Interfacing neBEM and Garfield++

RD51 collaboration meeting, 24 June 2020

neBEM

- BEM field solvers are an attractive technique for MPGDs as they require discretising only the boundary of the domain, but not the domain itself.
- For an in-depth introduction to the subject see Supratik's lecture from earlier this week: `https://indico.cern.ch/event/911950/contributions/3898133/`
- Up to now, neBEM has been available as a standalone package (written in C), and through an interface to classic (Fortran) Garfield.

Garfield++ interface to neBEM

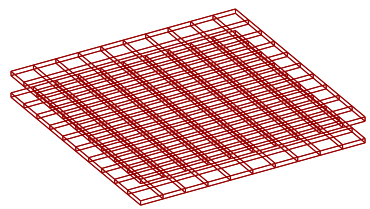
- Implementation follows closely what was done for (Fortran) Garfield.
- The geometry is described in terms of a list of `Solid` objects.
- Each `Solid` knows how to determine its surface panels (polygons).
- The `ComponentNeBem3d` interface class
  - retrieves the surface panels of the solids present in the geometry,
  - finds/resolves overlaps, and
  - splits the polygons into rectangles and rectangular triangles ("primitives").
- The primitives (together with their boundary conditions) are then passed on to neBEM.
- neBEM splits the primitives into smaller rectangles and rectangular triangles ("elements"), according to the specified target size, fills the influence matrix, performs the inversion, and calculates the charge density at each element.

Example 1

- As an example with only fixed-voltage boundary conditions, let's consider a parallel-plate capacitor.

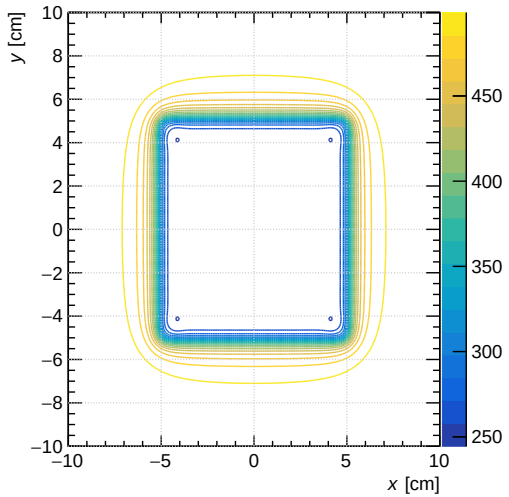
```
// Create a  geometry consisting of two boxes.
SolidBox box1(0, 0, -0.5, 5, 5, 0.1);
SolidBox box2(0, 0,  0.5, 5, 5, 0.1);
GeometrySimple geo;
MediumConductor metal;
geo.AddSolid(&box1, &metal);
geo.AddSolid(&box2, &metal);
// Apply fixed potential boundary conditions to the boxes.
box1.SetBoundaryPotential(1000.);
box2.SetBoundaryPotential(0.);

// Pass the geometry to the neBEM interface class.
ComponentNeBem3d nebem;
nebem.SetGeometry(&geo);
nebem.SetTargetElementSize(1.);
nebem.Initialise();
```



"Elements" as seen by neBEM.

```
// Plot isopotential contours at z = 0.2.
ViewField fieldView;
fieldView.SetComponent(&nebem);
fieldView.SetArea(-10, -10, -1, 10, 10, 2);
fieldView.SetPlane(0, 0, 1, 0, 0, 0.2);
fieldView.PlotContour();
```



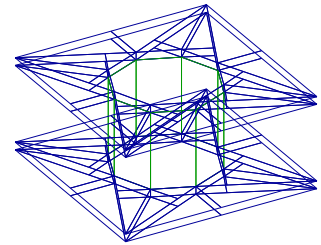
Example 2

- Standard GEM.

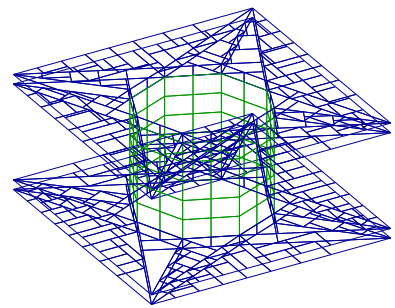
```
// Top and bottom planes.
SolidBox box1(drftx, drfty, drftz, 0.5 * lX, 0.5 * lY, 0.5 * drftlZ);
box1.SetBoundaryPotential(drftV);
SolidBox box2(bkplx, bkply, bkplz, 0.5 * lX, 0.5 * lY, 0.5 * bkplLZ);
box2.SetBoundaryPotential(bkplV);
// Top and bottom metal layers.
SolidHole topCu(upcprx, upcpry, upcprz,
                0.5 * dia, 0.5 * dia, 0.5 * lX, 0.5 * lY, 0.5 * upcprLZ);
topCu.SetBoundaryPotential(upcprV);
topCu.SetSectors(3);
SolidHole btmCu(lwcprx, lwcpry, lwcprz,
                0.5 * dia, 0.5 * dia, 0.5 * lX, 0.5 * lY, 0.5 * lwcprLZ);
btmCu.SetSectors(3);
// Kapton
SolidHole Kapton(kptx, kpty, kptz,
                 0.5 * dia, 0.5 * dia, 0.5 * lX, 0.5 * lY, 0.5 * kptLZ);
Kapton.SetSectors(3);
Kapton.SetBoundaryDielectric();
```

```
GeometrySimple geo;
geo.AddSolid(&box1, &Cu);
// ...

ComponentNeBem3d nebem;
nebem.SetGeometry(&geo);
// Set the target element size.
nebem.SetTargetElementSize(10.e-4);
nebem.SetMinMaxNumberOfElements(1, 5);
// Request periodicity.
nebem.SetPeriodicityX(lX);
nebem.SetPeriodicityY(lY);
nebem.SetPeriodicCopies(10, 10, 0);
nebem.UseLUInversion();
nebem.Initialise();
```



"Primitives" (after overlap removal).



"Elements".

Next steps

- Finalise validation.
- Update user guide and provide tutorials/examples on the website.
- If you are curious: checkout the `nebem` branch of the Garfield++ repository (`https://gitlab.cern.ch/garfield/garfieldpp/-/tree/nebem`).
  - Additional dependencies are minimal (GSL).

To do

- Improve the speed of the field evaluation (in particular for periodic layouts).
- Implement charging up and other "advanced features".
- ...
- As usual, help and feedback are very welcome!