **.PA#** file has been saved in the current directory ***before*** starting to refine it. ***The first thing SIMION will try to do is load the disk copy and check it.***

Loading and Checking the .PA# File

SIMION will now load the **.PA#** file from disk and check for adjustable and/or scalable electrode/pole definitions. Remember: The potential of adjustable electrode/pole number one must be ***exactly*** one volt or Mag (*adjustable electrode/pole two ≡ 2.0000 and so on*). Any skipped adjustable electrode values (*e.g. 1,2,3,6,7,8*) will abort the refining process with an error message. SIMION will also check for non-zero non-adjustable electrode/pole definitions that can be fast scaled. If ***either*** legal adjustable electrode

points and/or ***non-zero*** scalable electrode points are found SIMION will start its array creation and refining process.

Creating, Refining, and Saving the .PA0 and .PA_ files

The **.PA0** file is the Fast Adjust file. It contains the solution for all non-adjustable electrodes and poles. SIMION first loads a copy of the **.PA#** file and then coverts it into the **.PA0** file by setting all adjustable electrode/pole points to zero. All non-electrode/non-pole points are also set to zero to insure consistent solution patterns.

SIMION 7.0 supports fast proportional scaling of non-adjustable electrodes/poles in Fast Adjust files (***.PA0***). All non-adjustable electrode/pole points are scanned to find the maximum absolute potential. If the maximum absolute potential is found to be zero (*e.g. all non-adjustable electrode/pole points have a potential of zero*) the zeroed image of the array is simply saved as the Fast Adjust file **.PA0** in the current project directory.

However, if the maximum absolute non-adjustable potential is non-zero (*e.g. at least one non-adjustable electrode/pole point has a non-zero potential*), the non-adjustable electrode/pole points are proportionally scaled so that the maximum potential is 10,000 volts or Mags. The array is then refined using the parameters set before refining began. After refining is completed the file is saved as a **.PA_** file in the current project directory. The in-memory copy of this solution file is then proportionally re-scaled so that the non-adjustable electrode/pole points have the same potentials they had in the original **.PA#** file. This proportionally refined and scaled in-memory array is then saved as a **.PA0** Fast Adjust file in the current project directory. Finally, SIMION appends a trailer to the end of the **.PA0** Fast Adjust file containing all the information that **Fast Adjust** will need when adjusting the potentials of the array.

Creating, Refining, and Saving the .PA1-Z files

The **.PA1** file is the specific solution file for adjustable electrode/pole number ***one***. Files **.PA2-9 & A-U** (*numbers 2-30*) contain the specific solution for each additional adjustable electrode/pole.

SIMION first loads a fresh copy of the **.PA#** file and then coverts it into the **.PA1** file by setting the potentials of all electrode/pole number one points to 10,000 volts or Mags and setting ***all*** other electrode/pole points (*adjustable or non-adjustable*) to zero. All non-electrode/non-pole points are also set to zero to insure consistent solution patterns. The **.PA1** array is then refined using the parameters set before refining began. After refining is completed the array is saved as a **.PA1** file in the current project directory.

The process above is repeated for each adjustable electrode/pole defined. The displayed file name is changed and a progress bar is displayed to update you on SIMION's progress.

Reloading the .PA# File

When all this activity is completed SIMION will reload the original **.PA#** file and exit to the Main Menu Screen.

Saving the Results

There is no need to save the results when you refine a **.PA#** file, because SIMION does the saving ***automatically*** for you.

Fast Adjusting .PA and .PA0 Files

SIMION has specific methods for fast adjusting the solutions of **.PA** and **.PA0** (*or* **.PB0** *and etc.*) arrays. Each of the two types of arrays are fast adjusted differently. ***A .PA file must have been refined for Fast Adjust to serve any purpose.***

The fast adjust process only changes the in-memory copy of the potential array (*e.g.* **.PA** *or* **.PA0**). Be sure to ***save*** the changed in-memory arrays to disk if you want these changes to persist between SIMION sessions. Note: **.IOB** files save and can restore these potentials too (*see Chapter 7*) without having to save the changed potential arrays (*e.g.* **.PA** *or* **.PA0**).

How to Access Fast Adjust

The **Fast Adjust** function can be accessed either from the Main Menu Screen or from within the **View** Function.

Access From Main Menu Screen

To fast adjust a file from the Main Menu Screen, be sure that the file is selected (*its button is depressed*). Now click the **Fast Adjust** button or hit the **F** key to enter the **Fast Adjust** function. ***If you have selected a .PA# array SIMION will automatically load the refined .PA0 array in its place and fast adjust it*** (*if it exists*).

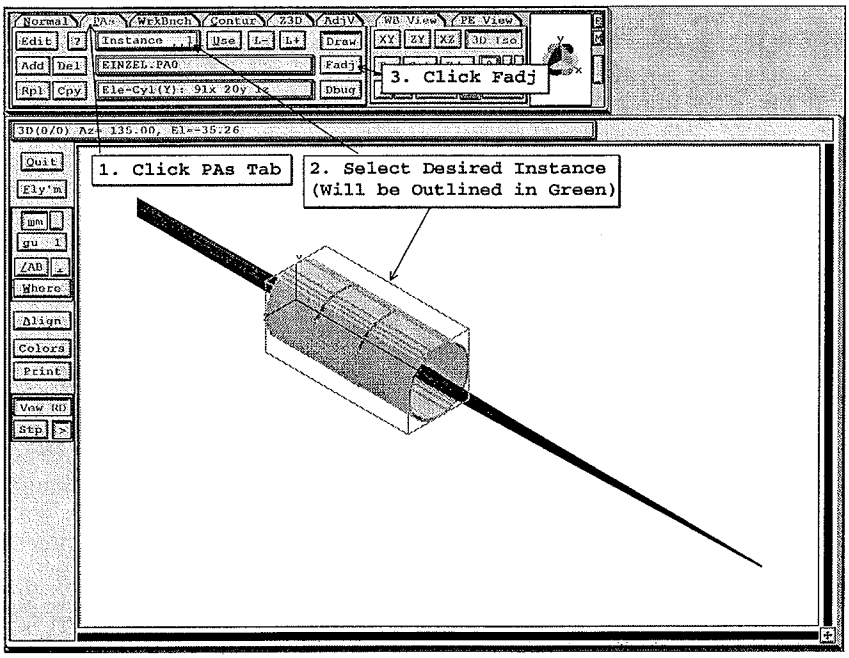

Figure 6-5 Accessing Fast Adjust from View screen

Access From the View Function

To fast adjust within **View** (*Figure 6-5 above*) ***even with ions flying***: Click the **PAs** tab, select the desired array with the **Instance** panel (*the selected instance will be outlined in green as an aid*), and click the **Fadj** button. You many also select the desired instance by pointing the cursor to it in a non-PE view and hitting the **<U>** key. ***Note: Any changes in potentials also impact all other instances that use the same potential array (Chapter 7).***

How .PA Arrays are Fast Adjusted

Remember **.PA** arrays must have been previously ***refined*** for fast adjusting to be meaningful. SIMION fast adjusts **.PA** arrays by proportional fast scaling. Each point in the array is multiplied by a scaling factor. When you enter **Fast Adjust** with a **.PA** array SIMION scans the array to find the electrode/pole point with the largest absolute potential.

Fast Adjust then shows a picture of the array and creates an adjustable panel with the point's current potential (*Figure 6-6*). A red line connects the panel and the point on the view so you can see which point serves as the reference. The **Restore Panel** button is used to restore the potential on the panel to the value it had when **Fast Adjust** was entered. Three view control buttons are provided for 3D arrays (*to assist in seeing point's location*).

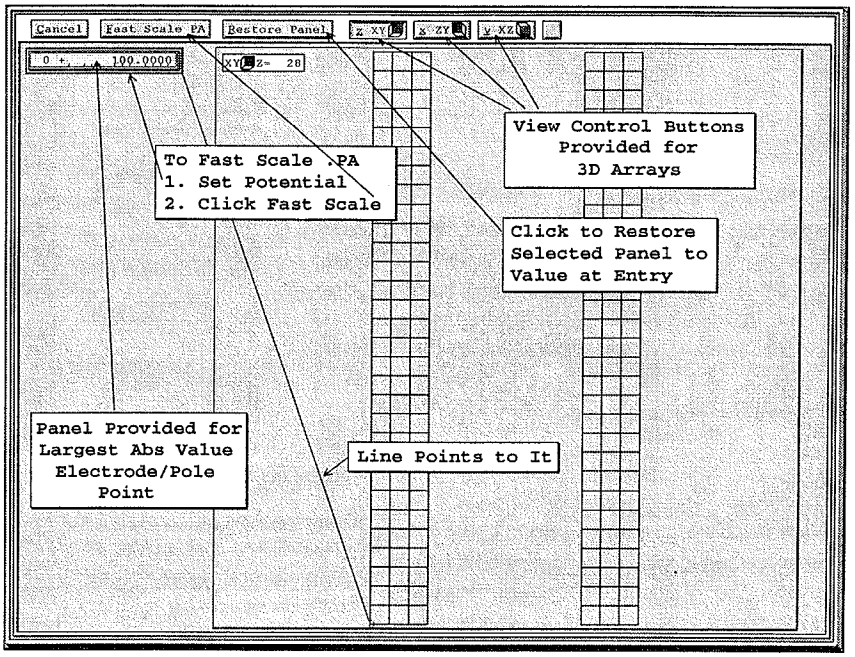

Figure 6-6 Fast scaling control screen for a .PA array

To fast adjust a* .PA, *change the panel to the desired new value *(zero is illegal),* ***and click the*** **Fast Scale** ***button***. Zero is illegal because it would result in all potentials of all points being proportionally scaled to a value of zero. This would result in a loss of all refining information. If you want the ability to fast scale to a zero potential save the array as a **.PA#** file and refine it again to create a **.PA_** fast scaling solution file for the **.PA0** file. Then you can fast scale to zero potential as described below.

How .PA0 Arrays are Fast Adjusted

SIMION fast adjusts **.PA0** (*base solution arrays*) by using data from the specific adjustable solution arrays to provide scaling for potential changes of specific adjustable/scalable electrodes in the **.PA0** (*or* **.PB0** *and etc. arrays*). ***See Appendix E for specific details.***

What you must do is change the desired adjustable or scalable electrode/pole potentials (*via their panel objects*) and click the **Fast Adjust PA** button (*Figure 6-7*). When you point the cursor to a panel it will draw a red line to one of its points. When you point the cursor to point marked with its electrode number, a red line will point to its panel. Three view buttons are provide for any 3D array to aid in electrode viewing.

If any non-zero non-adjustable electrode/pole points are defined, SIMION will automatically create a **.PA_** fast scalable solution file for them during refining and these points can be fast scaled via the ***panel 0*** provided (*Figure 6-7*). Unlike fast scaling of **.PA** arrays ***panel 0*** can be set to a zero potential (*another good reason for using Fast Adjust Definition arrays .PA# to define your geometry*).

Remember, you can share the specific solutions arrays among more than one fast adjust base array *(saves disk space and refine time)*. This allows you to have multiple instances of the same fast adjust array with independently adjustable potentials via additional **.P?0** base arrays *(where* **?** *is* **B-Z**). To create an additional fast adjust base array *(many can be created)*, load the array's **.PA0**

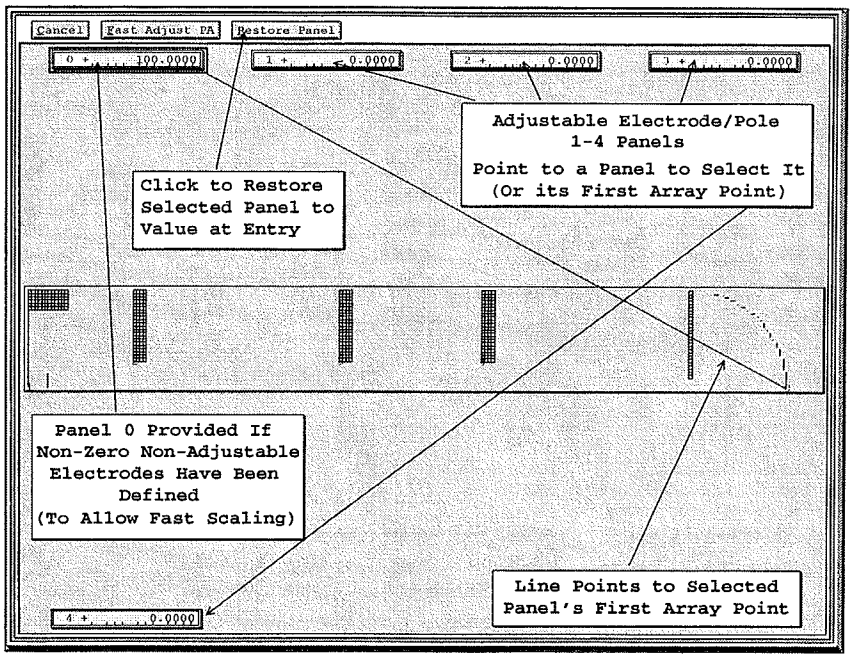

Figure 6-7 Fast Adjust screen for a fast adjust file with four fast adjust electrodes plus a fast scaling electrode

file and save it as a **.PB0** file. The **.PB0** file automatically uses the same fast adjust files as the **.PA0** file. In this case, one instance would be linked to the **.PA0** array and another would be linked to the **.PB0** of the same name *(e.g.* **TEST.PA0** *and* **TEST.PB0**). ***The quick way to switch array connections to an instance in*** **View** ***is to use the*** **Rpl** ***Button in the*** **PAs Control Screen** ***to replace the currently selected instance's array with another one.***

User Programs can Fast Scale/Adjust Arrays too

User programming, discussed in Appendix I, can be used to fast scale and/or fast adjust arrays by various means. These methods can be used to dynamically alter electrode potentials as the ions fly to simulate ion traps, bunchers, time-of-flight instruments, and etc.