GEM Geometry File

A SIMION geometry file (GEM) defines electrode geometries using constructive solid geometry (CSG) primitives. CSG operations define shapes using unions and intersections of other basic shapes (e.g. a rectangular slab with a cylinder hole cut through it), a bit similar in concept to machining. GEM files are text files and have the file name extension of ".GEM". SIMION can convert a GEM file to a PA file.

Macro Support (e.g. variable expansion and preprocessing) - 8.0.4

As of SIMION 8.0.4, the GEM processor incorporates a preprocessor that allows things like variable expansion and loops (Issue-I296). which allows GEM files to be more maintainable and easily modified. (This new feature is not described in the 8.0.4 manual except in a footnote.)

A new example GEM file (`examples\geometry\macro.lua`) illustrates the use of this new feature:

```
; Example SIMION GEM file using preprocessing to expand Lua variables
; and macros.
; Preprocessing is a new feature SIMION 8.0.4.
;
; Lines starting with '#' or text inside $() are interpreted as Lua
; code and evaluated.  If $() contains an expression (rather than a
; statement), the result is outputted.  All other text is outputted
; verbatim.

; These variables are intended to be adjusted by the user.
; Note: both # and $ syntax are shown for demonstration.
# local ro = 45     -- outer radius, gu
# local ri = 40     -- inner radius, gi
$(local nshells = 4) ; number of shells

; Now do some calculations on those variables.  Note: math.ceil(x) is
; the ceiling function: it returns the integer closest to but no
; greater than x.
# local hw = math.ceil(ro * 1.1) -- half width, gu
# local nx = hw * 2 + 1          -- num array points in x

; Define array size.
pa_define($(nx),$(nx),$(nx), planar, non-mirrored)

; Declare a function to be used later.
; This returns the voltage for electrode number n.
$( local function volts(n)
     return n^2 * 10 + n + 1
   end )

; This locate centers the shells in the array.
locate($(hw),$(hw),$(hw), 1, 0,0,0) {
; Now define a loop that creates each shell.
# for n=1,nshells do
#   local rop = ro * n / nshells
#   local rip = ri * n / nshells
    electrode($(volts(n))) {  ; example calling a function
      fill {
        within{sphere(0,0,0, $(rop))}
        notin {sphere(0,0,0, $(rip - 1))}
      }
    }
# end
}

; Note: the following two lines are equivalent:
fill { within { box3d(0,0,0, $(nx),0,$(nx)) }}
# _put("fill { within { box3d(0,0,0, " .. nx .. ",0," .. nx .. ") }}")

; Print output to the log window.  This can be
; useful for debugging.
# print("Hello from GEM file. nx=" .. nx)
```

When SIMION loads a GEM file containing macros (e.g. `macro.gem`), it processes those macros, writes the result to temporary file (e.g. `macro.processed.gem`), and loads that temporary file.

It also possible to do your own GEM file preprocessing outside of SIMION with your own programming tools. This was more popular prior to the introduction of the above feature but can still be useful in some cases.

Extensions in 8.1

- The `pa_define` command has various extensions: `pa_define (nx,ny,nz,symmetry,mirroring,type,ng,dx_mm,dy_mm,dz_mm,gu,surface=fractional)`
  - `pa_define` was extended to allow definition of grid cell sizes `(dx_mm, dy_mm, dz_mm)` in anistropically scaled PAs. See *Anisotropically scaled grid cell - GEM File Support*
  - `pa_define` allows a PA to be marked "not refinable" by using an underscope in the `pa_define` command `type` argument (e.g. `electric_` or `magnetic_`). This is equivalent to setting **`simion.pas pa.refinable`** to `false`. This is useful for PAs that store some type of special field that should not be automatically refined in the usual manner (e.g. magnetic vector potential, space-charge, dielectric constants, permeability, gas flow, etc.). It is used extensively in SIMION Example: magnetic_potential and SIMION Example: dielectric. Added in 8.1.1.0.
  - `pa_define` supports a new `surface=fractional` named parameter to enable surface enhancement, for higher accuracy. See *Electrode Surface Enhancement / Fractional Grid Units*.
  - `pa_define` supports a new `surface=auto` named parameter which improves accuracy of curved surfaces position when not using surface enhancement. This is not as ideal as surface enhancement.

surface=auto option in pa_define [8.1.1.25]

Obtaining optimal field accuracy of curved surfaces had been tricky prior to the addition of the *Electrode Surface Enhancement / Fractional Grid Units* feature in SIMION 8.1.1. Electrode surfaces that did not exactly align to PA grid points had to be placed instead on nearby grid points, and certain ways of making this placement were more accurate than others. Cutout volumes (e.g. `notin` GEM commands) were also prone to error. Consider the example of defining the field between concentric spheres of radii 40 and 80 mm. We can now accurately and simply define this in a GEM file using the ("fractional") surface enhancement option like this:

```
pa_define(101,101, cylindrical,xy, surface=fractional)
e(1) { fill { within { circle(0,0, 40) } } }
e(2) { fill { notin  { circle(0,0, 80) } } }
```

However, perhaps there are still rare cases where you prefer not to use surface enhancement. For whatever the reason (see below), SIMION 8.1.1.25 supports a new `surface=auto` option in the `pa_define` statement of GEM files to achieve a more accurate alignment of surfaces in a simple manner when not using surface enhancement. It can be used simply like this:

```
pa_define(101,101, cylindrical,xy, surface=auto)
e(1) { fill { within { circle(0,0, 40) } } }
e(2) { fill { notin  { circle(0,0, 80) } } }
```

To understand this feature, let's consider the way this used to be done in SIMION 7.0/8.0....

You might naively define the concentric spheres like this:

```
pa_define(101,101, cylindrical,xy)
e(1) { fill { within { circle(0,0, 40) } } }
e(2) { fill { notin  { circle(0,0, 80) } } }
```

However, without care, surfaces could be off effectively by about 0.5-1.0 grid unit (gu), thereby distorting fields by the same amount as if the electrodes were misaligned by this distance in the real physical system (e.g. machining or assembly error). The above is nearly equivalent to this:

```
pa_define(101,101, cylindrical,xy)
e(1) { fill { within_inside_or_on { circle(0,0, 40.5) } } }
e(2) { fill { notin_inside_or_on  { circle(0,0, 80.5) } } }
```

The `within` and `notin` commands apply a +0.5 grid unit (gu) adjustment to the radius of the fill, which improves field accuracy for the `within`, but for the `notin` we actually want more like a -0.5 gu adjustment. In fact, electrode #2 in the above two examples is visibily off slightly when measured in the View screen. We have typically recommended using `notin_inside` rather than `notin`:

```
pa_define(101,101, cylindrical,xy)
e(1) { fill { within        { circle(0,0, 40) } } }
e(2) { fill { notin_inside  { circle(0,0, 80) } } }
```

Visually that looks about right if you examine the PA on the View screen, and the calculated field is reasonably good. However, the `notin_inside` (as with `within_inside`) actually applies a nearly 0 gu not -0.5 gu adjustment, so fields are slightly biased. But even this is not ideal in the case of spheres. In fact, the optimal adjustment is often not 0.5 gu but about 0.35 gu, as has been observed for spherical capacitor studies ref and other internal studies on flat and curved shapes rasterized to PA's.

In SIMION 8.1.1.25, a new `surface=auto` option has been added that automatically applies the best adjustment to `within` and `notin` fill types. It can be used simply like this:

```
pa_define(101,101, cylindrical,xy, surface=auto)
e(1) { fill { within { circle(0,0, 40) } } }
e(2) { fill { notin  { circle(0,0, 80) } } }
```

and it's nearly identical to this:

```
pa_define(101,101, cylindrical,xy)
e(1) { fill { within_inside { circle(0,0, 40.35) } } }
e(2) { fill { notin_inside  { circle(0,0, 79.65) } } }
```

`surface=auto` also affects other shape types like `box`, `parabola`, `hyperbola`, and `polyline` by applying approximately 0.35 gu offsets in the appropriate directions.

`surface=auto` is even useful when electrodes **do** exactly align to PA grid points (like surfaces that are flat, orthogonal to the axes, and have coordinates that are integral multiples of grid units) because it will

automatically get the surfaces boundary points right. Consider defining a 4x4 mm box with a 2x2 mm hole inside, which obviously easily aligns to points on a 1 mm/gu grid. You might naively try to do it like this:

```
pa_define(11,11,1, planar,n)
e(1) { fill {
  within { box(1,1,  5,5) }
  notin  { box(2,2,  4,4) }
} }
```

However, the `notin { box(2,2, 4,4) }` (as is also true when using `notin_inside_or_on`) excludes from the cut the PA grid points on both the inside and border of the 2x2 box. You want to instead use a `notin_inside` to only exclude the grid points on the "inside" (not border) of the 2x2 box:

```
pa_define(11,11,1, planar,n)
e(1) { fill {
  within       { box(1,1,  5,5) }
  notin_inside { box(2,2,  4,4) }
} }
```

That's pretty simple once you know about it, but it's easier to do this right just by enabling `surface=auto` like this:

```
pa_define(11,11,1, planar,n, surface=auto)
e(1) { fill {
  within { box(1,1,  5,5) }
  notin  { box(2,2,  4,4) }
} }
```

or even (if you prefer) expressing it like this:

```
pa_define(11,11,1, planar,n, surface=auto)
e(1) { fill { within { box(1,1,  5,5) } } }
n(0) { fill { within { box(2,2,  4,4) } } }
```

When `surface=auto` is enabled (and this is also true of `surface=fractional` as of 8.1.1.25), any PA grid points that precisely align to an edge of a `within` or `notin` fill volume are made to be electrode points, which is usually what you want. In other words, a `notin` inside an `e` or a `within` inside an `n` (both of which remove electrode material) will be treated as an "inside" fill to avoid removing the electrode points from the border. This largely avoids fiddling with `within_inside`/`within_inside_or_on`/`notin_inside`/`notin_inside_or_on` fill variants since `within` and `notin` will now just do the right thing. In fact, with `surface=auto`, it's not usually needed nor recommended to use fill types other than `within` and `notin`; for example, the following is **not** correct but behaves similar to the "naive" approach mentioned earlier:

```
pa_define(11,11,1, planar,n, surface=auto)
e(1) { fill { within{box(1,1, 5,5)}  notin_inside_or_on{box(2,2, 4,4)} } }
```

In fact, the following four GEM files are all correct and generate exactly the same PA:

```
pa_define(11,11,1, planar,n)
e(1) { fill { within{box(1,1, 5,5)}  notin_inside{box(2,2, 4,4)} } }

pa_define(11,11,1, planar,n, surface=auto)
e(1) { fill { within{box(1,1, 5,5)}  notin{box(2,2, 4,4)} } }

pa_define(11,11,1, planar,n, surface=fractional)
e(1) { fill { within{box(1,1, 5,5)}  notin{box(2,2, 4,4)} } }

pa_define(11,11,1, planar,n, surface=auto)
e(1) { fill { within_inside_or_on{box(1,1, 5,5)}
                    notin_inside{box(2,2, 4,4)} } }
```

Please note, however, that although `surface=auto` is generally preferred over `surface=none` (i.e. the default old behavior in SIMION 7.0/8.0) for ease and accuracy reasons, `surface=fractional` (see *Electrode Surface Enhancement / Fractional Grid Units*) is still preferred over both of these. So, there's not really a reason to use `surface=auto` unless for some reason you don't want to use surface enhancement. It's not really clear what that reason would be except maybe to generate PA's that are refinable under SIMION 8.0 or to compare accuracy differences with/without surface enhancement. A GEM file using `surface=fractional` can be quickly switched to `surface=auto` without any changes, but switching to `surface=none` may cause 1 gu errors unless additional changes are made to `within/notin` fill types due to the semantic difference these have under `surface=none`. `surface=auto` is somewhat of an anachronism that behaves between `surface=none` and `surface=fractional` but happened to be implemented after both for completeness.

Changes

- 8.1.1.25: A GEM: New `surface=auto` parameter in `pa_define`. This provides a simple way to more accurately position curved surfaces and cutout (`notin`) volumes when not using surface enhancement (`surface=fractional`). This particularly applies to `circle`/`sphere`/`cylinder`, `hyperbola` (Issue-I371), `parabola`, `polyline`, `notin`, and others. Surface enhancement is still better, but if you don't want to use surface enhancement for some reason, then this is better than nothing. See *surface=auto option in pa_define [8.1.1.25]*.
- 8.1.1.25: C GEM: `within_inside_or_on`/`notin_inside_or_on` now shift the electrode surface outward by a small 0.0001 gu offset. (`within_inside` and `notin_inside` already apply this offset but in the reverse redirection.) This more reliably ensures points "on" the shape boundary are filled even when small numerical round-off occurs in operations like `polyline` and scaling. Example: `fill{within_inside_or_on{box(1,1,5,5)}}` and `fill{within_inside_or_on{polyline (1,1, 5,1, 5,5, 1,5)}}` behave identically now, as expected. [*]
- x GEM/polyline: `polyline` GEM command accuracy has been improved. Previously, the points on the polygon edge might not be optimally filled with the desired electrode/non-electrode point type. This particularly affected `within_inside`/`notin_inside`/`within_inside_or_on`/`notin_inside_or_on` rather than `within`. However, under `surface=fractional`, it also affected `within`/`notin` (but is somewhat cosmetic since electrode point types on the surface are not critical for field accuracy under `surface=fractional`). Example: `polyline(10,5 15,5 15,10 20,10 20,15 15,15 15,20 10,20 10,15 5,15 5,10 10,10)` (cross).
- 8.1.1.25: C GEM/Surface: PA grid points that precisely align to an edge of a `within`/`notin` fill volume are made to be electrode points if `surface=fractional` or `surface=auto` is enabled. In other words, a `notin` inside an `e` or a `within` inside an `n` (both of which remove electrode

material) will be treated as an “inside” fill to avoid removing the electrode points from the border. This largely avoids fiddling with `within_inside`/`within_inside_or_on`/`notin_inside`/`notin_inside_or_on` fill variants since `within` and `notin` will now just do the right thing. The default option (`surface=none`) retains the old behavior for compatibility though. Example: `e(1){fill{within{box(1,1,5,5)} notin{box(1,1,5,5)}}}` and `e(1){fill{within{box(1,1,5,5)}}} n(0){fill{notin{box(1,1,5,5)}}}` both create a 4x4 hollow box with infinitesimally thin walls under `surface=auto` or `surface=fractional`. See *surface=auto option in pa_define [8.1.1.25]*.

- 8.1.1.25: c GEM: `polyline`, `points`, and `points3d` commands can now accept an arbitrary number of points. Previously these were limited to 100, 100, and 65 points.
- 8.1.1.25: x GEM: Fix `polyline` problems when using exactly 100 points. [*sf-t1338]