A Symmetry Jump Example

Introduction

The user program example in this directory demonstrates how to jump ions between instances. There are applications like SIMS extraction regions where the target, primary beam tubes, and secondary ion extraction tubes cannot be all integrally aligned with the array. This means that if the target's surface is parallel to the grid points, a beam tube or some other important element will have jags for its surface.

The trick

Given that SIMION's grid elements are square there would appear to be no way to get around the problem. However, in this case we could create two images of the extraction volume. The first would be aligned with the target, and the second would be aligned with, perhaps, the exit tube. If we could start the ions flying in instance one and them somehow jump them magically into the second instance at some appropriate point in their trajectories, the ions would start and exit the assembly within aligned electrode elements.

Strategy

One could create two gem files of the same extraction housing. The first would be aligned with the target (**lower.gem**) and the second would be aligned with the exit tube (**upper.gem**).

These files exist in the directory. Please create potential arrays with them and save **lower.gem** as **lower.pa#** and **upper.gem** as **upper.pa#**. It is suggested that you examine the gem files to confirm how they work as well as the arrays that they produce. Please refine both **lower.pa#** and **upper.pa#** so that the demo will work.

The housings are defined based on 0.010" grid spacing (0.254 mm/gu). The lower housing is located with its target ion emission point at the workbench origin. The upper housing is located 22mm above it in z.

Each array has its own user program file (**lower.prg** and **upper.prg**). The following discusses how they work.

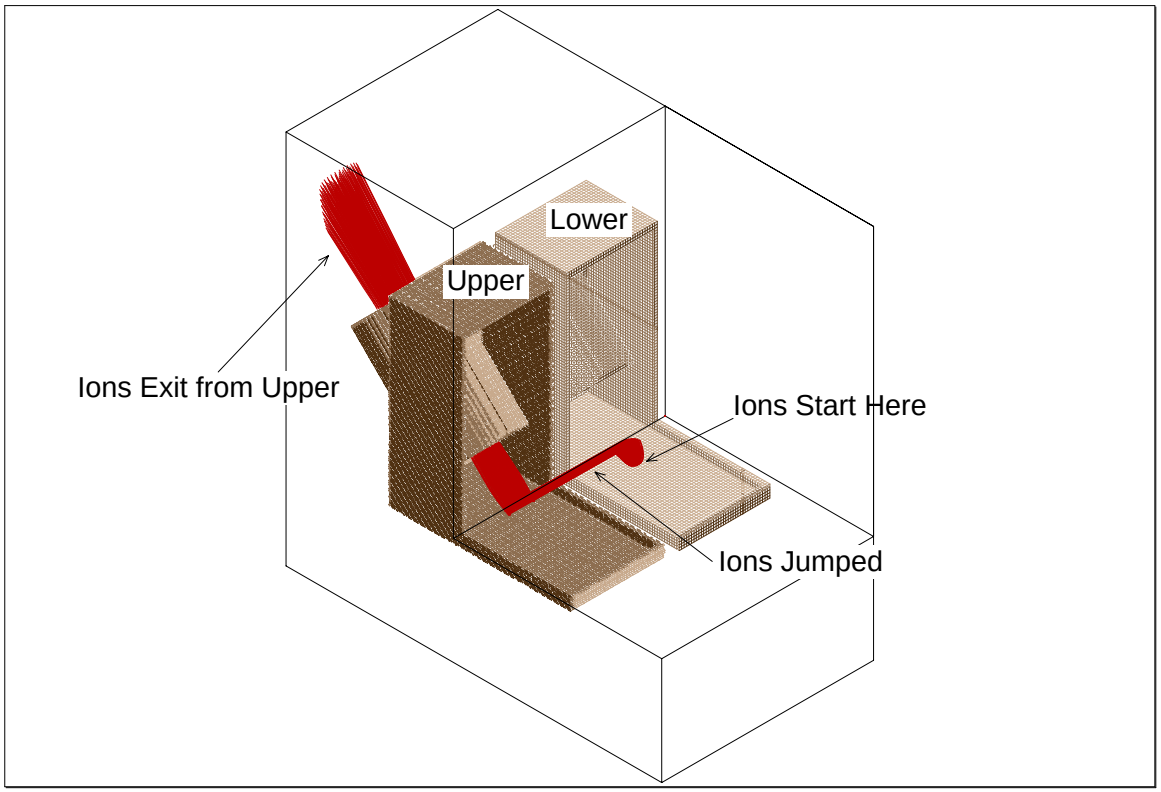
Potential adjustment

Both user program files contain potential adjustments for the primary beam tube and the extraction tube potentials. These potentials are defined by adjustable variables (user adjustable). The program segment used is fast_adjust. You could also use init_p_values (commented out). Fast_adjust was chosen because the arrays are large and the simulation should run faster on machines with limited memory. If your computer has a lot of ram try switching the calls and compare the relative speeds.

Jump control

The **lower.prg** contains an other actions program segment that acts to jump the ions into the upper assembly. Since all ions start at the origin we can calculate the distance from the start point (3D radius) in mm using the **ion_px_mm**, **ion_py_mm**, and **ion_pz_mm** reserved variables. When the

radius exceeds some preset limit – **_jump_radius_mm** (default 4 mm) the z position is jumped by adding 22 mm to the current z position and saving it to **ion_pz_mm**. That is all there is to it.



Running the simulation

Be sure you have created and refined **lower.pa#** and **upper.pa#** as discussed above. Remove all pas from ram. Load the file **jump.iob** into view and click fly'm. With a little 3D cutting you should see the view shown above. Have fun.

Other issues

This simulation was created to demonstrate a simple symmetry jump. It is not intended to serve as a functional design. To drive this point home, you should load and run the 1 eV and 5 eV **.FLY** files. Clearly higher energy secondary ions will have a tough time being extracted.