LArTPC Detector Response Calculations
Using **B**oundary **E**lement **M**ethod

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

μBoone Sim

9 Aug 2016

Boundary Element Method (BEM) Overview

1. Discretize (mesh) boundary electrode **surfaces**.
2. Define (Dirichlet) scalar potential on each mesh element (triangle).
3. Fit (Neumann) surface-normal boundary field.
4. Integrate Laplace equation $\nabla^2\phi = 0$, evaluate at boundary.
5. Evaluate solution at points in the volume.

Compare BEM and FEM:

	BEM	FEM
domain:	2D surface mesh	3D volume mesh
easy:	away from surface	near to surface
fits:	boundary-normal field	volumetric field
eval:	arb. volume point	volume mesh points
both:	CPU and memory intensive, limited geometries	
external:	stepping and averaging current responses	

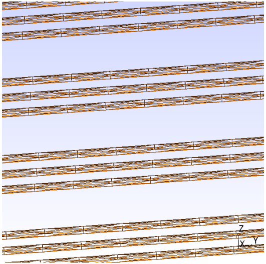
General Calculation Overview

High-level steps:

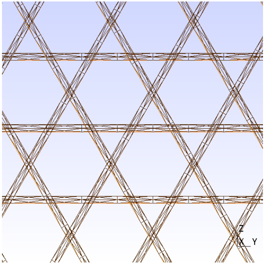
- Drift fields: $\phi_{drift} \to \vec{E}_{drift} \to \mu \to \vec{v}_{drift} \to$ **paths**: $\{p\}$
  - $\vec{v}_{drift} = \mu(E_{drift})\vec{E}_{drift}(\vec{r}(t))$
  - Get path $\vec{r}_p(t)$ by stepping through velocity field $\vec{v}_{drift}$.
- Shockley-Ramo "**weighting**" potential for electrode k
  - $\phi_{weight,k} \to \vec{E}_{weight,k}$
  - Electrode k at 1V, all others at 0V.
- **Current on wire** k due to charge moving along path p:
  - $i_{k,p}(t) = q\vec{E}_{weight,k} \cdot \vec{v}_{drift}|_p$
- **Response function** for wire k is average over paths:
  - $< i_k(t) >= \frac{1}{N} \sum_{p=1}^{N} i_{k,p}(t)$

$\to$ for now: paths start on 1mm grid, 16mm in front of U-plane,
$\to$ response is average over paths in half-pitch "wire region"

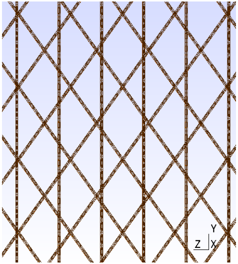
Wire Meshes



"Parallel":
3mm pitch and gap
all wires parallel



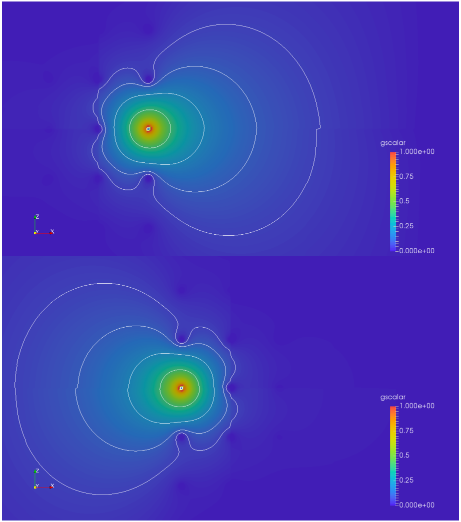
"MicroBooNE":
3mm pitch and gap
60° angles for U/V.

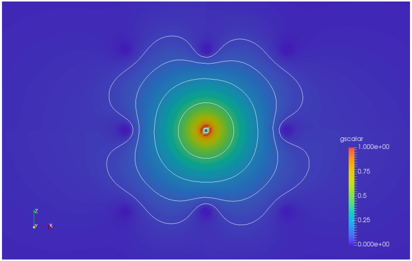


"DUNE":
5mm pitch and gap
35.7° angles for U/V.

- "Parallel" used to reproduce 2D calculations.
- Geometry parameterized to facilitate exploring different configurations.

Parallel Wires - Slice Through Weighting Potentials

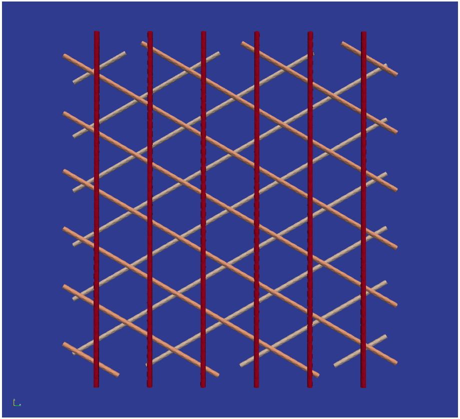


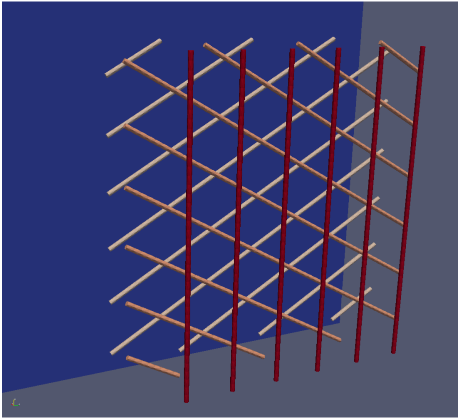


- U and W (left) and V (above) planes.
- X-Z slices through plane of symmetry.
- Lines: 5%, 10%, 20%, 40% weights.
- Initial qualitative agreement with Garfield 2D calculations.

→ more exhaustive comparisons needed, but satisfactory enough to push on.

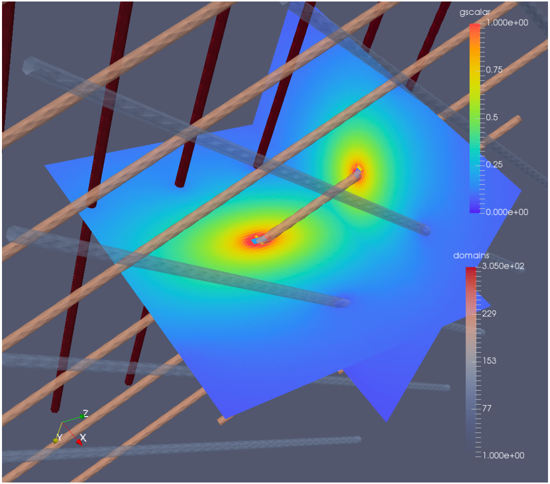
MicroBooNE Geometry Patch





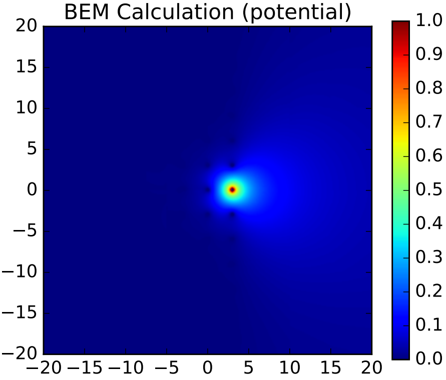
- Wires parameterized by pitch, angle, bounding box, radius.
- Single plane at +20mm for drift potential

V-plane $\vec{E}_{weight}$ Slices

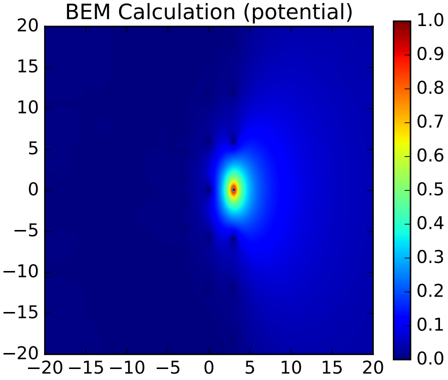


- Slices in X-Z and X-Y planes.
- Note mismatchs between evaluated voxel grid and wire mesh.

Weighting Potential - 2D vs. 3D wire pattern



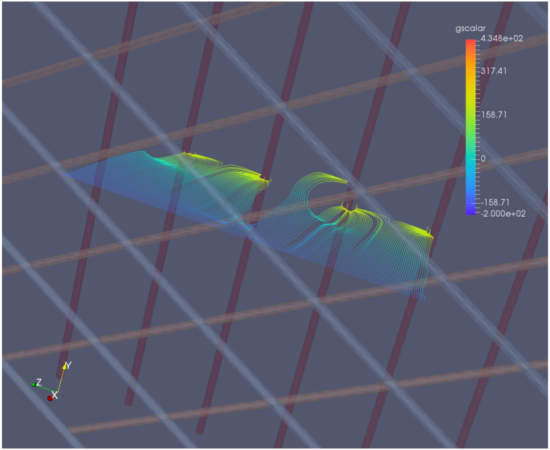
Parallel wires



MicroBooNE wires.

Clear distortion in extent and shape. Not surprising.

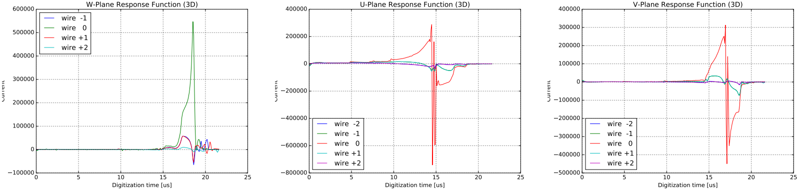
Paraview stepping from line source



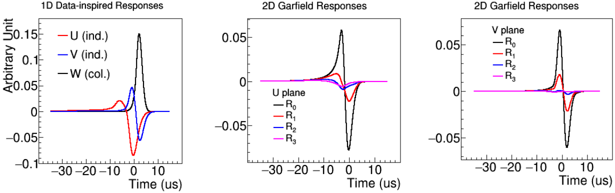
- Line source 2mm in front of U-plane, paths colored by drift potential.
- U-plane transparency violated due to imprecision right near U-wire.

Initial Response Functions

Preliminary and Subject to Change!



Collection and U and V Induction. Central and nearest neighbor wires.



Data inspired and 2D Garfield calculations.

Known Issues Needing Work

1. Improve precision of fields near wires.
   - Voxelization $\rightarrow$ **on-demand sampling** of ϕ_{drift} while stepping
   - Will remove big RAM pig, allow for batch processing.
   - Gives room for **finer surface meshing**.
2. Correct termination of paths.
   - Currently, stepping just knows fields (including interiors of wires!)
   - Must **terminate stepping** when path “hits” a wire.
3. Explore systematics in response function vs path location.
   - Understand **spatial variations** and properly smooth them.
4. Limited transverse coverage
   - Expect to need $\sim \pm 10$ wires (based on data obs.)
   - Problem scales somewhere between $N_{wires} \rightarrow N^2_{wires}$
5. uBoone is “easy first test”, next up:
   - DUNE wire crossing pattern
   - APA edge regions

The Software

https://github.com/brettviren/larf

- Expect a name change!
- Ready for other users and developers, **welcome!**
- Interfaces: **simple command line program** or **Python modules**.
- Supports fantastic **Paraview** visualization app.
- Provides various "management systems": configuration, data storage, result provenance, workflow.

→ Warning: **documentation is trailing code development** so let me know if you are interested in using/developing and I'll do some freshening.