Utilizing open-source toolkits for the simulation of avalanche formation and space-charge effects in Resistive Cylindrical Chambers

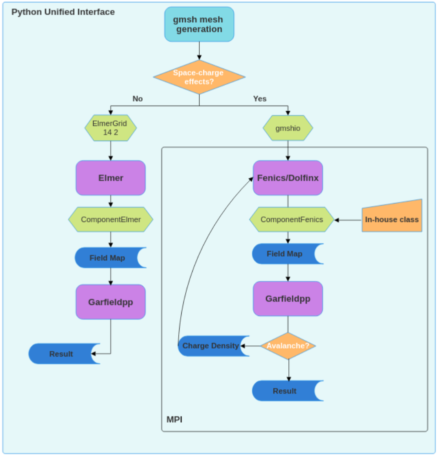
Elton Shumka[1], Peicho Petkov[1], Borislav Pavlov[1], Mihaela Pehlivanova[1], Anton Petrov[1], Leandar Litov[1]

[1]University of Sofia "St. Kliment Ohridski"

Overview

- Workflow overview
- Open-source FEM solvers (Elmer, FEniCS)
- In-house interface class
- Space-charge effects
- MPI parallelization
- Unified Python interface

Workflow overview



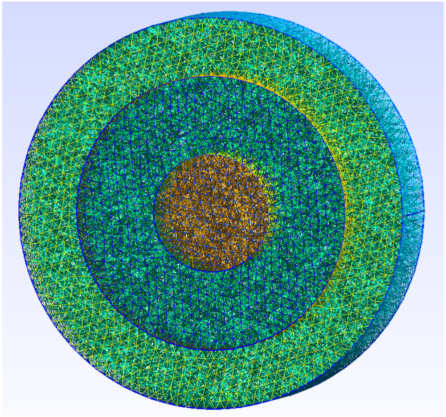
Geometry definition and meshing

gmsh

- Open-source 3D finite element mesh generator
- Constructive solid geometry
- Built around four modules: geometry, mesh, solver and post-processing
- In our workflow, the first two modules are utilized

RCC mesh

- "Hockey Puck" mesh
- Three coaxial cylindrical layers
- Middle layer is the gas gap; the outer layers - the respective electrodes



Finite-Element method

Field calculation alternatives

Open-source packages

- Elmer
- Fenics/dolfinx

RCC signal with Elmer-calculated field

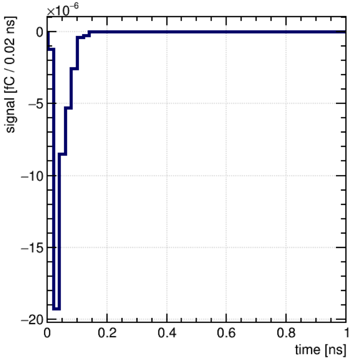


Figure: RCC current

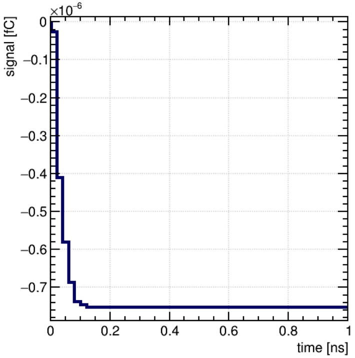


Figure: RCC total charge

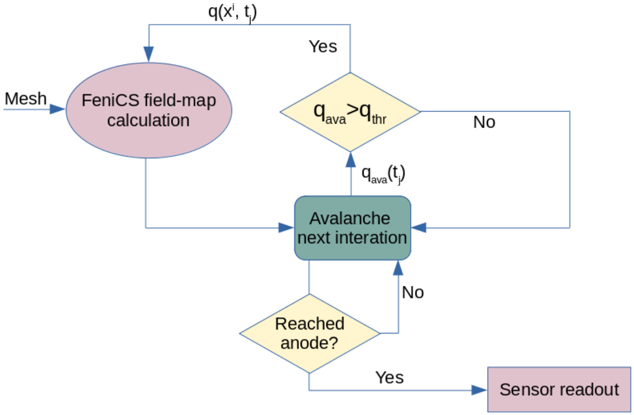
"Space-Charge" effects

Avalanche growth does not follow an exponential rate!

Factors

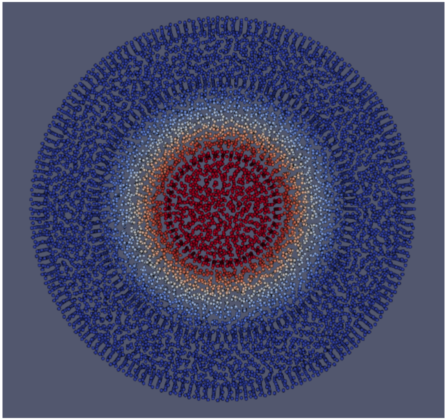
- Ovelap of the electronic and the ionic charge density clouds
- The overlap region is characterized by a reduced electric field, leading to a decrease in the avalanche gain
- This phenomenon is known as the "space-charge effect"

"Space-Charge" effects implementation



Dolfinx field map calculation

- Positive voltage applied on the inner electrode
- Negative voltage applied on the outer electrode
- Periodic Boundary Conditions applied on the gas gap closing disks



Fenicsx/Dolfinx

- Platform for solving PDEs with the Finite Element Method
- Offers a Python API
- Provides an interface to PETSc - offering parallel and distributed (via MPI) solving of PDEs

Poisson equation

$$-\nabla^2 u(x) = f(x), \;\; x \,\epsilon\, \Omega$$

$$u(x) = u_D(x), \;\; x\epsilon \,\partial\, \Omega$$

$$u(x) \rightarrow \text{the unknown function}$$

$$f(x) \rightarrow \text{a prescribed function,}$$

e.g. charge-density map

Solving a Boundary-value problem in FEniCSx

- Identify the computational domain Ω, the PDE, and its corresponding boundary conditions and source terms f
- Reformulate the PDE as a finite element variational problem
- Write a Python program defining the computational domain, the boundary conditions, the variational problem and the source terms, using FEniCSx
- Run the Python program to solve the boundary-value problem. Optionally, you can extend the program to derive quantities such as fluxes and averages, and visualize the results

Finite Element Variational Formulation

The starting point for FEMs is a PDE expressed in a variational form

- Multiply the PDE by a function v
- Integrate the resulting equation over the domain Ω
- Perform integration by parts of those terms with second order derivatives

The function v which multiplies the PDE is called a test function

$$\int_\Omega (-\nabla^2 u) v dx = \int_\Omega f v dx$$

$$\int_\Omega (-\nabla^2 u) v dx = \int_\Omega \partial u \cdot \partial v dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v ds$$

$$\int_\Omega \nabla u \cdot \nabla v \; dx = \int_\Omega f v \; dx$$

FEM solving - how it's done

The problem's weak formulation

Find $u \in V$ such that

$$\int_\Omega \nabla u \cdot \nabla v \; dx = \int_\Omega fv dx \qquad \forall \; v \in \hat{V}$$

For the present problem, the trial and test spaces are defined as:

$$V \;=\; \{v \in H^1(\Omega) | v = u_D \; on \; \partial\Omega\}$$

$$\hat{V} \;=\; \{v \in H^1(\Omega) | v = 0 \; on \; \partial\Omega\}$$

$$\int_\Omega \nabla u_h \cdot \nabla v \; dx = \int_\Omega fv dx \qquad \forall \; v \in \hat{V}_h$$

Garfield++

- CERN toolkit for the simulation of particle detectors based on ionization measurement in gases and semiconductors
- Utilizes Magboltz for the simulation of avalanche formation in different gas mixtures
- Provides interfaces to the finite element packages: Elmer, Comsol etc.
- Interface classes can be extended to include field map outputs from other platforms, such as FEniCS

In-house interface class

class ComponentFEniCS

The class facilitates the importing of FEniCS-produced field maps into Garfield++

- Non-trivial because of the XDMF format of the FEniCS output files, tailored to HPC applications
- Metadata stored separately in .xmdf file, while the data itself stored in .h5 database files
- Pre-treatment of the h5 data to a more human-readable format

Parallel and distributed computation

High Performance Computing

Modern HPC facilities employ all levels of parallelism, combining shared and distributed memory models and utilizing mainly GPGPU accelerator computing power. In this way, users can run highly parallel applications on a large number of cores, gaining the computational performance needed.

Message-Passing Interface (MPI)

The *MPI standard* regulates the message types and communication methods between processes working in parallel with distributed memory. The most popular free implementations of the MPI standard are *OpenMPI* and *MPICH*.
mpi4py provides Python bindings for the MPI system.

Parallel Avalanche Simulation I

Parallelisation strategy

- each process holds its own avalanche object
- evaluation of the avalanche multiplication is discretised in time steps
- at the beginning of each time step, the parallel processes exchange the number of charged objects between them (All-to-All communication). The free charges are exchanged to leverage the computation of the avalanche development
- upon user request, the total avalanche information can be gathered

Parallel Avalanche Simulation II

Parallelisation implementation

- uses Garfield++ *MicroscopicAvalanche*, augmented with methods for adding and removing *Electron* objects from electrons and holes private containers
- a load balancing procedure is responsible for the proper distribution of the electrons and holes among the MPI processes
- both the avalanche charges and the induced signal are additive, so the total quantities are gathered when needed

Parallel Avalanche Simulation - test - HPC infrastructure

Cluster module of Prototype Modular Supercomputer DEEP-EST* @ Juelich Supercomputing Center:

- 50 nodes
- 2 sockets Intel Xeon 6146 (24 cores @ 3.2 GHz)
- 192 GB DDR4 per node
- EDR-IB (100 Gb/s) Fat-tree network

* `https://www.fz-juelich.de/en/ias/jsc/systems/prototype-systems/deep_system`

Parallel Avalanche Simulation - test case

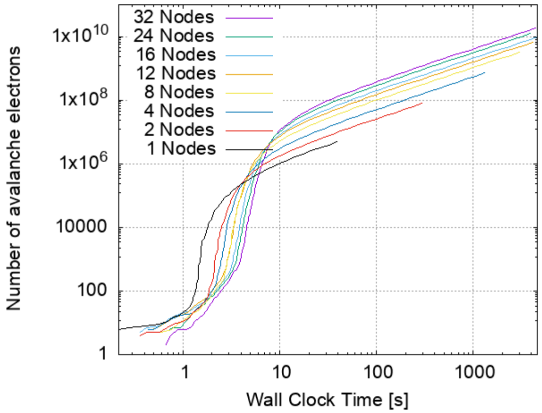
Test case

- Single gap RPC
- 2 mm resistive electrodes, 2 mm gap width
- Gas mixture (90% $C_2H_2F_4$ / 5% iC_4H_{10} / 5% SF_6)
- Electric field 55 kV/mm (it was chosen so as to ensure large electrons number in the gas gap for testing purposes)

Performance test

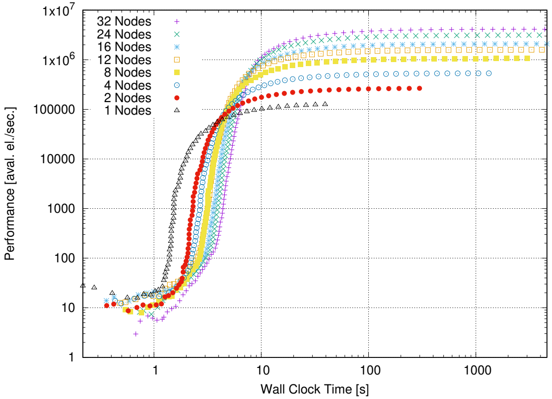
- Performance metrics - Number of avalanche electrons calculated per unit wallclock time
- The simulation performance evaluated on 1, 2, 4, 8, 12, 16, 24 and 32 nodes, number of MPI procs. 24, 48, 96, 192, 288, 384, 576 and 768, respectively.

Parallel Avalanche Simulation - test case results



The simulations stopped due to memory limitations. For testing purposes, every collision point in the electron trajectories was stored

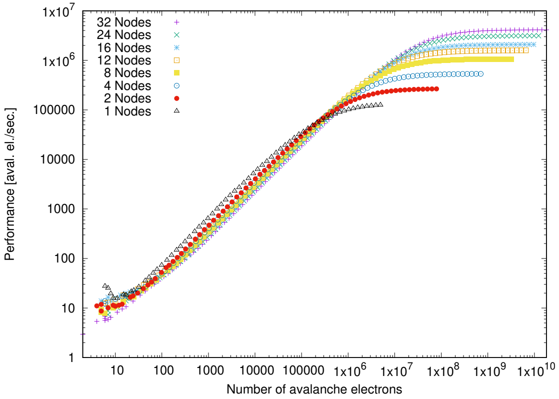
Parallel Avalanche Simulation - test case results



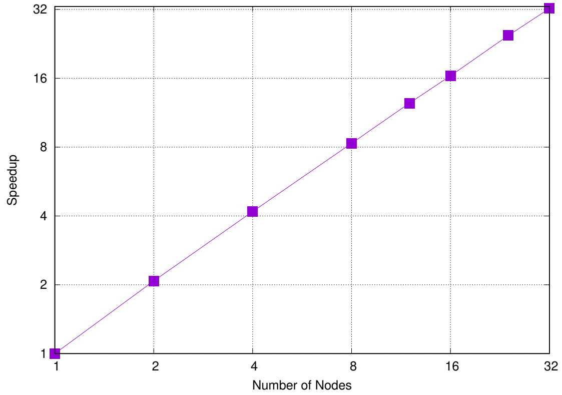
NB

The performance saturates in about a minute

Parallel Avalanche Simulation - test case results



Parallel Avalanche Simulation - Speedup results



The performance scales perfectly with the number of nodes for the performance plateau. Parallel efficiency 100%.

Conclusions

- Development of self-consistent "open source"-based environment for the simulation of gaseous detectors with resistive electrodes that offers gas multiplication calculations and PDEs is feasible with Python
- The combination of Garfield++ and Fenicsx turned out to be the most convenient combination of libraries
- Distributed memory parallelization is crucial for overcoming the memory limitations when accounting for space charge effects.
- The prototype parallel avalanche simulation shows promising results on homogeneous CPU-based HPC machines. The test reached a number of electrons in the avalanche of the order 10^{10} on 768 MPI processes.

Near Future Plans...

- Development of user-friendly object for parallel avalanche simulations
- Development of Fenicsx-Garfield++ interface for potential and charge density exchange
- Development of user-friendly self-consistent procedure for accounting the space-charge effects

Thank You for You Attention!