Numerical simulation of space charge effects in MPGDs

Presented by - MAXIM TITOV

S. DUTTA, P. BHATTACHARYA, T. DEY, N. MAJUMDAR, S. MUKHOPADHYAY

The 8th international conference on Micro-Pattern Gaseous Detectors

USTC, Hefei, China

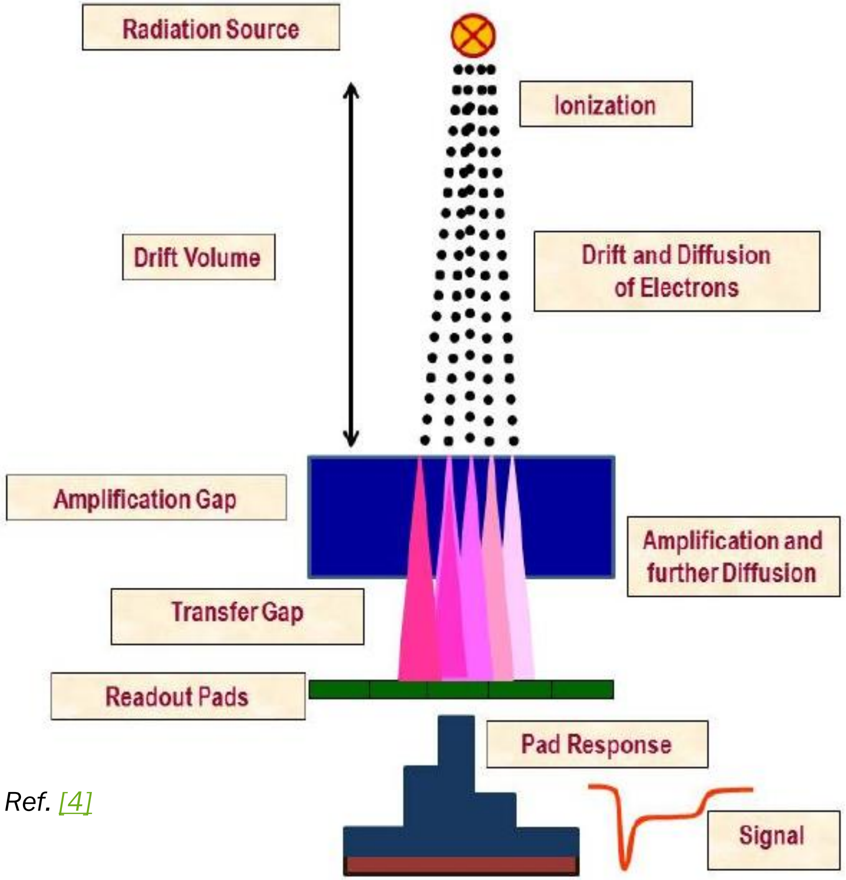
14th - 18th October 2024

17th October, 2024

OUTLINE

- Garfield++
- Electric field solvers: neBEM, among several others
- Integration of CUDA GPU code in neBEM
- Electric field estimation for THGEM
- Initial work on space charge simulation for GEM
- Summary

Garfield++



Ref. [4]

Garfield++: A toolkit for the detailed simulation of particle detectors based on ionisation measurement in gases and semiconductors.

Heed: Provides computed information on interaction of fast charged particles with matter and its ionization.

Magboltz: Calculates the transport parameters of electrons drifting in the gases under the influence of electric and magnetic fields.

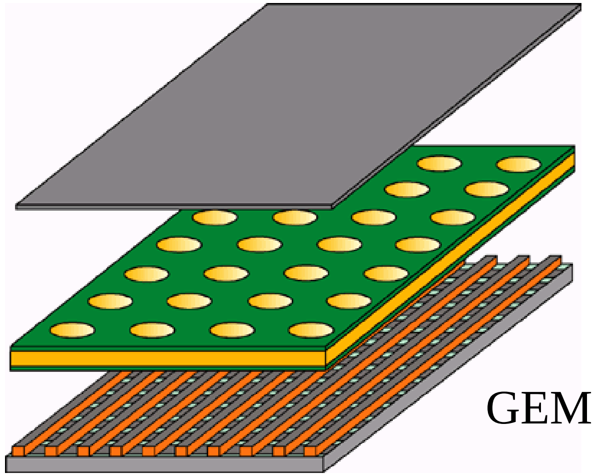
Field Solver: Computes electromagnetic field; interfaces to Garfield++ are available for commercially available FEM packages such as Ansys, Comsol, and open source BEM and FEM packages such as neBEM and Elmer.

- Device dynamics depends crucially on electric field.
- Field solving is very important for MPGDs due to their intricate, essentially 3D geometry.

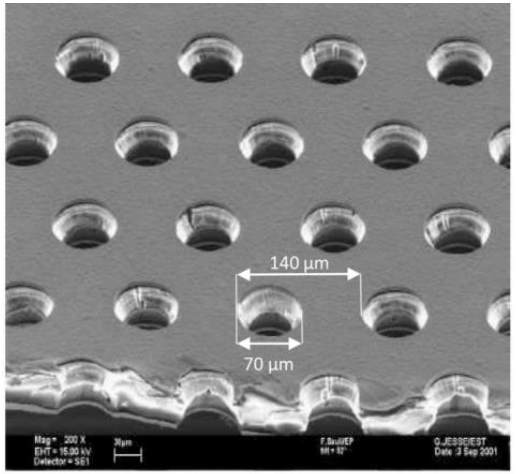
EXPECTED FEATURES OF FIELD SOLVER FOR MPGDS

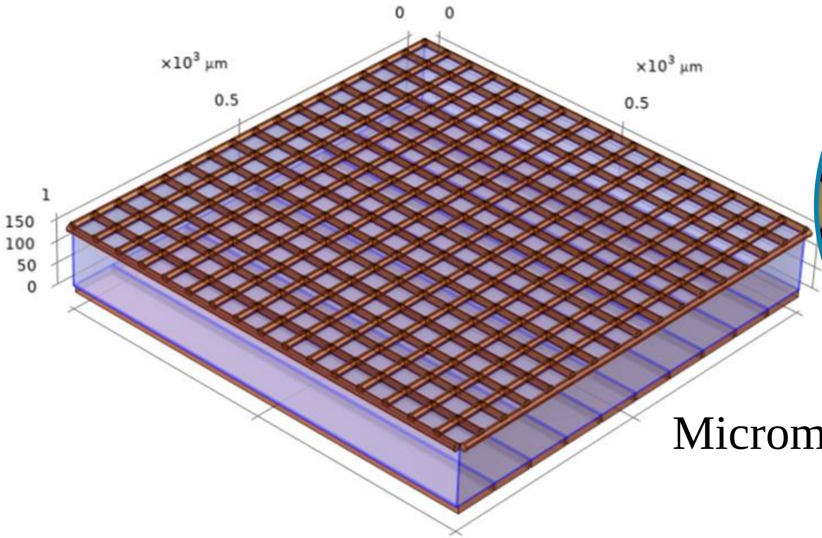
1. Handle large variation in length scale (µm to m).
2. Model intricate geometrical features using triangular elements as and when needed.
3. Reproduce space charge effects and other dynamic charging processes.
4. Model multiple dielectric devices.
5. Model nearly degenerate (closely packed) surfaces
6. Provide fields at arbitrary locations on demand.

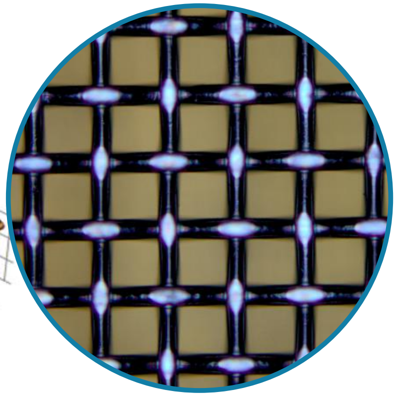
There are various options available: analytical, FEM and BEM, as mentioned earlier.



GEM







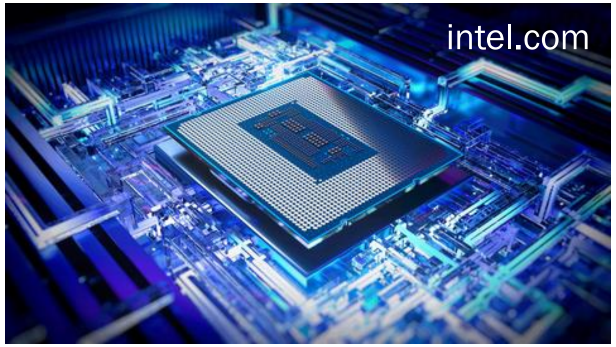
Micromegas

WHY neBEM NEEDS GPUS?

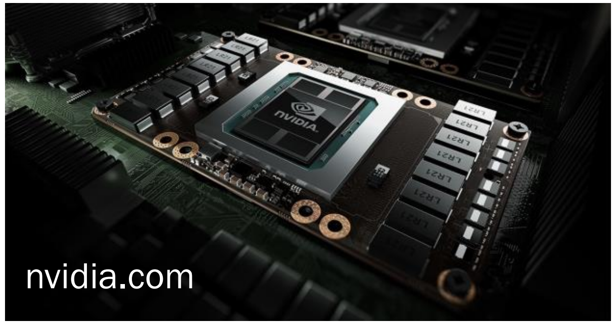
- For complicated and large geometry, the influence coefficient matrix (***A***) that neBEM needs to create, and decompose / invert, can be very large.
- It is computationally expensive to handle such large system of linear equations.
- There exists additional places in the neBEM source code where parallelization is required, such as evaluation of space charge and charging up effects.
- GPU-s can come to the rescue.
- Please note that OpenMP was already implemented several years back and it proved to be very useful.
- We propose to add GPU capabilities to existing OpenMP parallelization.

CPU VS GPU

	CPU	GPU
Function	Generalized component that handles main processing functions of a server	Specialized component that excels at parallel computing
Processing	Designed for serial instruction processing	Designed for parallel instruction processing
Design	Fewer, more powerful cores	More cores than CPUs, but less powerful than CPU cores
Suitable for	General purpose computing applications	High-performance computing applications



13th generation intel CPU



NVIDIA Tesla P100 GPU

aws.amazon.com/compare/the-difference-between-gpus-and-cpus

USED HARDWARE AND SOFTWARE

Workstation 1

- Intel Xeon(R) E5-2698v3
- 2.3 GHz, 64 cores
- 128 GB memory
- Quadro K2200
- Compute Capability : 5.0
- 640 CUDA cores
- 4 GB memory
- Clock Rate : 1124 MHz

CUDA 12.3 with NVIDIA driver 535 running Ubuntu 22.04

- Compiler: nvcc V12.3.107

Workstation 2

- Intel Xeon(R) Gold 6142
- 2.6 GHz, 32 cores
- 64 GB memory
- NVIDIA T1000
- Compute Capability : 7.5
- 896 CUDA cores
- 8 GB memory
- Clock Rate : 1395 MHz

CUDA 12.5 with NVIDIA driver 535 running Ubuntu 22.04

- Compiler: nvcc V12.5.40

HPC Cluster

- Intel Xeon(R) Gold 6140
- 2.3 GHz, 36 cores per node
- 384 GB memory per node
- Tesla V100-PCIE-16GB
- Compute Capability : 7.0
- 5120 CUDA cores
- 16 GB memory
- Clock Rate : 877 MHz

CUDA 12.2 with NVIDIA driver 535 running CentOS 7

- Compiler: nvcc V12.2.128

BUILDING neBEM WITH CUDA SUPPORT

- To install Garfield++ with CUDA [2] option enabled for neBEM, changes has been made while issuing cmake.
- Build separate library for neBEM and link it to the existing Garfield++ target.

```
# Create the NeBem library target
add_library(NeBem INTERFACE)
target_sources(
    NeBem
    INTERFACE
        NeBem/ComputeProperties.cu
        NeBem/Isles.cu
        NeBem/neBEM.cu
        NeBem/neBEMInterface.cu
        .....
)
target_include_directories(NeBem INTERFACE ${CMAKE_CURRENT_SOURCE_DIR}/NeBem)
# Link the NeBem library to the Garfield target
target_link_libraries(Garfield PRIVATE NeBem)
```

```
int bestDevice = 0; // Default to the first device
int maxComputeCapability = 0;
for (int i = 0; i < deviceCount; ++i) {
    cudaDeviceProp prop;
    cudaGetDeviceProperties(&prop, i);
    if (prop.major > maxComputeCapability) {
        maxComputeCapability = prop.major;
        bestDevice = i;
    }
}
cudaSetDevice(bestDevice);
std::cout << "\nGPU device found"<< std::endl;
std::cout <<"Using GPU device: "<< bestDevice <<""<< std::endl;
std::cout << "Running on GPU" << std::endl;
```

Multiple GPU

Best GPU device is searched for and computation proceeds on it. Use of heterogeneous devices will be implemented in future.

DETAILS ON CODE CONVERSION

1. Several new functions have been added to existing neBEM codes that implements CUDA options.
2. Till now the approach is from a coding perspective, not physics. The GPU code uses the same algorithms / techniques as the non GPU version of neBEM.
3. General purpose functions encapsulating matrix-matrix and matrix-vector multiplication kernel have been implemented using cuBLAS library.
4. Predefined cuSolver library functions have been used for both SVD and LU inversion.
5. Custom CUDA kernel is being developed for handling space charges.

```
//Matrix-Matrix Multiplication
Void matMulGPU(double **h_Mat1, double **h_Mat2, double **h_SolutionMat…);
```

```
//Matrix-Vector Multiplication
Void matVecMulGPU(double **h_Mat, double **h_Vec, double **h_SolutionVec…);
```

```
//SVD using cuSolver
void svdcmpcu(double **a, int matrow, int matcol, double *w, double **v);
```

```
//LU using cuSolver
void ludcmpcu(double **a, double **i, int N, int *index, double *d);
```

```
//Known Point Charges
__global__ void PointChGPU(Point3D fieldPt, PointKnCh *d_PointKnChArr, double *d_value);
```

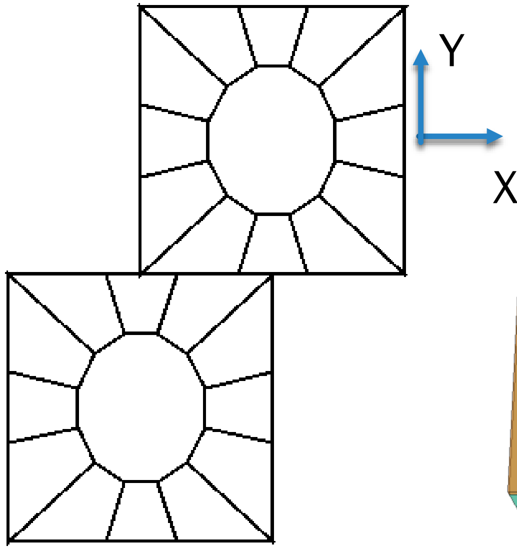
```
//Known Line Charges
__global__ void LineChGPU(Point3D fieldPt, PointKnCh *d_LineKnChArr, double *d_value);
```

USTC HEFEI 2024

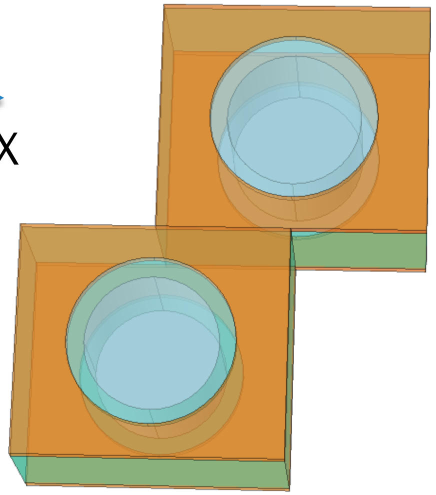
APPLICATION FOR THGEM SIMULATION

- Staggered THGEM geometry has been modelled in neBEM by repeating a unit cell, consisting of two holes. Same approach has been adopted for Comsol Multiphysics also.
- In neBEM, 35 repetitions has been used in both X and Y direction.
- After creation of surface mesh, the THGEM model contains 10622 elements in neBEM.

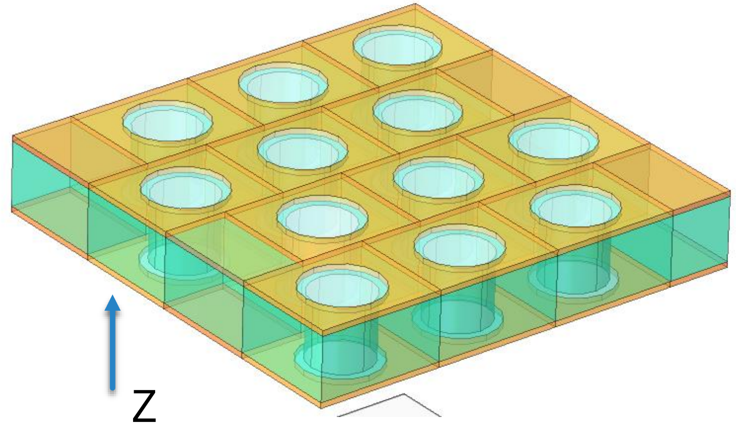
Parameters	Values
Hole diameter	400 μm
Pitch	800 μm
Rim diameter	500 μm
FR4 thickness	400 μm
Copper thickness	35 μm
Electrode thickness	5 μm
Drift gap	1 cm
Induction gap	2 mm
Anode voltage	0 V
Cathode voltage	-4200V
GEM top voltage	-2200 V
GEM bottom voltage	-800 V



Unit cell in neBEM

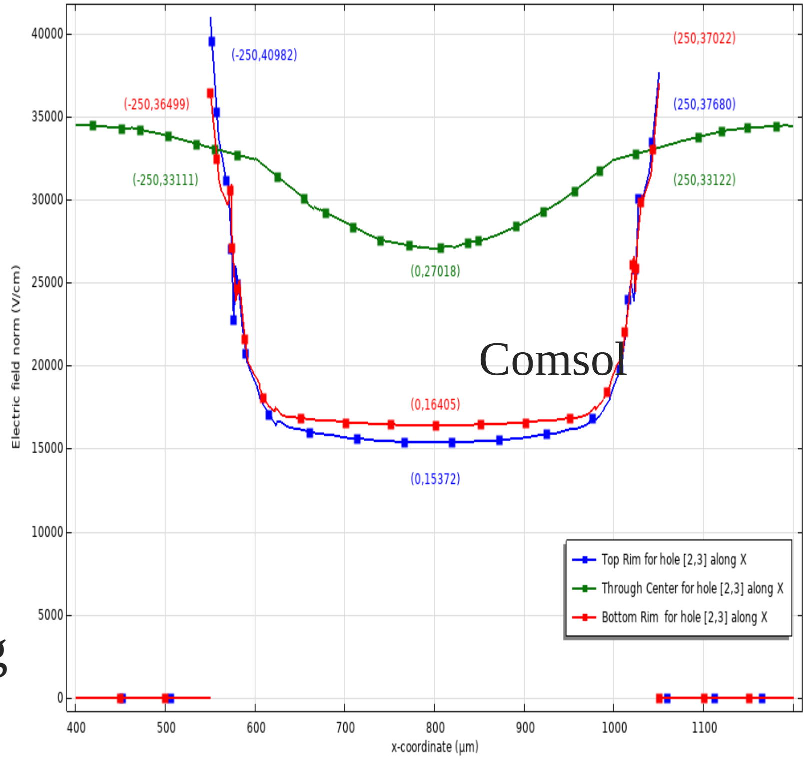


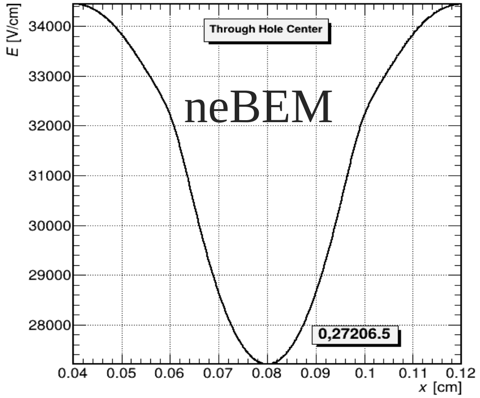
Unit cell repeated in Comsol

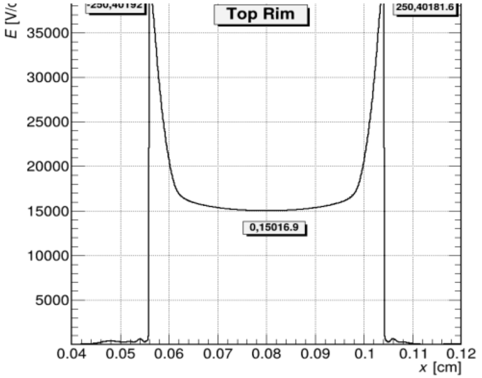


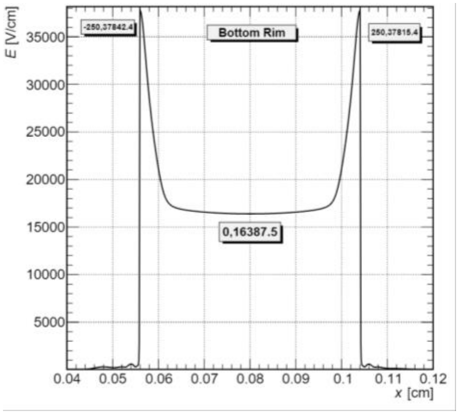
EVALUATION OF ELECTRIC FIELD

1. Field profiling has been carried out with GPU accelerated version of neBEM in Garfield++ and commercially available FEM package COMSOL Multiphysics to check the consistency of the model.
2. Electric field along various lines through the THGEM hole, estimated using Garfield++-neBEM and Comsol, has been compared with excellent agreement among different estimates.









CONCLUSION FROM ELECTRIC FIELD STUDIES

- Despite having used distinctly different mathematical models and computational techniques to solve the problem, it has been observed that the estimates obtained from Comsol and Garfield++-neBEM agree rather well

	X-coordinate	Garfield++ (in V/cm)	COMSOL(in V/cm)
Bottom Rim	-250	37842.4	36499
	0	16387.5	16405
	+250	37815.4	37022
Top Rim	-250	40192	40982
	0	15016	15372
	+250	40181.6	37680
Hole Center	-250	33127	33111
	0	27206	27018
	+250	33124	33122

There is no loss of accuracy due to the use of GPU-enabled neBEM

BENCHMARKING OF EXECUTION TIME: SVD APPROACH

- Can be selected by ***nebem.UseSVDInversion().***
- Workstation 1 experiences an *"out of memory"* error due to its comparatively less GPU memory.
- Workstation 2 and HPC cluster have been used for further benchmarking.

HPC Cluster

# of Threads	Without GPU	With GPU
1	**1503.94 min**	280.28 min
2	657.68 min	146.23 min
4	272.31 min	75.17 min
8	152.53 min	40.39 min
16	**107.73 min**	**21.26 min**

- On the HPC cluster, the speedup using only OpenMP is ~ 14 times (1504 / 108).
- OpenMP + GPU gives ~72 times speedup!

Workstation 2

# of Threads	Without GPU	With GPU
1	**1588.02 min**	361.43 min
2	691.82 min	195.43 min
4	294.61 min	105.02 min
8	147.26 min	56.69 min
16	**138.79 min**	**33.27 min**

- On Workstation2, the speedup using only OpenMP is ~ 11 times (1588 / 139).
- OpenMP + GPU gives ~50 times speedup!

BENCHMARKING OF EXECUTION TIME: LU APPROACH

- One can choose by ***nebem.UseLUInversion().***
- Much faster than SVD approach when GPU parallelization is not switched on.
- Reasonably faster in comparison to SVD with GPU parallelization.

HPC Cluster

# of Threads	Without GPU	With GPU
1	**361.95 min**	268.85 min
2	190.463 min	144.34 min
4	101.68 min	70.93 min
8	64.66 min	37.33 min
16	**59.49 min**	**19.36 min**

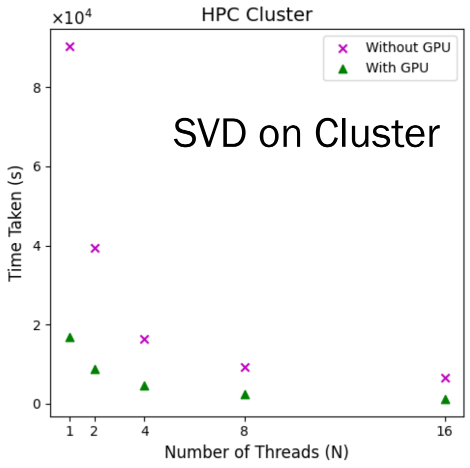
- ➢ On the HPC cluster, the speedup using only OpenMP is ~ 6 times (362 / 59).
- ➢ OpenMP + GPU gives ~19 times speedup!

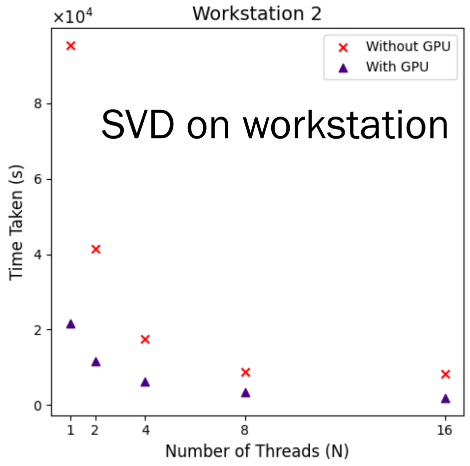
Workstation 2

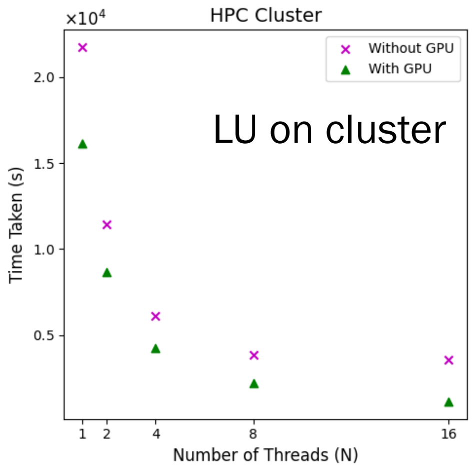
# of Threads	Without GPU	With GPU
1	**411.33 min**	335.65 min
2	210.61 min	178.42 min
4	110.69 min	93.12 min
8	61.79 min	47.36 min
16	**45.29 min**	**25.32 min**

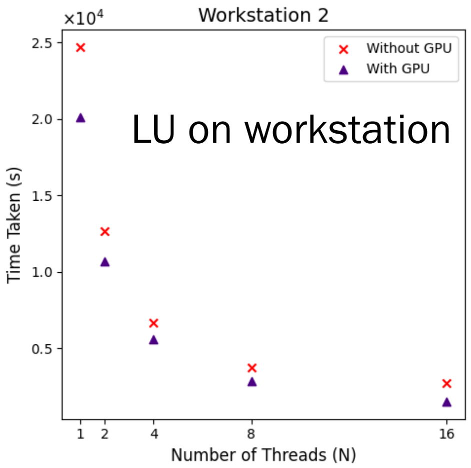
- ➢ On the Workstation 2, the speedup using only OpenMP is ~ 9 times (411 / 45).
- ➢ OpenMP + GPU gives ~16 times speedup!

BENCHMARKING SVD AND LU APPROACHES





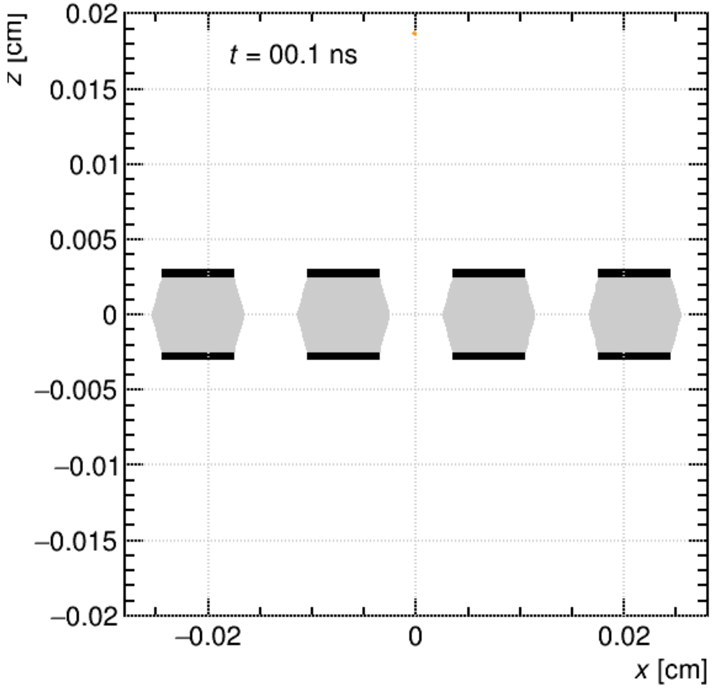




- Increase in number of threads in OpenMP leads to faster computation.
- In OpenMP multi-threading, number of threads ranging between 8 to 16 seems to be optimum.
- Application of GPU parallelization reduces time taken to compute, quite considerably.

PARALLELIZATION OF SPACE CHARGE EFFECTS

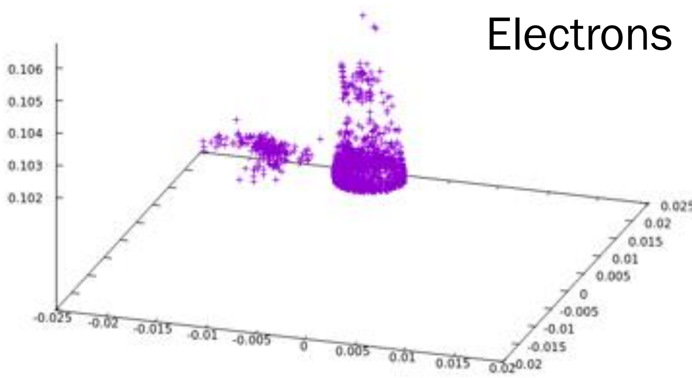
- Space charge effects are known to significantly influence the response of MPGDs.
- A large number of charge particles ($\sim 10^5 - 10^9$) are involved in the process.
- Conceptually simple but computationally very expensive to implement.
- Extensive use of parallelization needed in the neBEM solver.
- CUDA has been implemented for handling known point charges and line charges.
- Benchmarking process is ongoing.

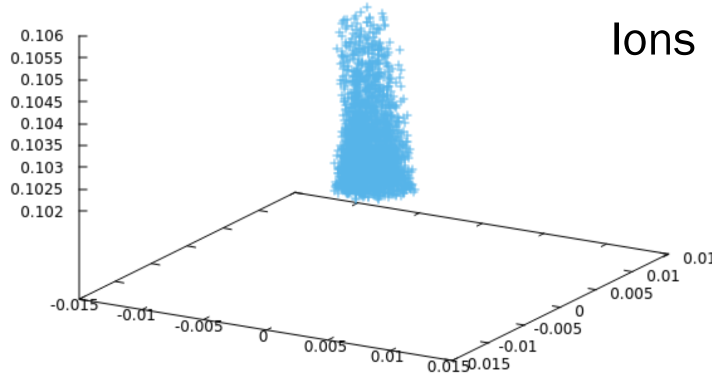


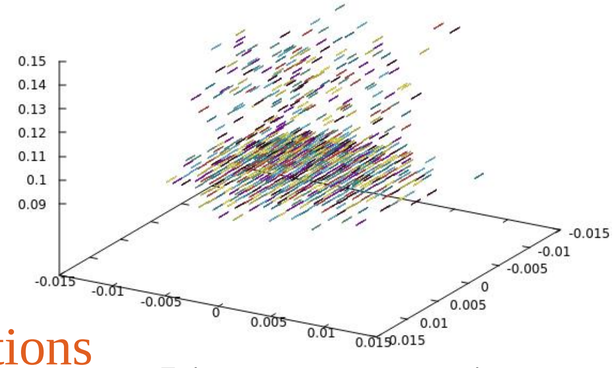
SPACE CHARGE IN GEM –PARTICLE MODEL

Material borrowed from the *Ageing and Stability Conference 2023* presentation of *P. Bhattacharya*

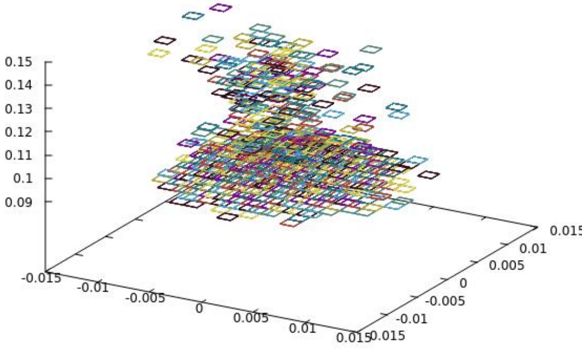
- Existing models to represent space charge
  - point
  - line
  - ring
  - area
  - Volume (PIC)
- Future plans
  - disc
  - volume (integrations being tried)



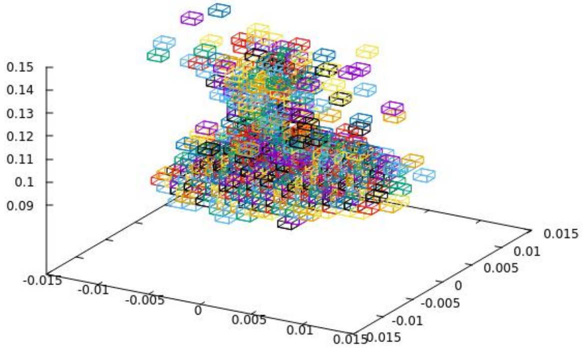




Line representation



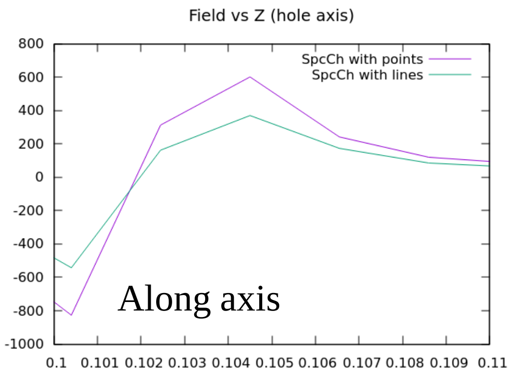
Area representation

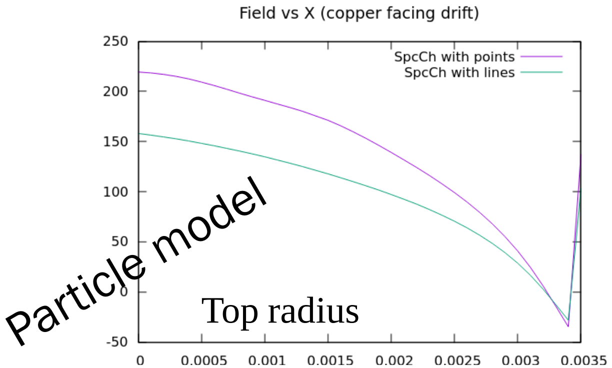


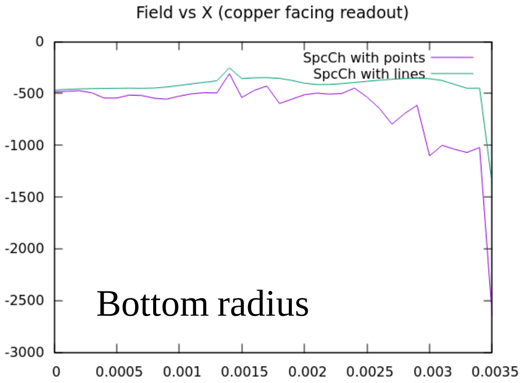
Volume representation

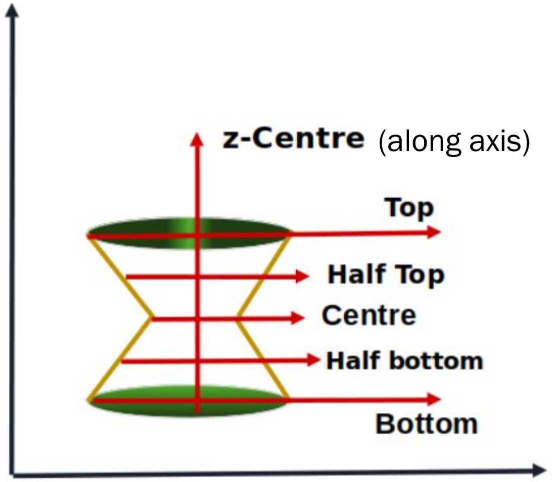
Several representations of space charge are already implemented.

FIELD DISTORTIONS DUE TO SPACE CHARGE EFFECTS



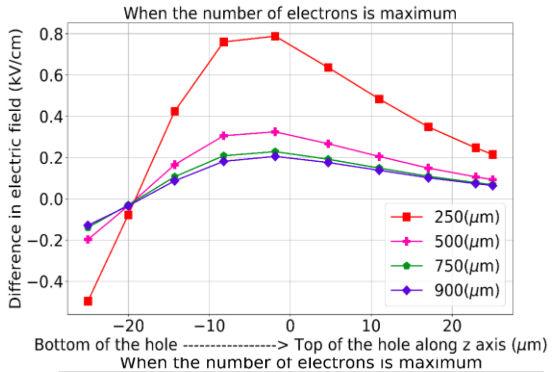


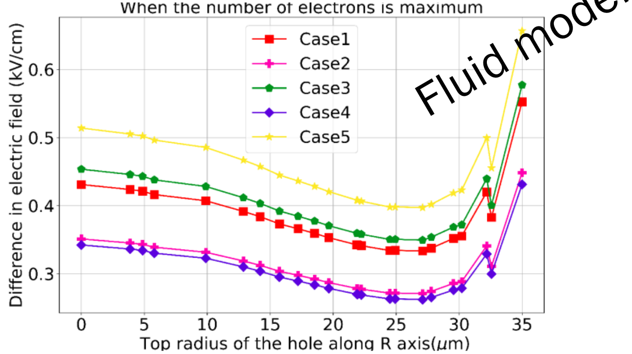


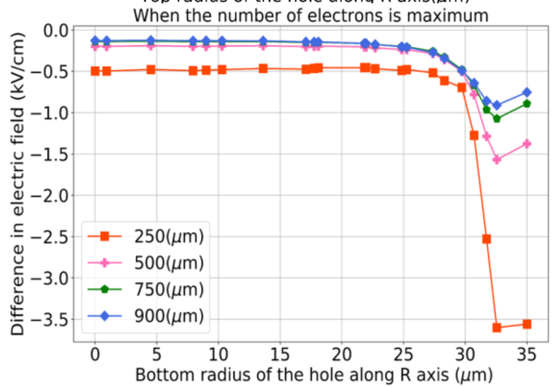


Field values are compared for lines indicated above (not all are shown)

Good agreement is observed among results obtained using particle and fluid models







SUMMARY

- We have successfully implemented CUDA in neBEM. It will be useful for solving large complex realistic model in much less time.
- The GPU version is much faster than the already existing CPU versions of neBEM, without any compromise in accuracy. We could achieve ~ 3 - 7x speed for GPU-accelerated versions in HPC cluster and ~ 2 - 4x speed in average workstation, on top of ~10x speedup due to multi-threading.
- Optimization and profiling to identify remaining bottlenecks is necessary. There appears to be scope for increasing this performance gain even further.
- Work on implementation and benchmarking of GPU-based space charge and charging up simulation are in progress.
- Successfully estimated of space charge effects with Garfield++-neBEM (without parallelization). Results agree well with Comsol fluid simulation.
- We hope to have this entire work integrated into the Garfield++ code-base soon enough.

REFERENCES

[1] Garfield++ [https://gitlab.cern.ch/garfield/garfieldpp]

[2] NVIDIA CUDA [https://developer.nvidia.com/cuda-toolkit]

[3] H. Schindler, Garfield++ User' s Guide [https://garfieldpp.web.cern.ch/garfieldpp]

[4] "Experimental and Numerical Investigation on Micro-Pattern Gas Detectors".Thesis by Purba Bhattacharya (December 2014), Department of Physics University of Calcutta, Kolkata, India

[5] Computation of 3D electrostatic weighting field in Resistive Plate Chambers, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 595 (2) (2008) 346–352. doi: https://doi.org/10.1016/j.nima.2008.07.033.

[6] COMSOL Multiphysics [https://www.comsol.co.in/.]

[7] OpenMP [https://www.openmp.org/.]

[8] T. Dey, S. Mukhopadhyay, S. Chattopadhyay, Numerical study of effects of electrode parameters and image charge on the electric field configuration of RPCs, Journal of Instrumentation 17 (04) (2022) P04015. doi:10.1088/1748-0221/17/04/p04015.

[9] Tanay Dey, Purba Bhattacharya, Supratik Mukhopadhyay, Nayana Majumdar, Abhishek Seal, Subhasis Chattopadhyay, Parallelization of Garfield++ and neBEM to simulate space-charge effects in RPCs, Computer Physics Communications, Volume 294, 2024, 108944, ISSN 0010-4655, https://doi.org/10.1016/j.cpc.2023.108944.

[10] "Boundary Element Method" Lecture given by Prof. Supratik Mukhopadhyay, RD51 Collaboration Meeting, 22-26 June 2020.

[11] I. B. Smirnov, "Modeling of ionization produced by fast charged particles in gas"; Nucl. Instrum. Meth. A vol. 554 (2005) 474. (online at http://cern.ch/heed)

[12] David B. Kirk, Wen-mei , "W. Hwu Programming Massively Parallel Processors: A Hands-on Approach"

[13] Jason Sanders , Edward Kandrot "CUDA by Example: An Introduction to General-Purpose GPU Programming"

[14] S. Biagi; "Magboltz - Transport of electrons in gas mixture"; online at http://cern.ch/magboltz

ACKNOWLEDGEMENT

- A very special thanks to Dr. Maxim Titov
  - – for presenting this work on behalf of our group.
- The Organisers
  - – for their unflinching support under all circumstances.
- Collaborators from RD51 / DRD1 collaborations
  - – for constructive suggestions / help, especially from Rob and Heinrich.
- Lab Mates : Subhendu Das, Pralay Kumar Das and Saikat Ghosh
  - – for their continuous help and encouragement.
- SINP, HBNI, DAE, Adamas University, SERB Project No. SRG/2022/000531
  - – for providing the necessary funding and equipment support throughout this work.

Thank You

Backup slides

VARIOUS APPROACHES TO COMPUTE ELECTRIC FIELD

- Direct measurement of the electric field inside MPGDs is a difficult task to perform.
- Indirect measurement relies upon the gas discharge process using spectral analysis.
- It becomes necessary to rely on analytical or numerical methods.

Analytical

- Accurate for simple geometries (e.g., wire chambers).
- Effective in 2D / axisymmetric geometries.
- Not suitable for complex 3D geometries.
- Inadequate for real detectors with imperfections.

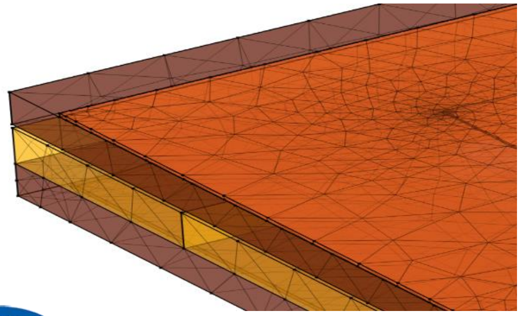
FEM

- Solves Laplace's equation at nodal points within the discretized volume.
- Interpolates / extrapolates potential / field values at non-nodal points.

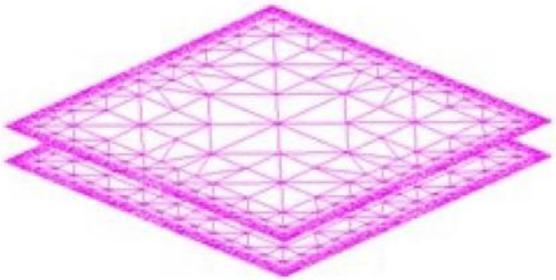
BEM

- Solves boundary integral equations from Poisson's equation.
- Evaluates potential/field based on charge distribution on boundaries.

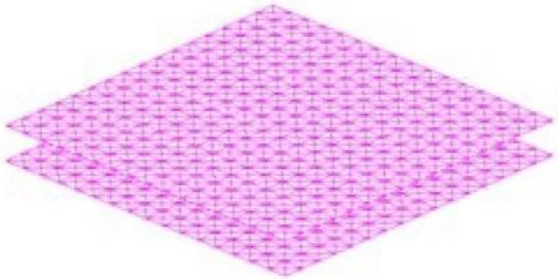
Structured volume mesh in FEM



Adapted surface mesh in BEM



Uniform surface mesh in BEM



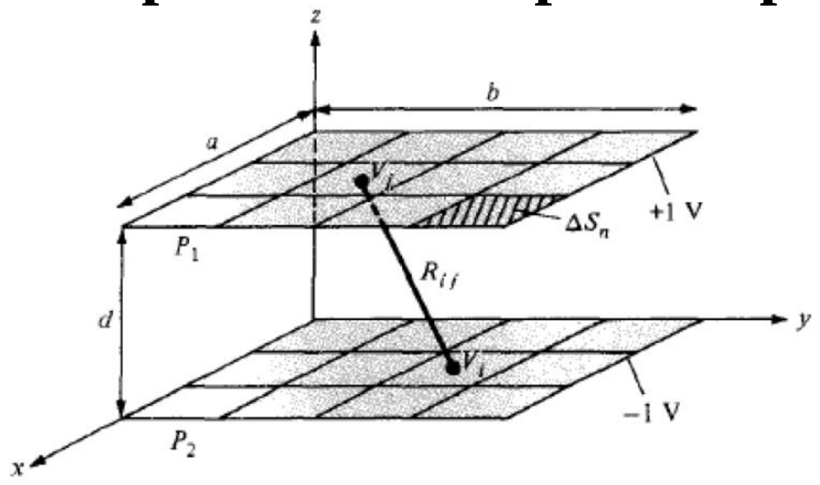
USTC HEFEI 2024

CONVENTIONAL BEM

❖ **Capacitance of a parallel plate capacitor :**

Step 1 : Get total charge Q.

Step 2 : Calculate $C = \frac{Q}{V} = \frac{Q}{2.0}$



Plates are raised to +1 and -1 volts

$$A_{ij} = \frac{1}{4\pi\epsilon_0}\int_{\Delta S_j}\frac{dS}{R_{ij}} \quad for\ i \neq j$$

Analytical expression for i = j exists for few useful shapes!

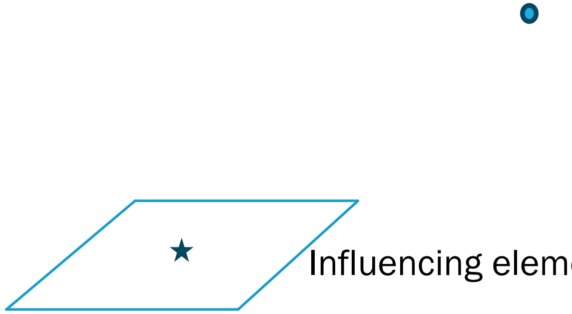
$$V_i = \frac{1}{4\pi\epsilon_0}\sum_{j=1}^{2n}\int_{\Delta S_j}\frac{\rho_j dS}{R_{ij}} \rightarrow V_i = \sum_{j=1}^{2n}\rho_j A_{ij} \rightarrow A\,\rho = V$$

$$\rho = A^{-1}\,V$$

- All we need to know is the influence coefficient matrix.
- Depends entirely on geometry and material budget.
- Weighting field calculation for signal generation is easy.
- Potential and field at any point can be calculated.

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1,2n} \\ A_{21} & A_{22} & \cdots & A_{2,2n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ A_{2n,1} & A_{2n,2} & \cdots & A_{2n,2n} \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \cdots \\ \cdots \\ \rho_{2n} \end{bmatrix} = \begin{bmatrix} +1 \\ +1 \\ \cdots \\ -1 \\ -1 \end{bmatrix}$$

CHALLENGES WITH CONVENTIONAL BEM



The conventional BEM approach has a few serious drawbacks

- **Major assumptions:**
  - Surface charges assumed to be concentrated at nodal points rather than distributed across elements.
  - Boundary conditions satisfied at predetermined points, not across the entire element.
- **Near-field inaccuracies:**
  - Estimation of potential and field near boundaries and interfaces become erroneous.
  - Complications with closely spaced surfaces, edges, corners, and geometric singularities.

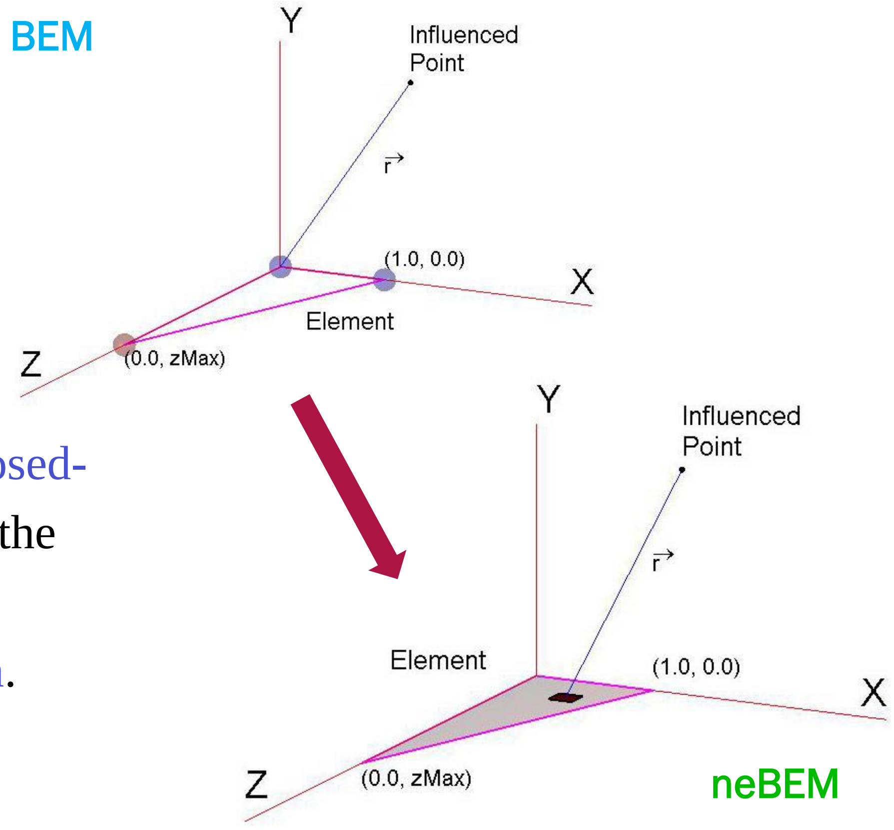
- Effect of the influencing element on the field point (●) is usually sought for.
- Charge distributed over the influencing element is assumed to be concentrated at the star (★).
- The boundary condition is also satisfied at the same point.
- This is the so-called collocation point.
- For self-influence, analytical expressions exist for few element shapes (rectangular, triangular) so that zero distance is well taken care of.

- To address the mentioned issues with BEM, neBEM (nearly exact Boundary Element Method) was introduced.
- Accounts for true charge density distribution on elements, shown in the next slide .

ADVANTAGES AND DISADVANTAGES OF neBEM

- **Improved Accuracy:**
  - Accurate across the entire physical domain, including near-field regions.
  - No need for specific formulations for different sections of the domain.
- **Enhanced Solutions:**
  - Effectively addresses problems involving complex geometries and closely spaced surfaces.

Ref. : https://gitlab.cern.ch/garfield/garfieldpp/-/tree/master/NeBem



- Major drawback of neBEM stems from the large number of complex closed-form analytic expressions employed to evaluate potential and field, and the usual BEM menace of fully populated influence matrix.
- Excellent accuracy is achieved at the cost of painfully slow computation.
- This was alleviated to a certain extent by implementing OpenMP [4].
- In this work, we present implementation of GPU capabilities.

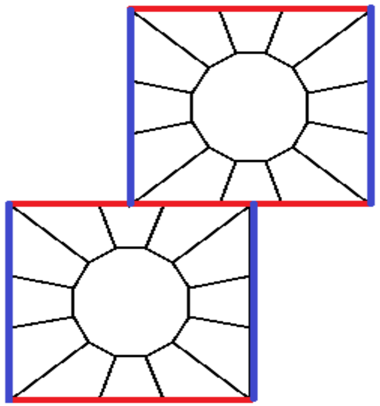
THGEM MODEL – A REALISTIC SCENARIO

- For this study a staggered model of THGEM has been used (similar to the experimental setup currently being pursued at the SINP laboratory).
- Field profiling has been carried out with GPU accelerated version of neBEM in Garfield++ and commercially available FEM package COMSOL Multiphysics to check the consistency of the model.
- Chrono library functions has been used to time the computation for solving the model and calculating electric field and potential value at a predefined point of the model.
- A study comparing the execution times of CUDA with OpenMP implementations has been performed for both Singular Value Decomposition (SVD) and Lower Upper Decomposition (LU) solution approach.

PREREQUISITES FOR LU APPROACH : RMPRIM

- One can choose by ***nebem.UseLUInversion().*** Needed some extra preparation for this type of model.
- To get correct solution using LU **some overlapping primitives needed to be removed** (in this model 7 primitives , marked with **Blue** and **Red** lines) by using ***nebem.SetOptRmPrim(1)*** option.
- The primitives to be removed should be listed on the neBEM input file ***"neBEMInp/neBEMRmPrim.inp".***
- One can define the primitive by indicating **a normal to it** and **a point** on the plane.

Primitives needed to be removed

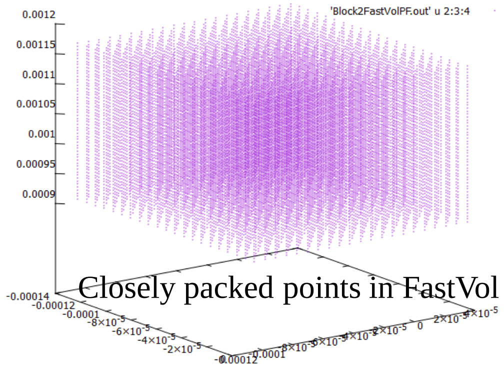


First few lines of *"neBEMRmPrim.inp"* file

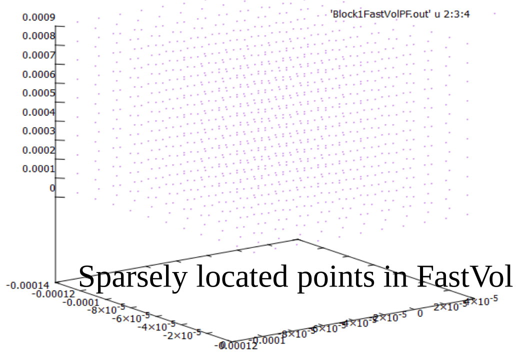
```
NbRmPrims: 7
Prim: 1
rmXNorm: 1.0
rmYNorm: 0.0
rmZNorm: 0.0
rmXVert: -400.0e-6
rmYVert: 0.0
rmZVert: 0.0
Prim: 2
rmXNorm: 1.0
rmYNorm: 0.0
rmZNorm: 0.0
rmXVert: 0.0e-6
rmYVert: 0.0
rmZVert: 0.0
```

PREREQUISITE FOR AVALANCHE : FastVol

- Simulation of avalanches, signals at pickup electrodes, effects of space charge, charging up, and similar other such phenomena, necessitates evaluation of physical and weighting, potential and field, at a large number of spatial locations and temporal instances.
- This turns out to be computationally very expensive. Hence, the necessity of a precomputed map of potential and field – the so-called, FastVol.
- Both physical and weighting (several of the latter, if we have a number of electrodes on which need to see the signal) potential and field are stored on a cartesian 3D map.
- Trilinear interpolation is used to find out these properties at any arbitrary point.



Closely packed points in FastVol



Sparsely located points in FastVol

```
OptFastVol: 1
OptStaggerFastVol: 1
OptCreateFastPF: 1
OptReadFastPF: 1
NbPtSkip: 0
NbStgPtSkip: 0
LX: 140.0e-6
LY: 140.0e-6
LZ: 2560.0e-6
CornerX: -140.0e-6
CornerY: -105.0e-6
CornerZ: 0.0e-6
YStagger: 70.0e-6
NbOfBlocks: 3
```

Example input file of a FastVol related to a single GEM