# 3D Detector Response Calculations Using **B**oundary **E**lement **M**ethod

Brett Viren

Physics Department

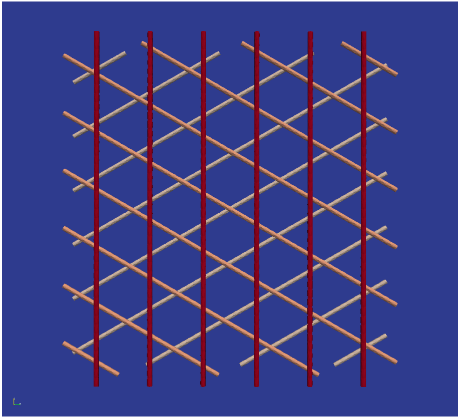
BROOKHAVEN NATIONAL LABORATORY

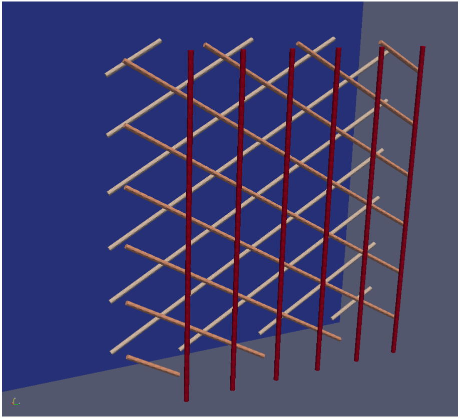
BNL-$\mu$Boone
10 Aug 2016

# Overview

- Status report with focus on active, technical problems.

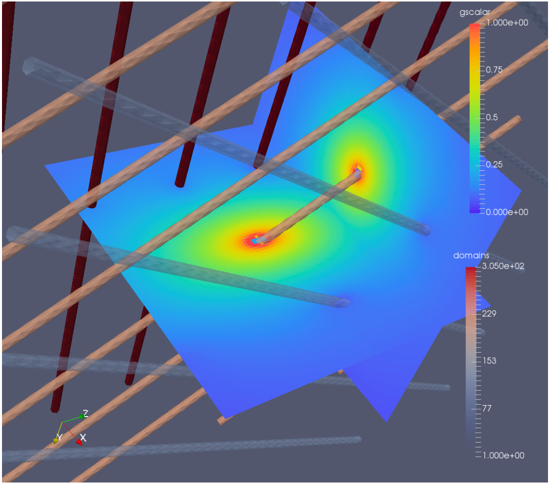
# MicroBooNE Geometry Patch





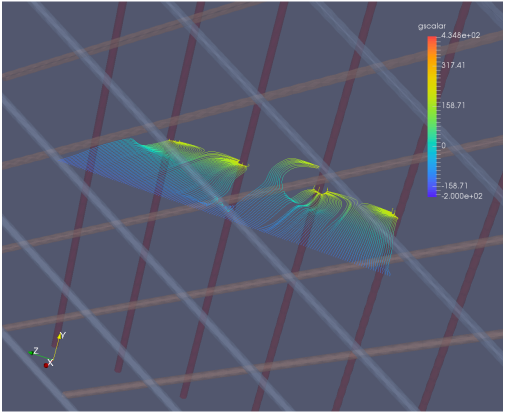
- Wires parameterized by pitch, angle, bounding box, radius.
- Single $40 \times 40$mm plane at +20mm for drift potential
- Wire planes fill a $20 \times 20 \times 6$ mm box

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




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

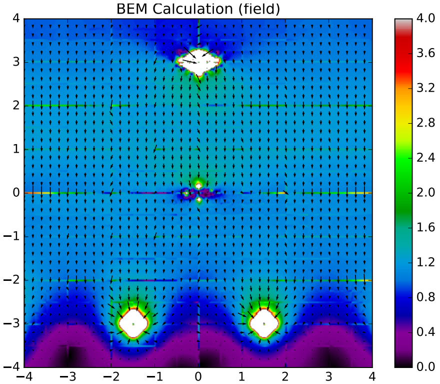
# Paraview stepping from line source




- Line source 2mm in front of U-plane, paths colored by drift potential.
- U-plane transparency violated due to imprecision right near U-wire.
- Likely, **voxelization** is mostly to blame. **Finite mesh size** and **gradient not respecting wire boundary** are 2nd order culprits.

# Mesh Sampling Precision Reminder

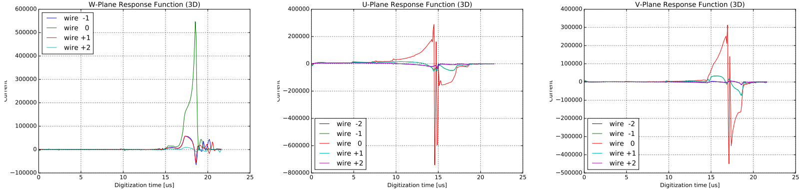
A particularly egregious example.



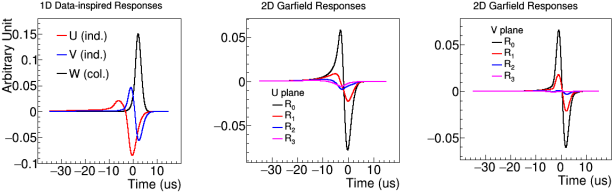

- BEM++ allows setting “Gaussian Quadrature Order”.
- 3 knobs: number of samples of boundary conditions on each mesh triangle for “near”, “middle” and “far” ranges.
- Large scale discontinuity artificats due to shifting between these ranges.

# Initial Response Functions (given known issues)

Preliminary and Subject to Change!

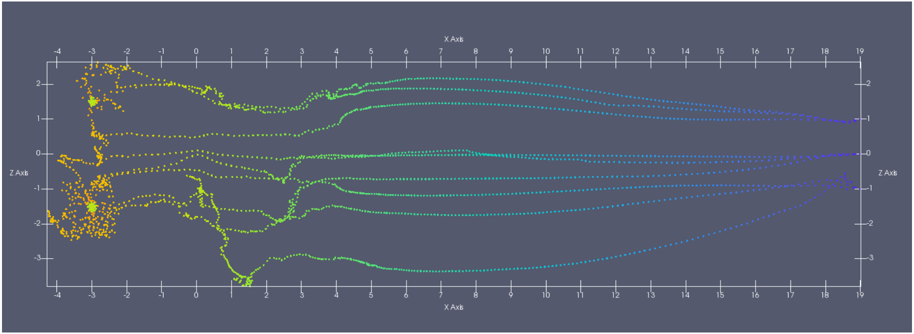



Collection and U and V Induction. Central and nearest neighbor wires.




Data inspired and 2D Garfield calculations.

# Stepping with on-demand field eval




- ∼20cm paths, color is drift potential, 1mm grid starting points
- Uses **on-demand field evaluations**
- Still **point-and-shoot** stepping (every 0.1us).
- Still **low Gaussian Quadrature precision**.
- Still **ignores wire hits**.

# To Do - fix/tune/optimize/test

- ~~Change from voxelized $\phi_{drift}$ to **on-demand**.~~(done)
  - Requires much less RAM and CPU for given spatial resolution.
  - But, does evaluation + stepping together (no eval result caching)
  - Each point uses 7 drift-potential evaluations to get gradient for $\vec{E}_{drift}$.
- ~~Replace point-and-shoot with **Runge-Kutta 5th order**~~(ready)
  - $\sim 6\times$ more CPU needed, tradeoffs to consider:
    - Force step size to desired (eg, $0.1\mu$s steps in time): Slow!
    - Use adaptive error control, bigger steps in general but then, spline fit and sample: Fast but is it accurate, especially at the wires?
- ~~Increase **Gaussian-quadrature order**.~~(cfg parameter)
  - Requires a "few" $\times$ more CPU (I've not carefully measured)
  - Hopefully fixes the abrupt shifts in field.
- Geometry. Need to:
  - be able to query geometry at step time to stop "hit" paths.
  - enlarge geometry to +/- 10 wires (Yichen+Xin's work).
  - enlarge "cathode" even more to provide more parallel drift field.

# Leon's Advances with FEM

DocDB 6236

2D comparison with garfield

- Good looking current waveforms
- Found a time offset difference.
- Wants Garfield's steps $(\vec{r}, t)$ for detailed comparison (Yichen, help?)

3D uBoone

- Can do drifts over "unit cell" ($\sim 3 \times 6$ mm)
- Drift paths look good and "less weird" than my current ones.
    - Ie, straight and parallel in region away from wires

FIN

# 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 | |

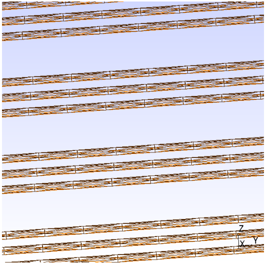
# General Calculation Overview

High-level steps:

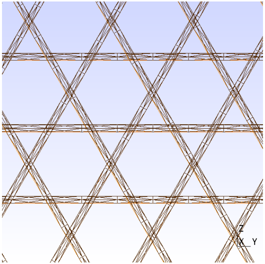
- Drift fields: $\phi_{drift} \rightarrow \vec{E}_{drift} \rightarrow \mu \rightarrow \vec{v}_{drift} \rightarrow$ **paths**: $\{p\}$
    - $\vec{v}_{drift} = \mu(E_{drift})\vec{E}_{drift}(\vec{r}(t))$
    - Get path $\vec{r}_p(t)$ by stepping through velocity field $\vec{v}_{drift}$.
- Shockley-Ramo "**weighting**" potential for electrode $k$
    - $\phi_{weight,k} \rightarrow \vec{E}_{weight,k}$
    - 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:
    - $< i_k(t) >= \frac{1}{N} \sum_{p=1}^{N} i_{k,p}(t)$

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

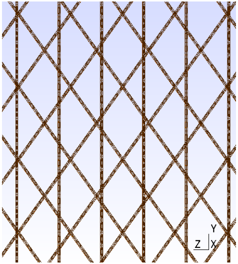
# Wire Meshes



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



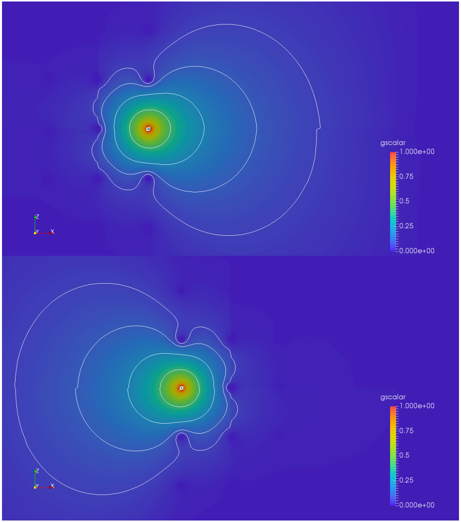
"MicroBooNE":
3mm pitch and gap
60° angles for U/V.

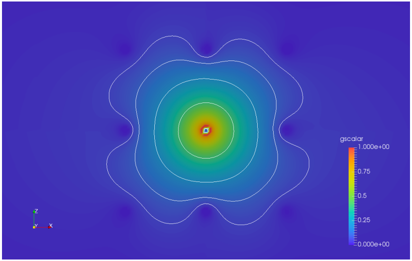


"DUNE":
5mm pitch and gap
35.7° angles for U/V.

- "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.

# 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.

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