online version

A new 3D simulation package for solid-state detectors in Julia - SolidStateDetectors.jl

F. Fischer, L. Hauertmann, X. Liu, O. Schulz,
M. Schuster, A. J. Zsigmond

MAX-PLANCK-GESELLSCHAFT

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

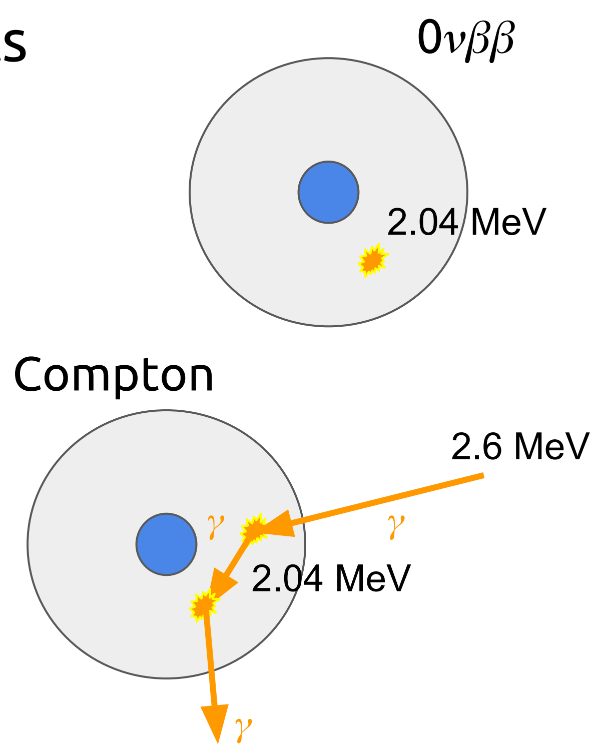
Deutsche Physikalische Gesellschaft
Aachen - 2019

Motivation

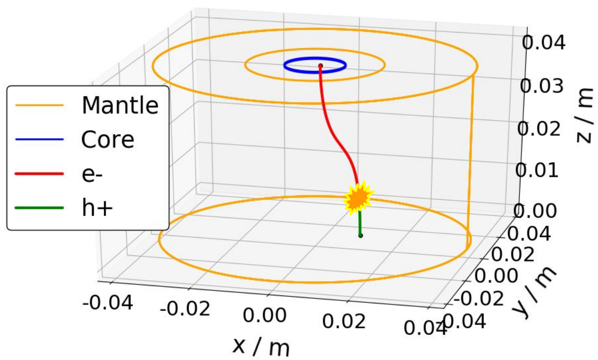
Why do we need pulse shape simulation?

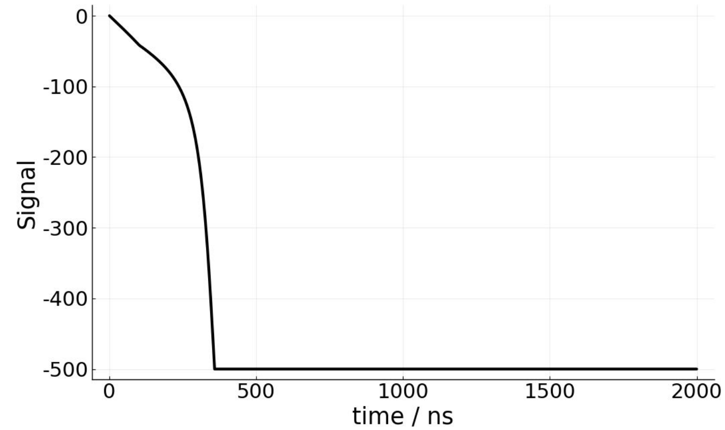
- Rare Event Searches: e.g. $0\nu\beta\beta$ decay
- Reduce and identify background events
- Understand your detector
  - Understanding mobilities
  - Temperature dependence of mobilities
  - ...

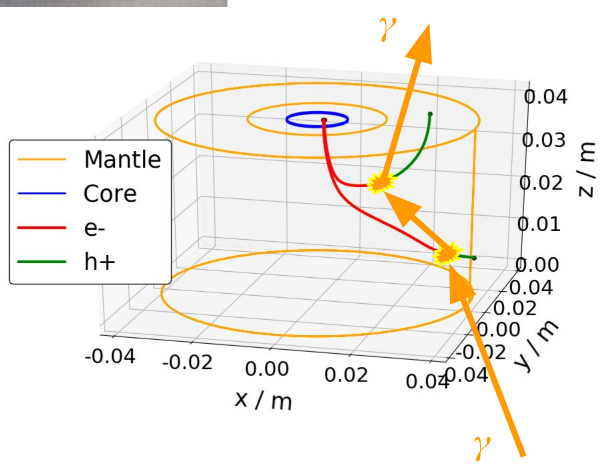
$\rightarrow$ Pulse Shape Discrimination

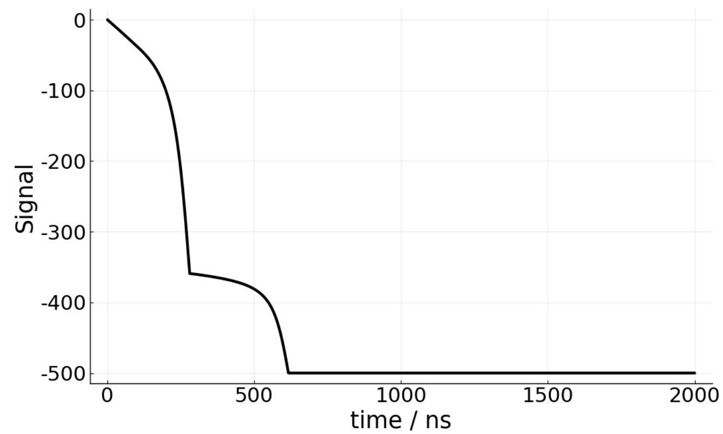


Motivation









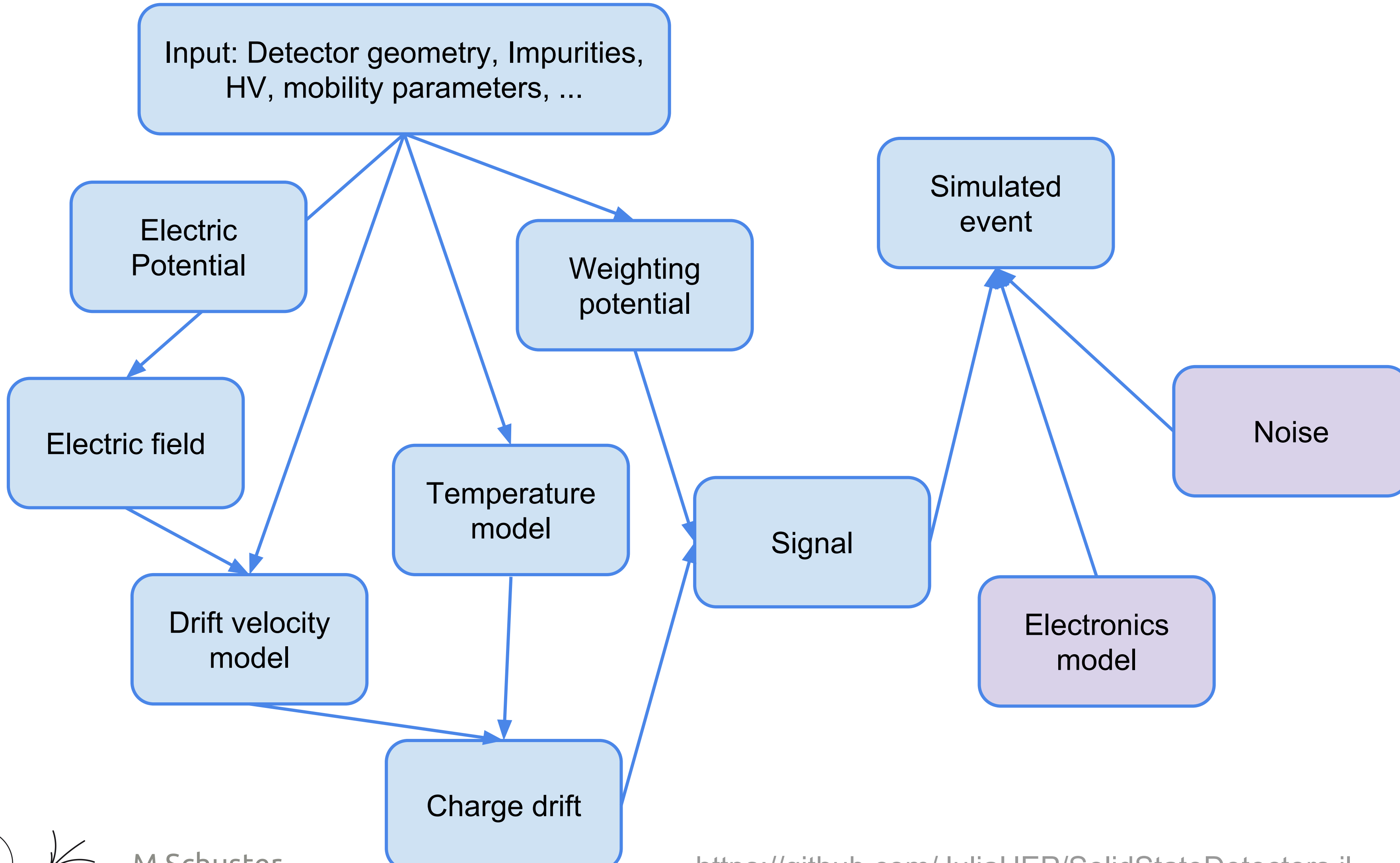
1) [Background-free search for neutrinoless double-β decay of 76Ge with GERDA] - [Agostini, M.] *et al.* Nature 544 (2017) 47 arXiv:1703.00570 [nucl-ex]

Motivation

Why do we need SolidStateDetectors.jl ?

- 3D field calculation
  $\longrightarrow$ segmented detectors / crystal axes effects
- Fast field calculation
  $\longrightarrow$ detector design
- Easily configurable parameters
  $\longrightarrow$ fit mobilities and their temperature dependence
- Open source
  $\longrightarrow$ easier development within community

Overview



Input: Detector geometry, Impurities, HV, mobility parameters, ...
Electric Potential
Weighting potential
Simulated event
Electric field
Temperature model
Noise
Signal
Drift velocity model
Electronics model
Charge drift

Human readable input files, e.g. json

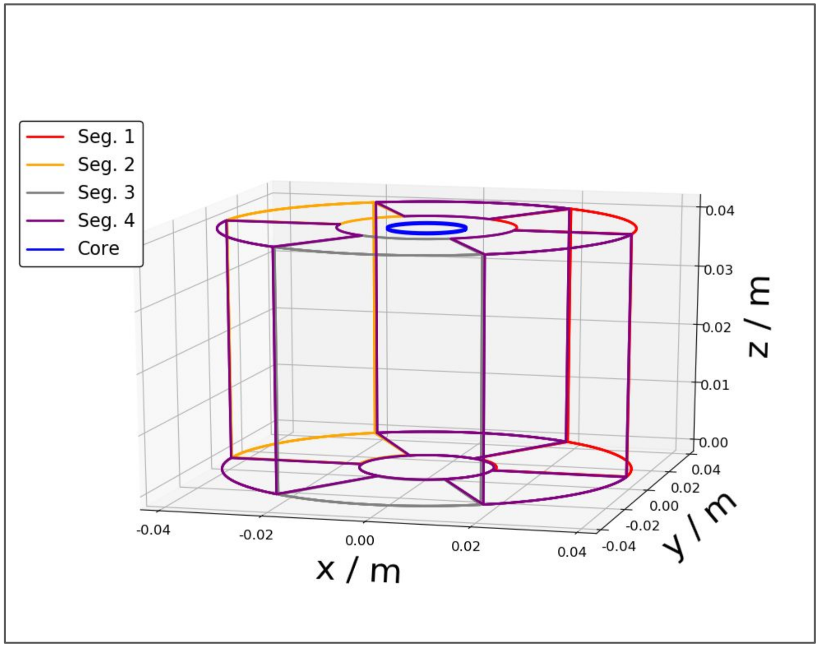
```
"bulk_type": "n",
"geometry": {
     "unit": "mm",
     "world": {
          "material": "Vacuum",
          "geometry": [
               {
               "type": "Tube",
               "rStart": 0.0,
               "rStop": 41.0,
               "phiStart": 0.0,
               "phiStop": 360.0,
               "zStart": -3.0,
               "zStop": 43.0
               }
          ]

     },
     "crystal": {
          "material": "HPGe",
          "geometry": {
          ...
```

- All parameters of the calculation set in config files (.yaml, .json, ...)
  - Geometry parameters
  - Segmentation
  - Drift model parameters
  - etc.
- No hard-coding
- Easy to read

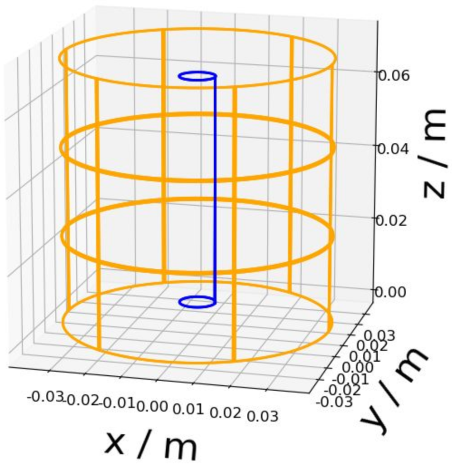
Broad range of geometries

Pre-defined geometries

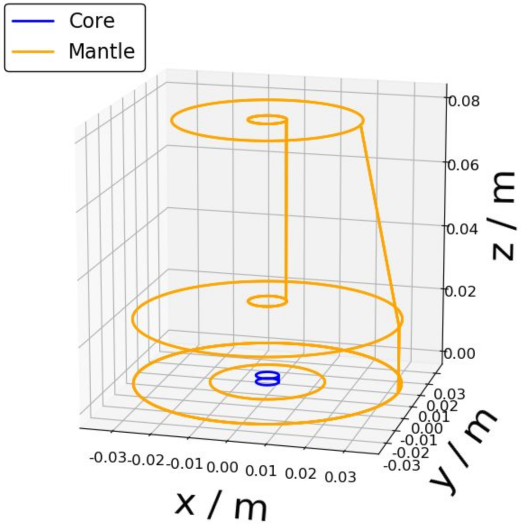
Segmented Point-Contact



True Coaxial



Inverted Coaxial



Field calculation

- Electric potential calculated by Gauss' law

$$\nabla(\epsilon_r(\vec{\mathbf{r}})\nabla\varphi(\vec{\mathbf{r}})) = \frac{\rho(\vec{\mathbf{r}})}{\epsilon_0}\ , \qquad \vec{\mathbf{r}} = (r, \phi, z)$$

$\varphi(\mathbf{r})$ electric potential, ϵ_0 vacuum permittivity, $\epsilon_r = 16$ dielectric constant of germanium, $\rho(\mathbf{r})$ impurity density

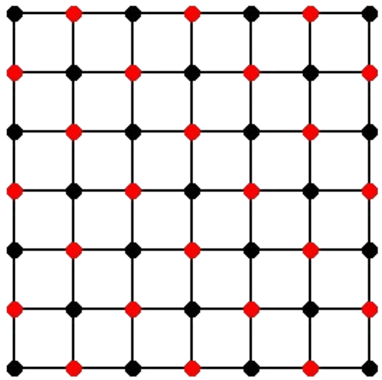
- Weighting potential calculated by Gauss' law

$$\nabla(\epsilon_r(\vec{\mathbf{r}})\nabla\varphi_W(\vec{\mathbf{r}})) = 0$$

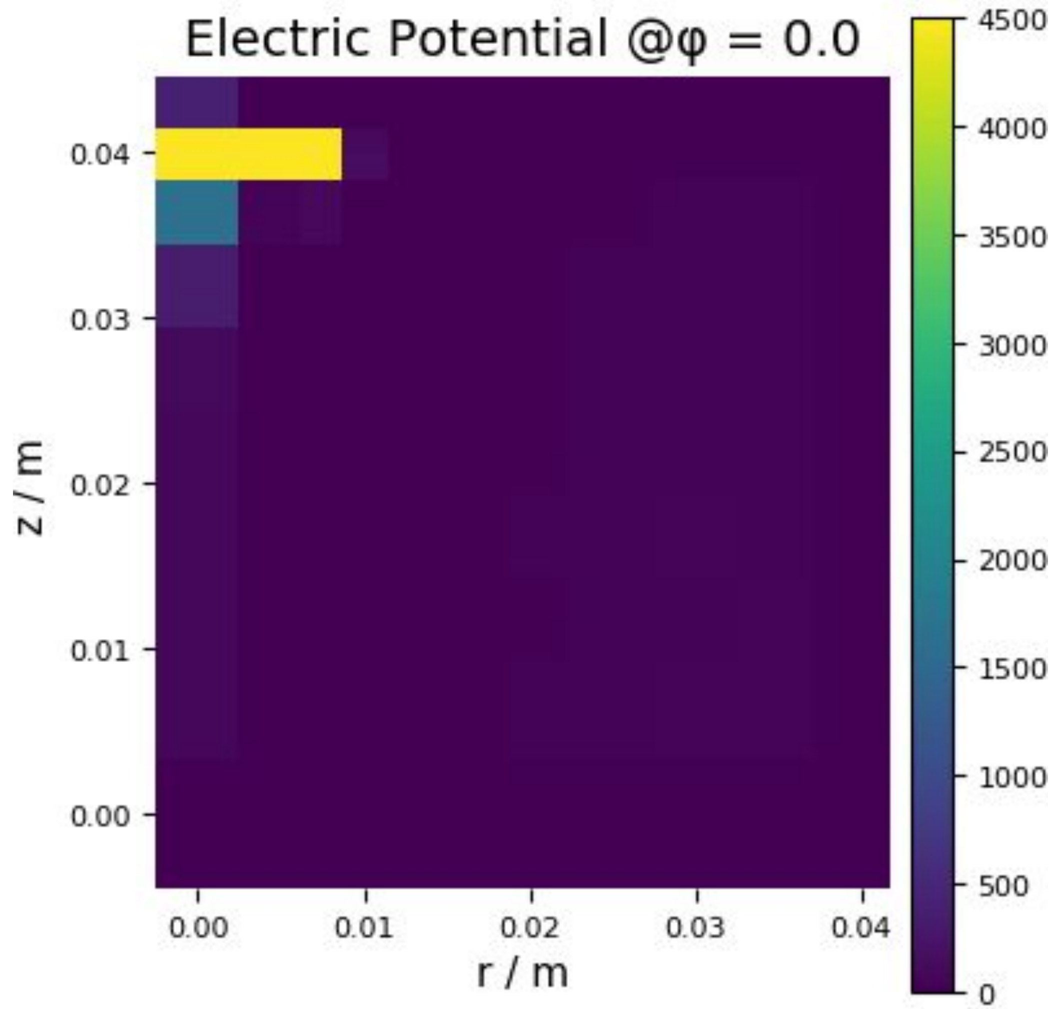
with the boundary conditions that the weighting potential equals unity on the considered electrode and zero otherwise

Field calculation

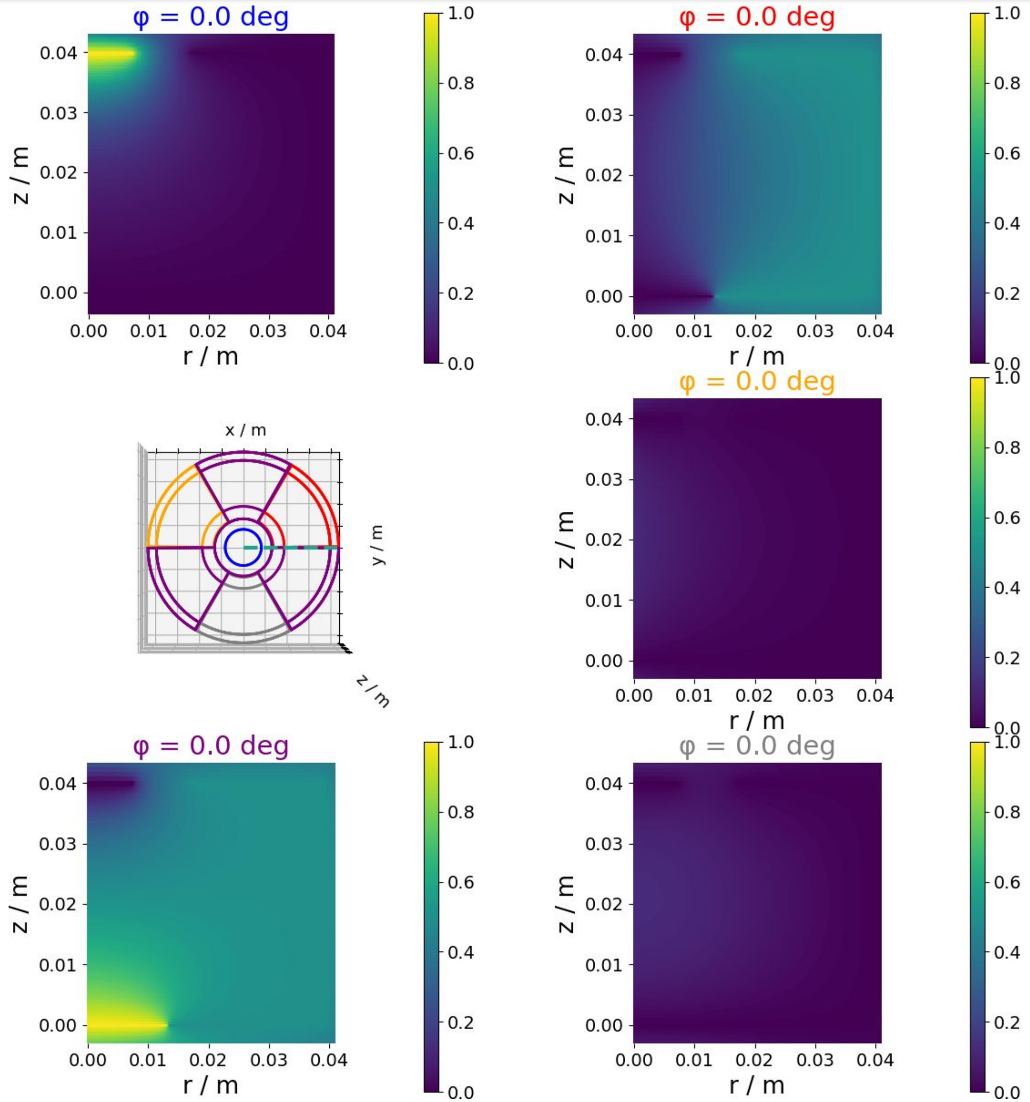
- Adaptive grid → save computation time
- Julia → easy to vectorize loops (AVX/AVX2) → speed up
  https://software.intel.com/en-us/articles/vectorization-in-julia
- Successive over-relaxation (SOR) → fast convergence
  https://people.eecs.berkeley.edu/~demmel/cs267-1995/lecture24/lecture24.html
- Red-Black-Algorithm → parallelism
  https://people.eecs.berkeley.edu/~demmel/cs267-1995/lecture24/lecture24.html



Electric Potential Calculation



Weighting Potentials



Drift velocity model

- Parallel to the principal crystal axis the drift velocity is parallel to the electric field

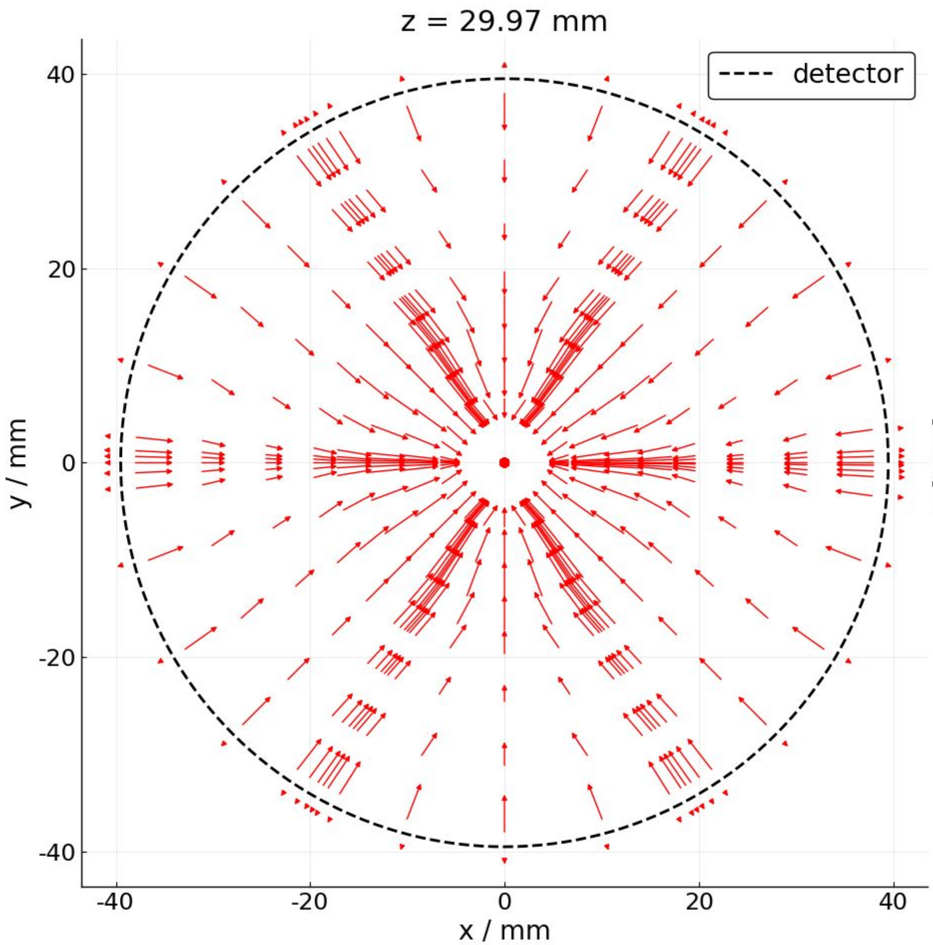
$$v_l = \frac{\mu_0 E}{(1 + (E/E_0)^\beta)^{1/\beta}} - \mu_n E$$

- μ_0, E_0, β and μ_n parameters different for electrons and holes and for <100> and <111> axes
- Different models for electrons and holes implemented from [1] B. Bruyneel et al., NIM A 569 (2006) 764 also used by e.g. AGATA
- Between these axes the drift velocity is more complicated

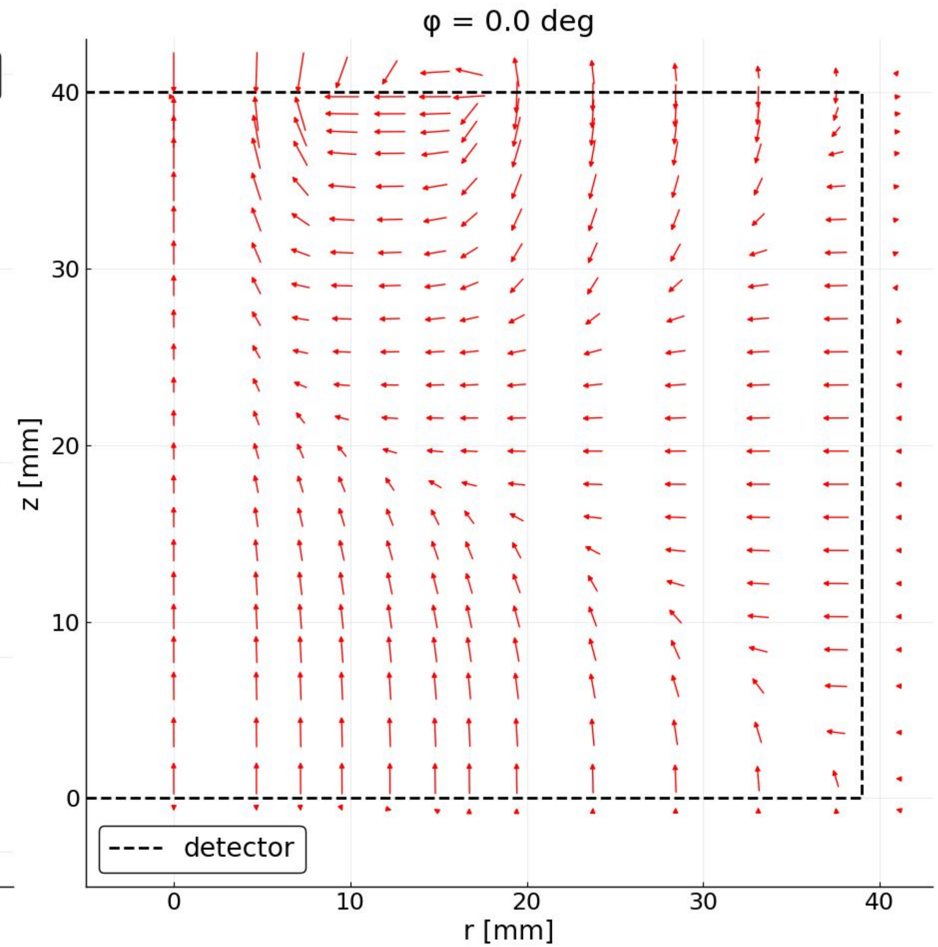
Drift velocity model

Electron Drift Velocity Field

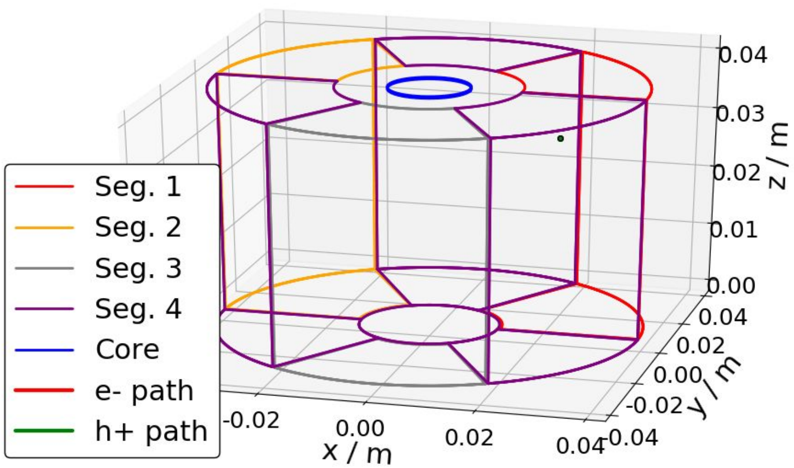
rφ - cut

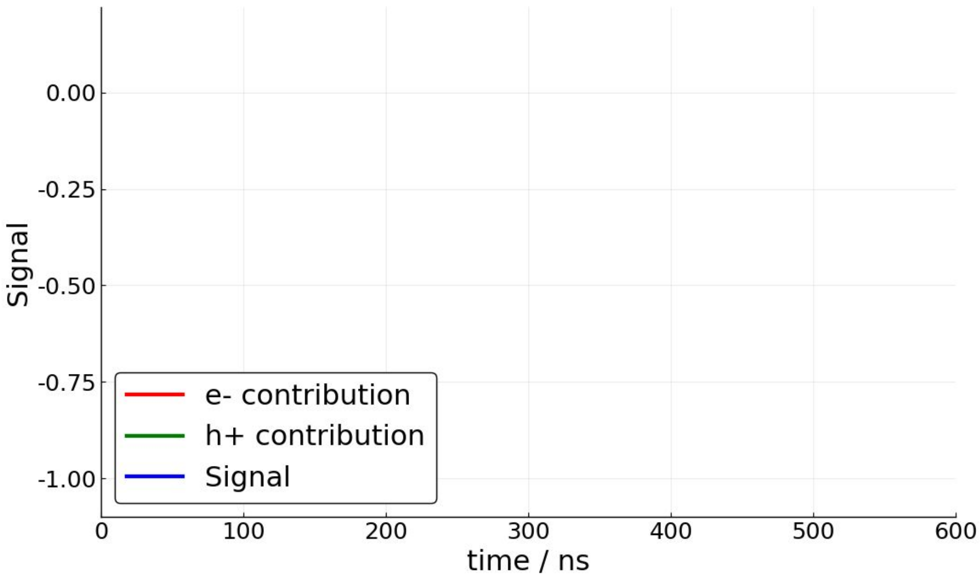


rz - cut



Charge drift and core signal





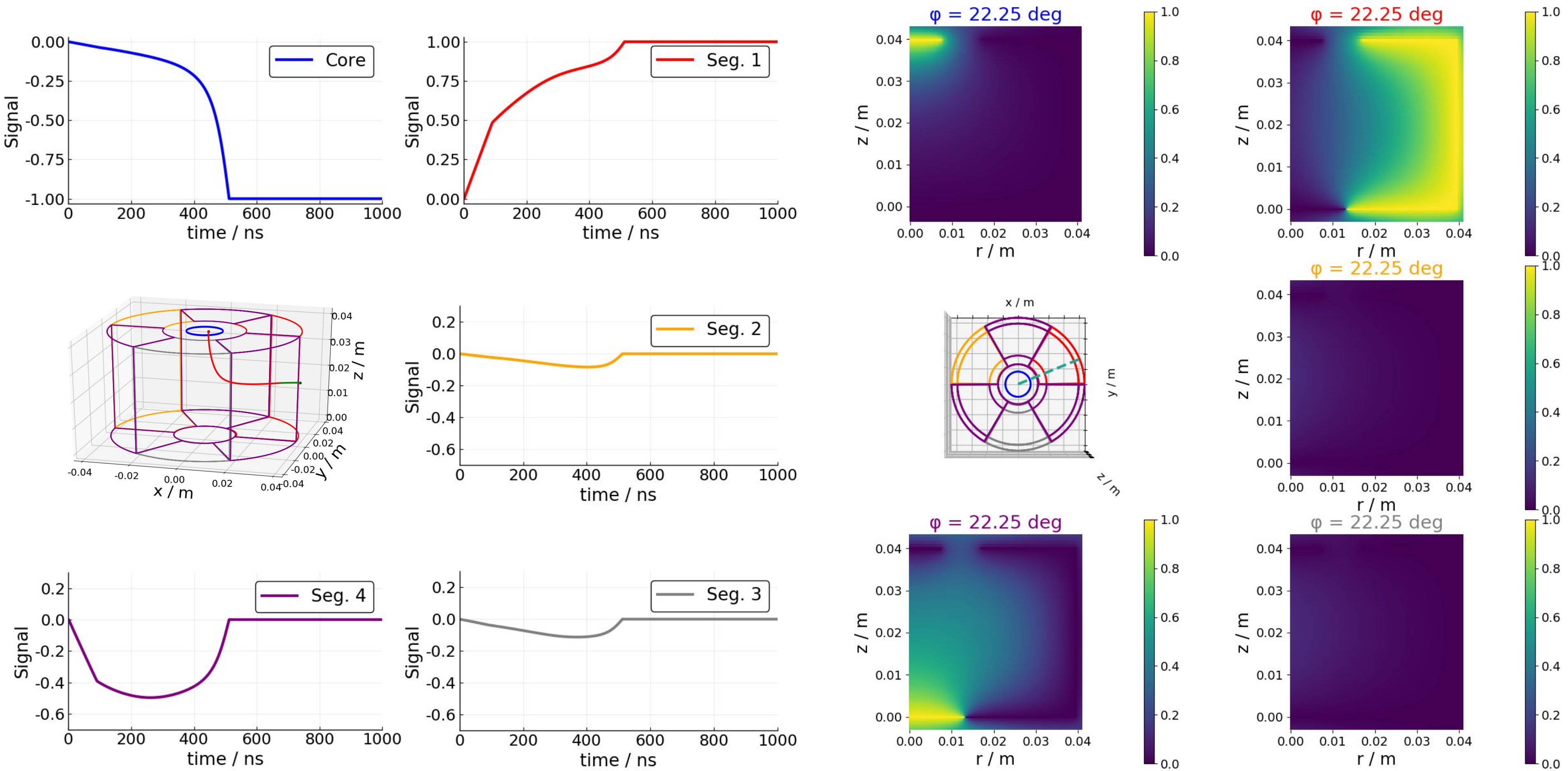
Drift path

$\mathbf{x}(t) = \mathbf{x}(t\text{-}\Delta t) + \mathbf{v}(\mathbf{x}(t\text{-}\Delta t))\cdot\Delta t$

Induced charge (Shockley-Ramo theorem)

$Q(t) = -q \cdot [\Phi_w(\mathbf{x}_h(t)) - \Phi_w(\mathbf{x}_e(t))]$

Signal and mirror pulses in segments



Summary and outlook

- Official Julia Package
  - Configurable geometry and drift parameters
  - Fast field calculation (3D; Also simulates environment)
  - Charge drift model takes care of crystal axes effects
  - Segmentation

- Next release (~ Easter)
  - Cartesian Coordinates
  - Geometry Primitives closer to GEANT4

- To do
  - Temperature dependence
  - Charge clouds; diffusion