Rapid Parametric Studies of Polywell Electron Injection

Devlin Baker
Convergent Scientific, Inc
d.baker@convsci.com

Outline

> Challenges for PIC Simulation of IEC / Polywell

> CSI's SimCode Algorithm

    > Dynamic Particle Sort (DSP)

    > Fast particle-grid and particle-geometry intersection

    > Load balanced movers

    > Hybrid-grid field solvers

> Cloud-based Batch Computing: “PlasmaCloud”

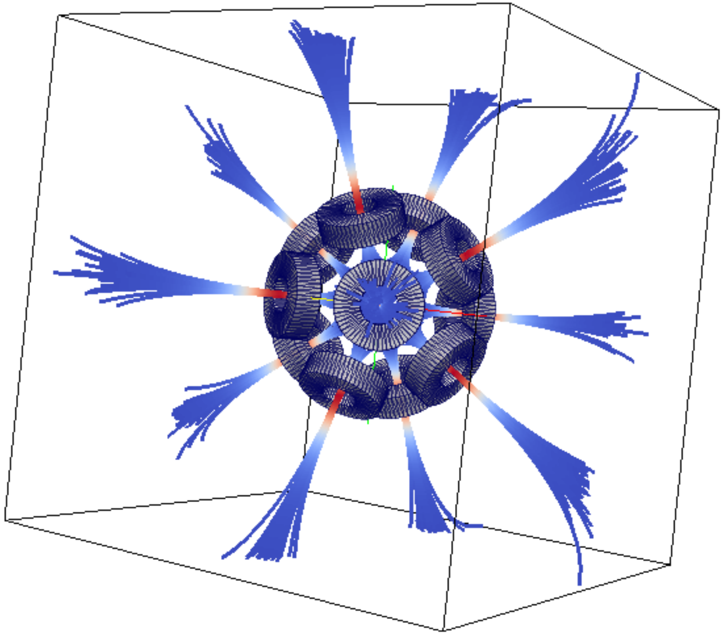
> Polywell Electron Injection Study

    > Setup

    > Effect of angular velocity spread

    > Lessons

> Future Work



Challenges for PIC Simulation of IEC

> Small Signal/Noise ratio for well formation

> Resolving disparate time scales

> Field Solvers must handle complex geometry

> Particle mover must handle (complex) geometry intersection

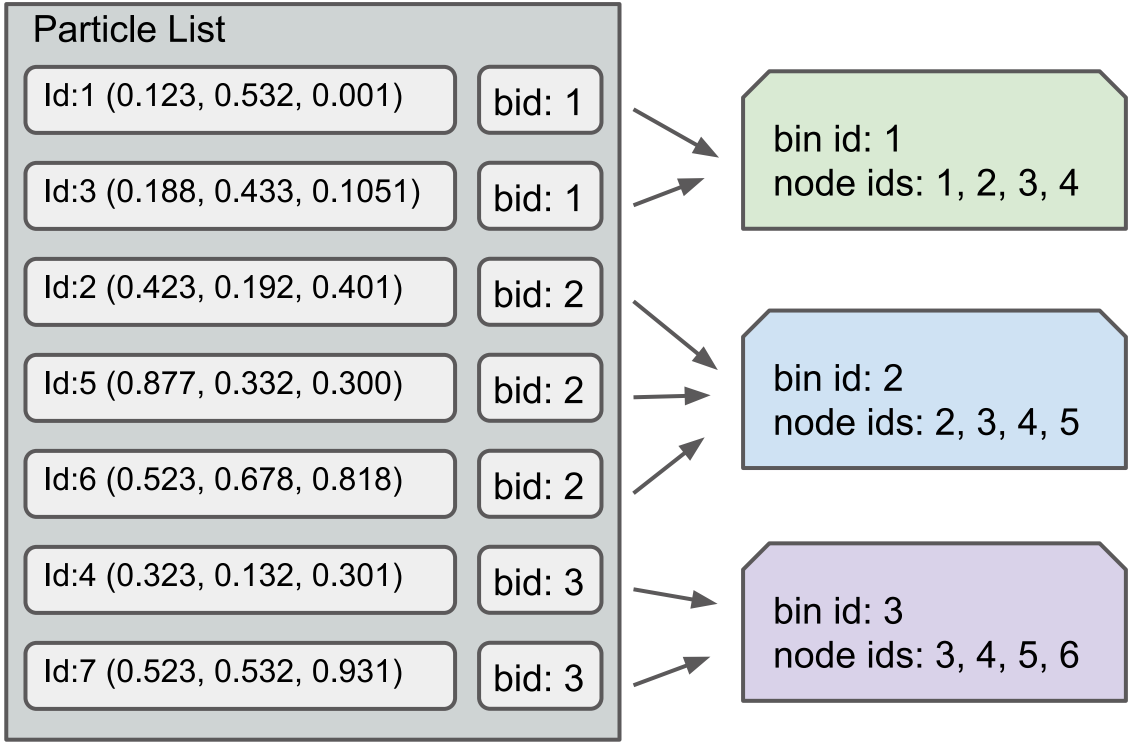
Solutions for Efficient PIC Sim of IEC

- Very large particle counts
- Load-balanced and/or adaptive movers
- Partitioned/Octree geometry intersection
- Hybrid-FEM field solvers

GP-GPU Based "SimCode"

(suggestions for better name welcomed)

SimCode- Sorted Charge Deposit

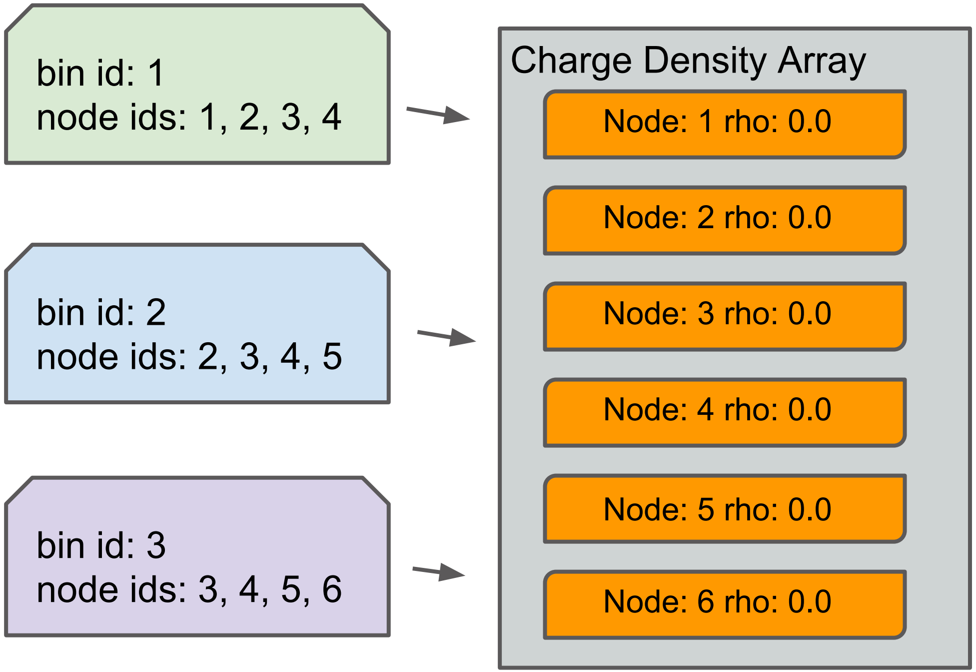


1. Deposit per bin (thread block) into shared memory

SimCode- Sorted Charge Deposit

2. Copy from shared to global array

Potential memory collision rate reduced from square of particle count to function of bin count.



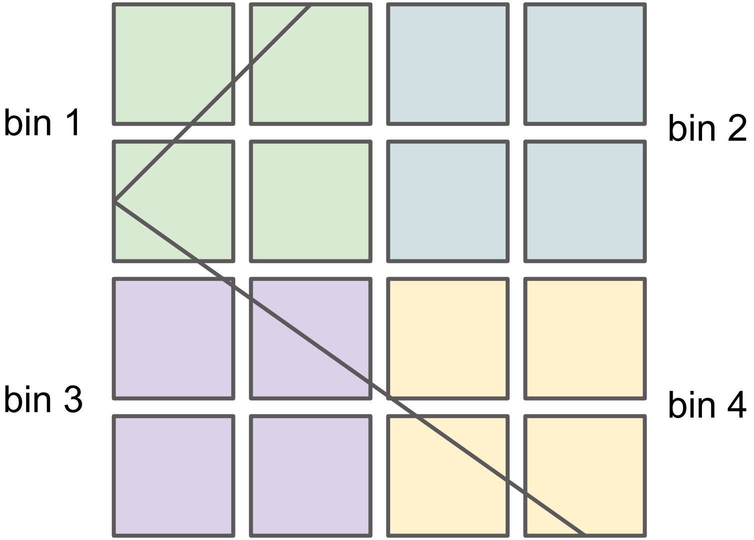
SimCode Algorithm- Particle/Geometry Intersections

Particle bins → Geometry bins

bin id: 1
Geometry: line 1, 2

bin id: 3
Geometry: line 2

bin id: 4
Geometry: line 2



Calculate intersections with geometry per bin (block), flag collisions

SimCode Algorithm- Load Balanced Movers

Goal: Fast Memory Access

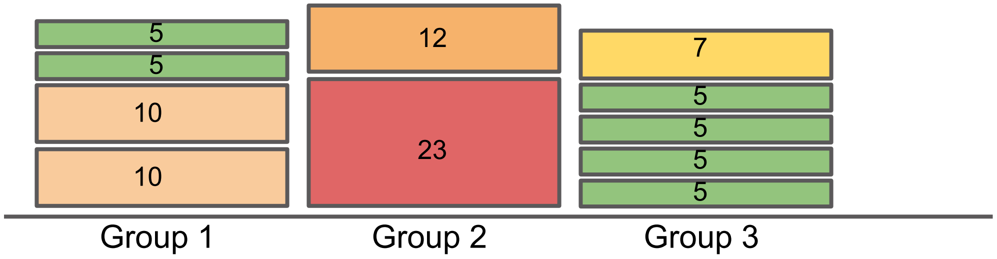
Use series of flattened component arrays in Texture Memory for field quantities.

$\vec{E}$ ⇨ int ind = i + j*dx + k*dy*dx
Ex[ind], Ey[ind], Ez[ind]

$\vec{B}$ ⇨ Bx[ind], By[ind], Bz[ind]

Goal: Approximately equal load on each SIMD core

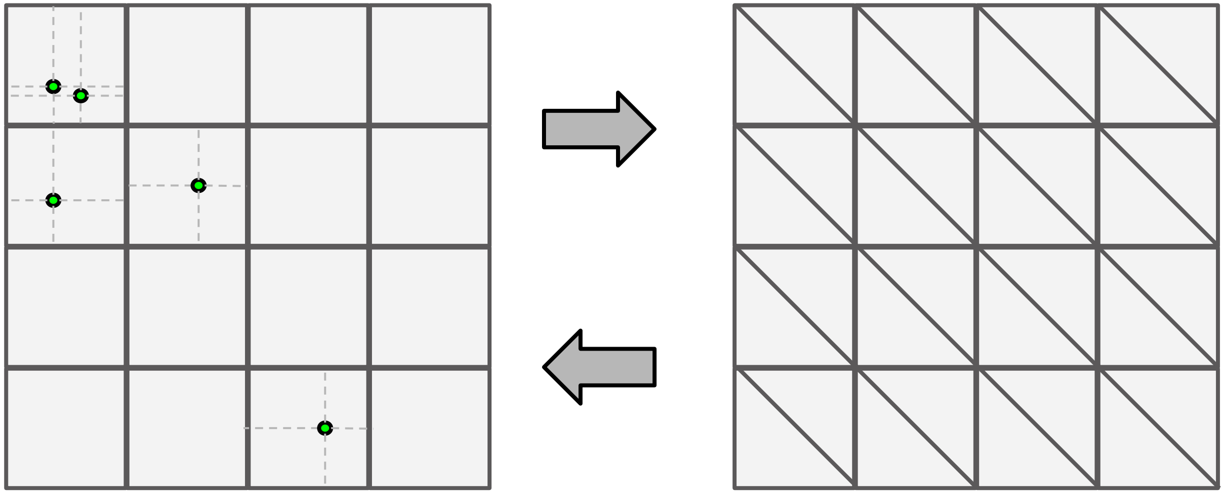
For sub-stepped movers, assign particles to thread block based on expected number of sub-steps (proportional to error between first and second order solution).



SimCode Algorithm- Hybrid Grid Field Solvers

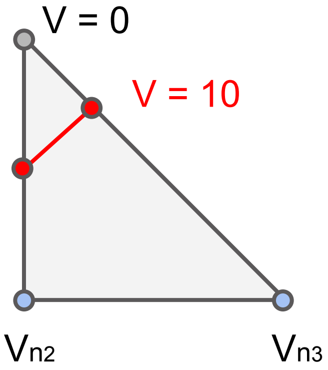
Particle quantities on rectilinear grid

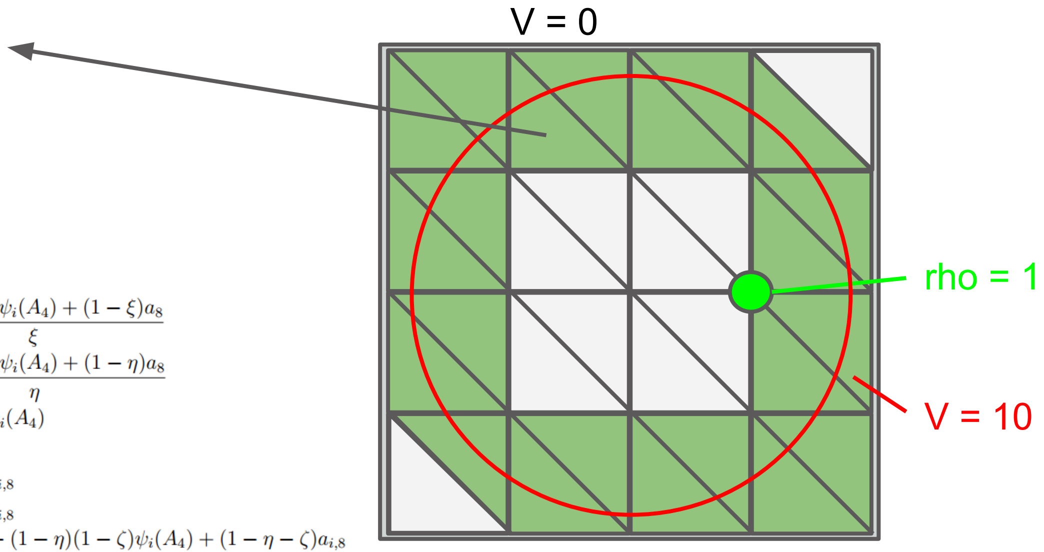
Field solve on rectilinear-based triangular (or tetrahedral) mesh



Based on: Kafafy, R.; Wang, J., "A Hybrid Grid Immersed Finite Element Particle-in-Cell Algorithm for Modeling Spacecraft-Plasma Interactions," *Plasma Science, IEEE Transactions on* , vol.34, no.5, pp.2114,2124, Oct. 2006

SimCode Algorithm- Immersed Boundary Method



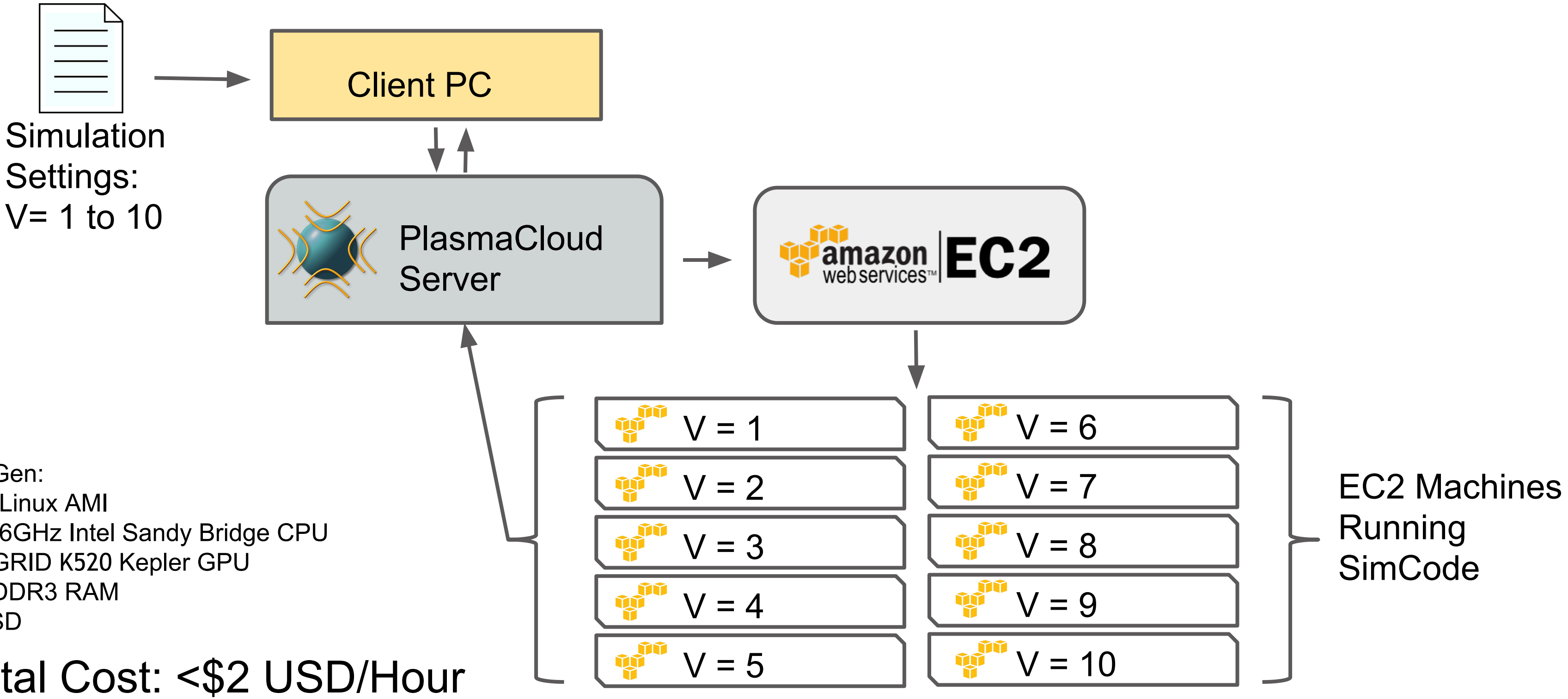


Basis Functions of the Form:

$$\begin{aligned}
a_{i,1} &= \frac{\xi\psi_i(A_1) - \psi_i(A_4) + (1-\xi)a_{i,8}}{\xi} \\
a_{i,2} &= \frac{\eta\psi_i(A_2) - \psi_i(A_4) + (1-\eta)a_{i,8}}{\eta} \\
a_{i,3} &= \frac{\zeta\psi_i(A_3) - \psi_i(A_4) + (1-\zeta)a_{i,8}}{\zeta} \\
a_{i,4} &= \psi_i(A_4) \\
a_{i,5} &= \psi_i(A_1) - a_{i,8} \\
a_{i,6} &= \psi_i(A_2) - a_{i,8} \\
a_{i,7} &= \psi_i(A_3) - a_{i,8} \\
a_{i,8} &= \frac{\lambda_{i,1}\beta^- + \lambda_{i,2}\beta^+}{\lambda_3\beta^- + \lambda_4\beta^+},
\end{aligned}$$

$$\begin{aligned}
a_{i,1} &= \frac{\xi\psi_i(A_1) - \psi_i(A_4) + (1-\xi)a_8}{\xi} \\
a_{i,2} &= \frac{\eta\psi_i(A_2) - \psi_i(A_4) + (1-\eta)a_8}{\eta} \\
a_{i,3} &= \psi_i(A_3) - \psi_i(A_4) \\
a_{i,4} &= \psi_i(A_4) \\
a_{i,5} &= \psi_i(A_1) - a_{i,8} \\
a_{i,6} &= \psi_i(A_2) - a_{i,8} \\
a_{i,7} &= \frac{\eta\zeta\psi_i(A_3) - (1-\eta)(1-\zeta)\psi_i(A_4) + (1-\eta-\zeta)a_{i,8}}{\eta\zeta} \\
a_{i,8} &= \frac{\lambda_{i,1}\beta^- + \lambda_{i,2}\beta^+}{\lambda_3\beta^- + \lambda_4\beta^+}.
\end{aligned}$$

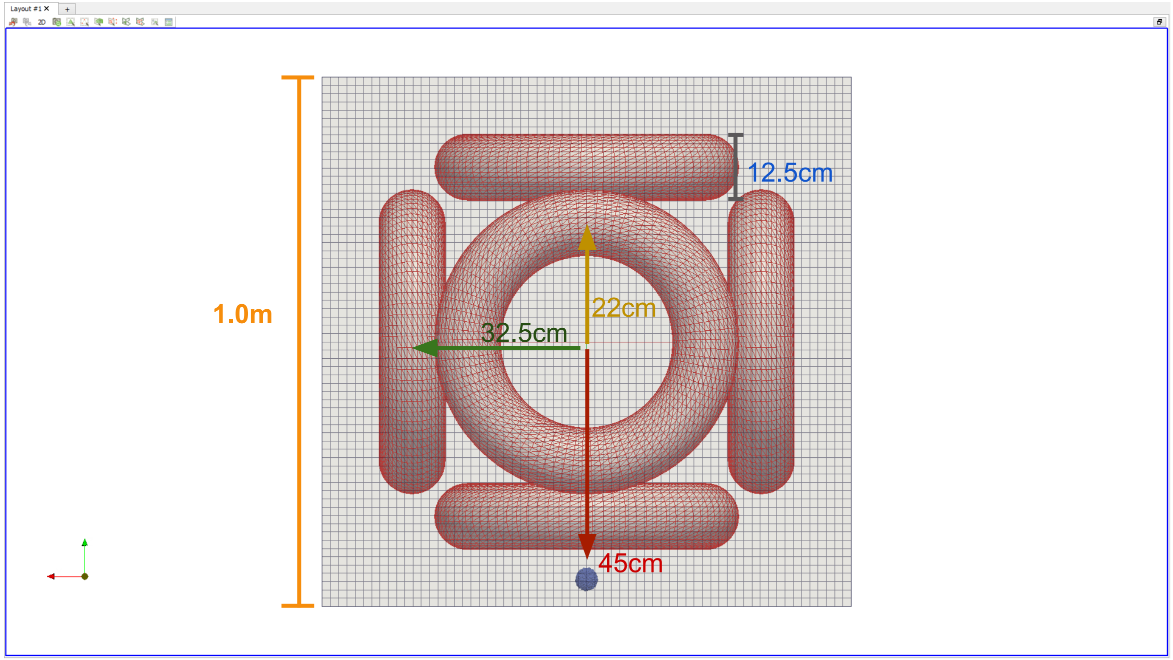
Cloud Computing: "PlasmaCloud" System



Current Gen:
Amazon Linux AMI
8-core 2.6GHz Intel Sandy Bridge CPU
NVIDIA GRID K520 Kepler GPU
15GB GDDR3 RAM
60GB SSD

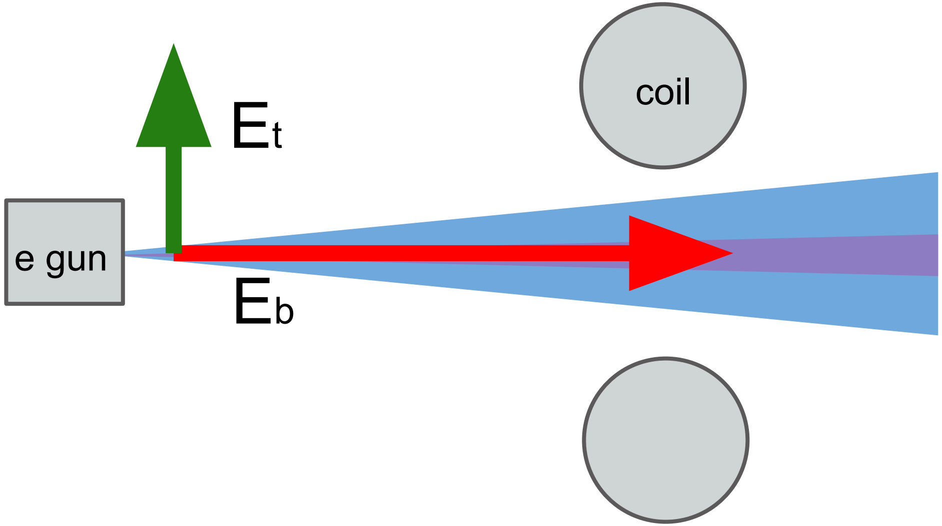
Total Cost: <$2 USD/Hour

Polywell Electron Injection Study



Device radius: 32.5cm
Coil radius: 22cm
Coil minor radius: 6.25cm
Emitter standoff: 45cm
Tank: 100cm cube
Emitter spot size: 0.5cm
Coil current: 200,000 amp-turns
Max on-axis B field: 0.57 T
Injection potential: 25kV
Injection Current: 0.8A

Polywell Electron Injection Study



Et/Eb varied from 0 to 1 among 20 instances

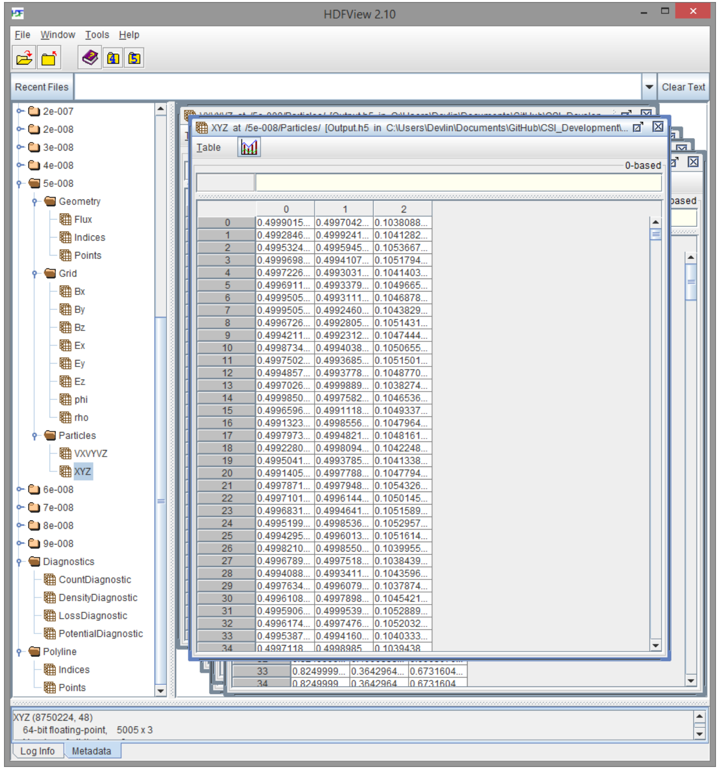
- 20 Concurrent instances
- 64x64x64 Simulation grid
- 40,800 Polygons
- Explicit integration mode (leapfrog)
- Particle sort, move, deposit and field interpolation on GPU, field solve on CPU.
- Boris mover
- Low frequency EM IBM solver (Darwin)
- Particle counts of ~1E6 to 5E7
- Mixed precision mode (float on GPU)
- Exit on equilibrium particle count (max it=8E5)

Electron Injection Study: Results

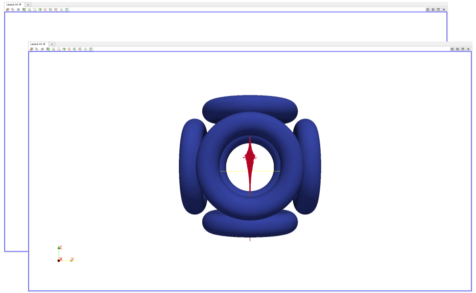
- Regional average potential
- Regional average density
- Loss rate (global)
- Geometry surface flux
- Field and mover integration error
- Fields, trace particles, density, current

17.5 GB of data.

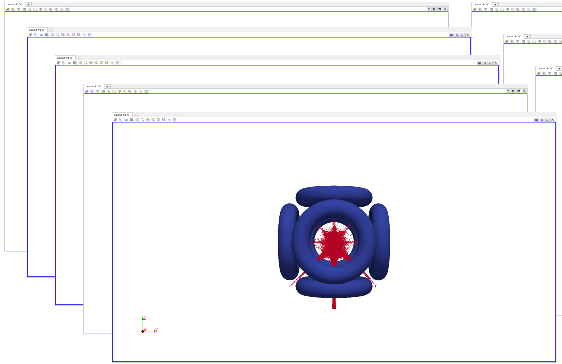
20 Instances, 8E5 time steps each, with diagnostic output every 2,000 steps



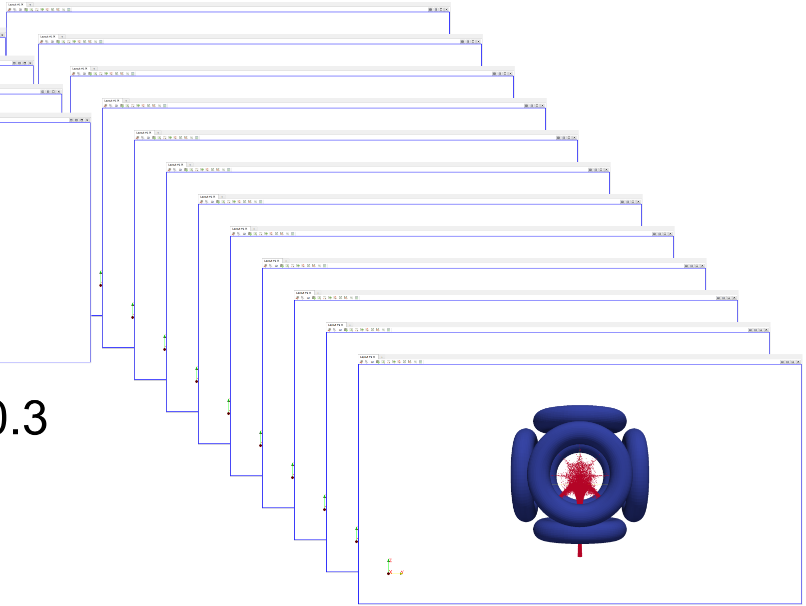
Electron Injection Study: Results



$E_t/E_b = 0.0 - 0.05$
“Pass through” Mode

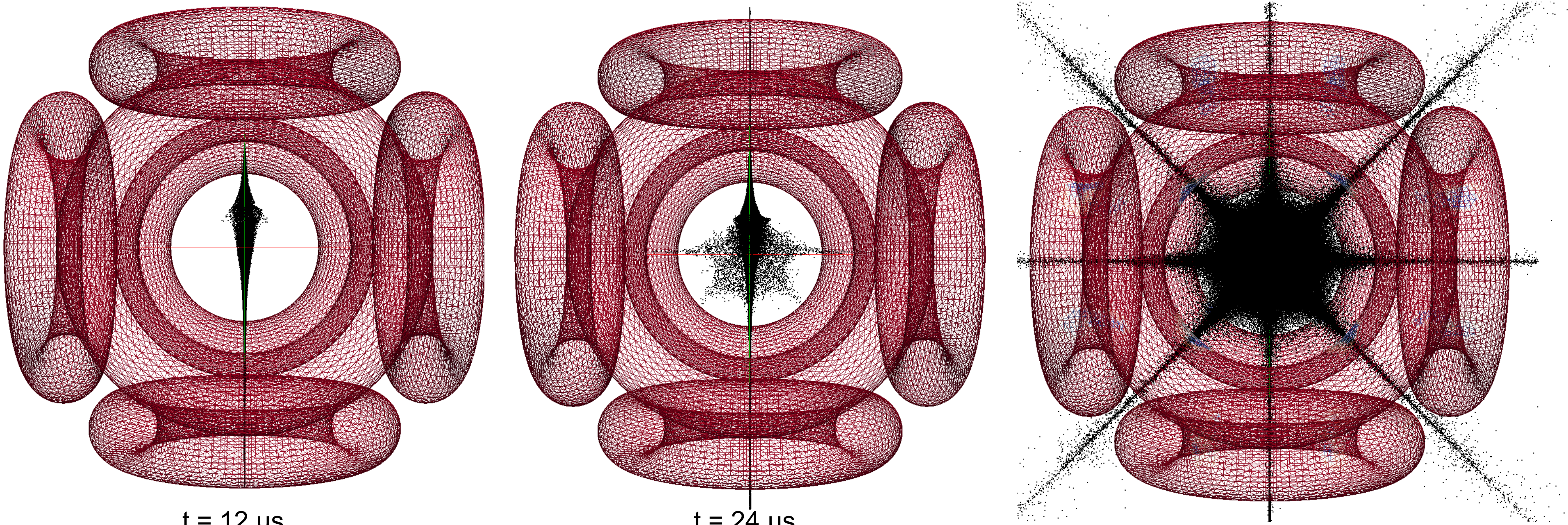


$E_t/E_b = 0.1 - 0.3$
“Fill” Mode



$E_t/E_b = 0.35 - 1.0$
“Rejection” Mode

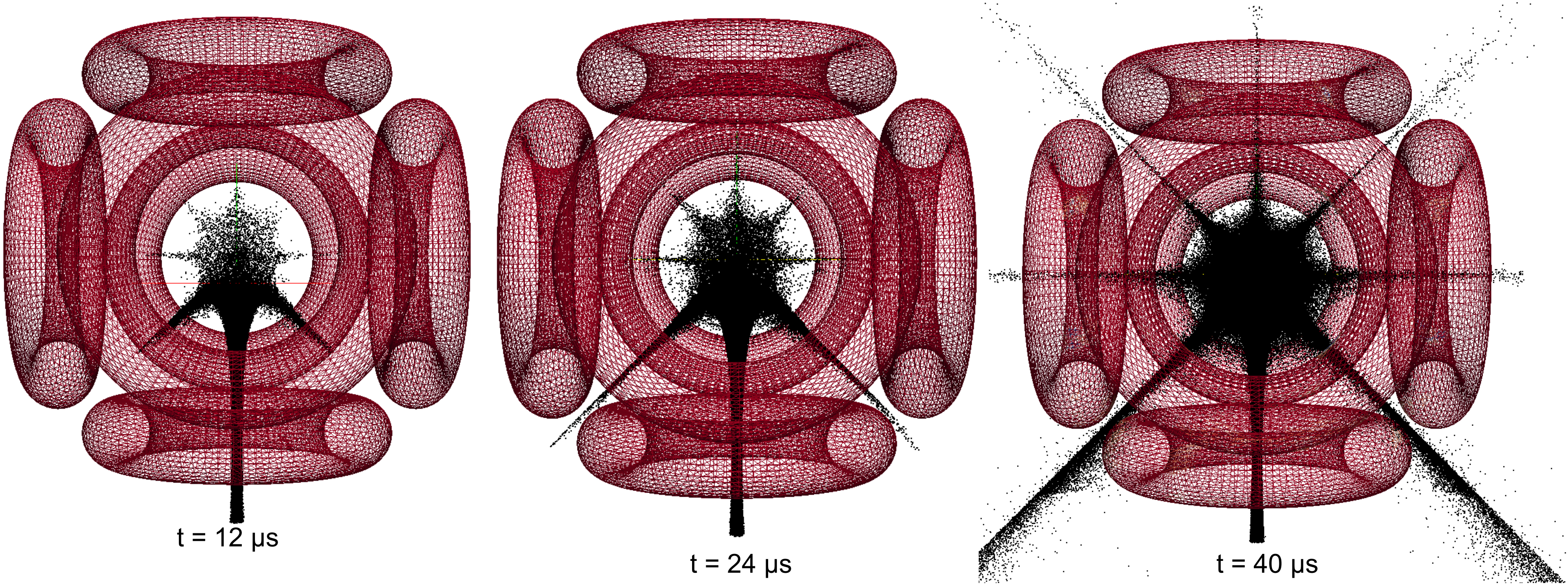
Electron Injection Study: Analysis



t = 12 μs   t = 24 μs   t = 40 μs

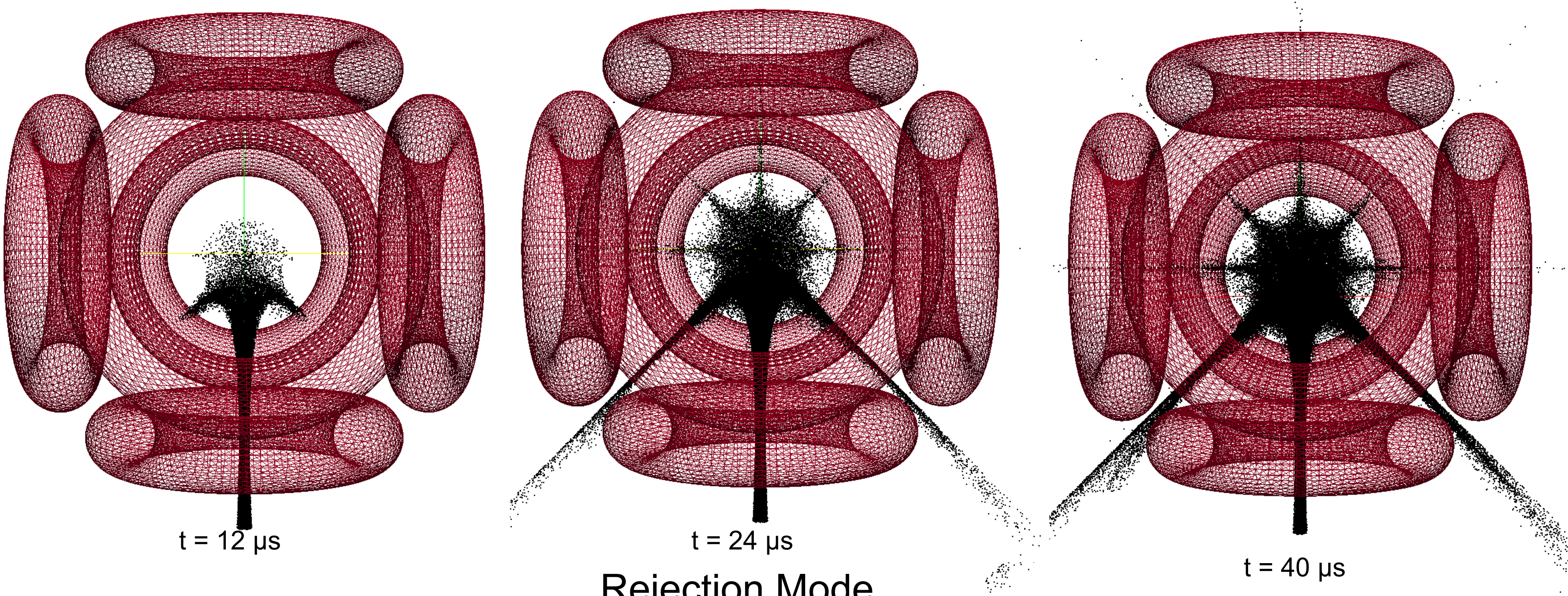
Pass-Through Mode

Electron Injection Study: Analysis



Fill Mode

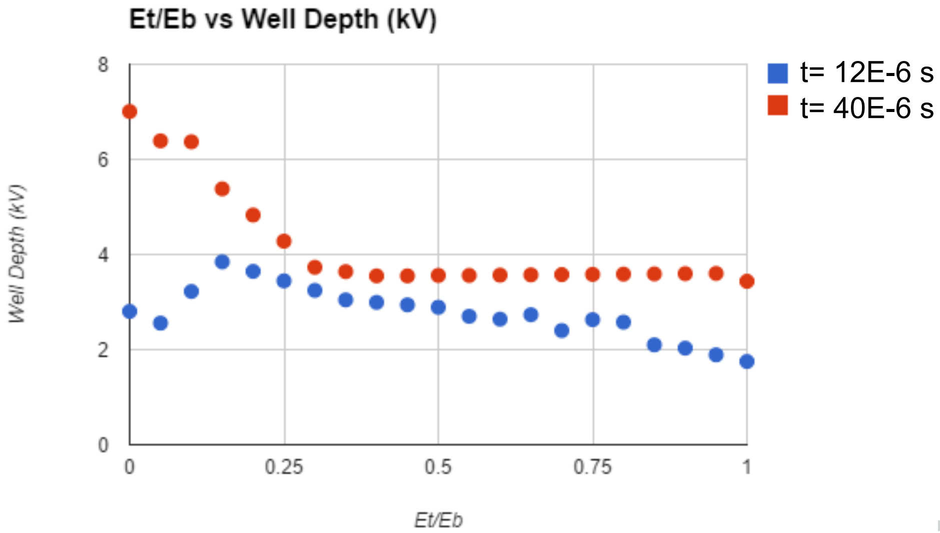
Electron Injection Study: Analysis



Rejection Mode

Electron Injection Study: Analysis

- In all cases, coil bombardment accounted for less than 1% of losses
- Pass-through mode has the best final density and well depth
- Fill mode has lowest losses initially, and achieves moderate well depth quickly
- Rejection mode is characterized by rapid losses through corner cusps of same side as injection source.

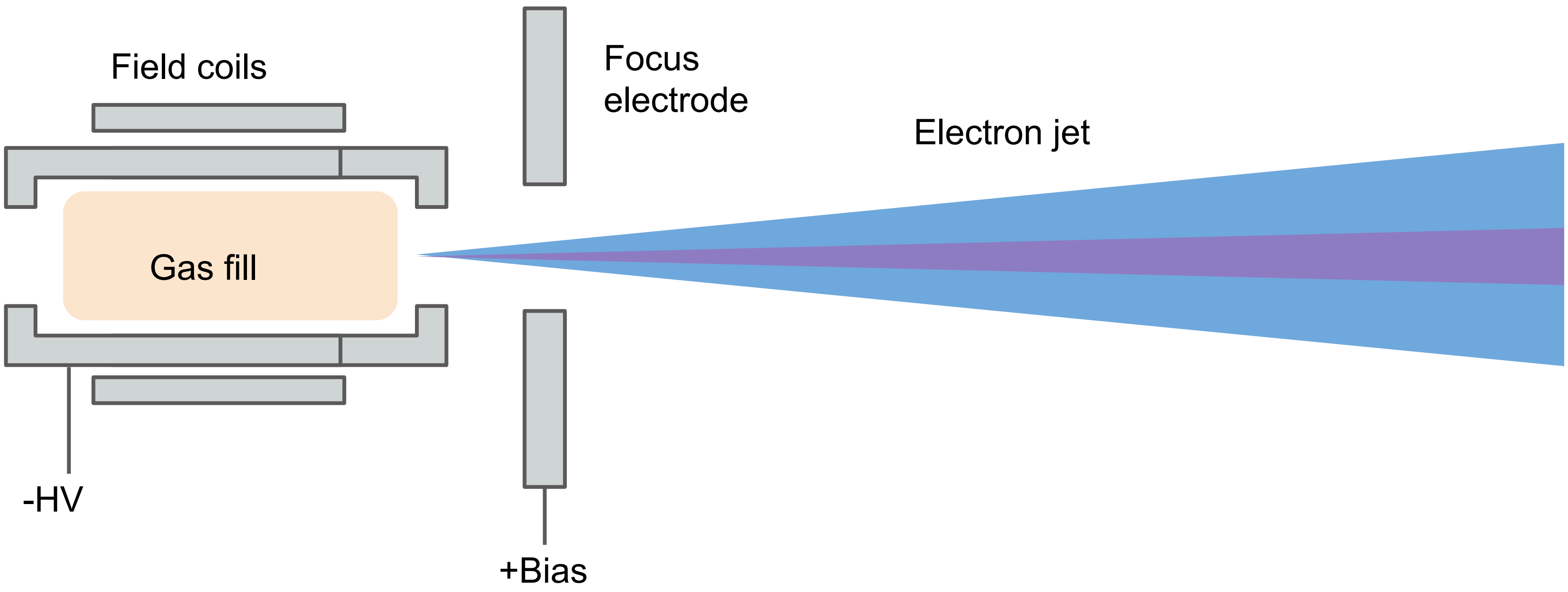


Electron Injection Study: Lessons

- Confirmation that spaced, circular (or elliptical) coil cross sections drastically reduce electron bombardment of coil containers.
- Divergent magnetic field lines and space charge will spread injected electron beams, reducing passthrough for even highly collimated beams.
- Overall well depth only a fraction of injection potential.
- Suggestion: Efficient startup sequence begins with $E_t/E_b \sim 0.15$, transitioning to $E_t/E_b < 0.05$ within 20μs.

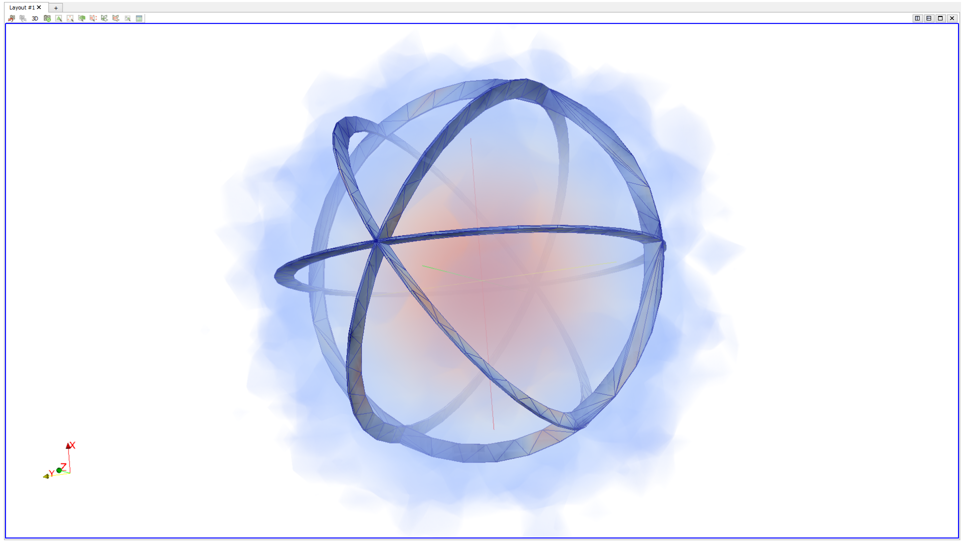
Variable Spread Electron Injector

Magnetically shielded, differentially pumped hollow cathode source with variable focus electrode



Future Work

- MCC and ionization
- Surface emission/reactions
- External circuit modeling
- Genetic algorithm?



Summary

- GP-GPU “SimCode” software has been developed to simulate plasmas in complicated geometry.
- Cloud computing “PlasmaCloud” system allows concurrent deployment of arbitrarily many SimCode instances via EC2.
- Concurrent parametric sweep of a small range of electron injection parameters demonstrated.
- Highly collimated beams not ideal for initial well formation (in absence of scattering neutrals or ions).
- Variable spread electron source proposed.

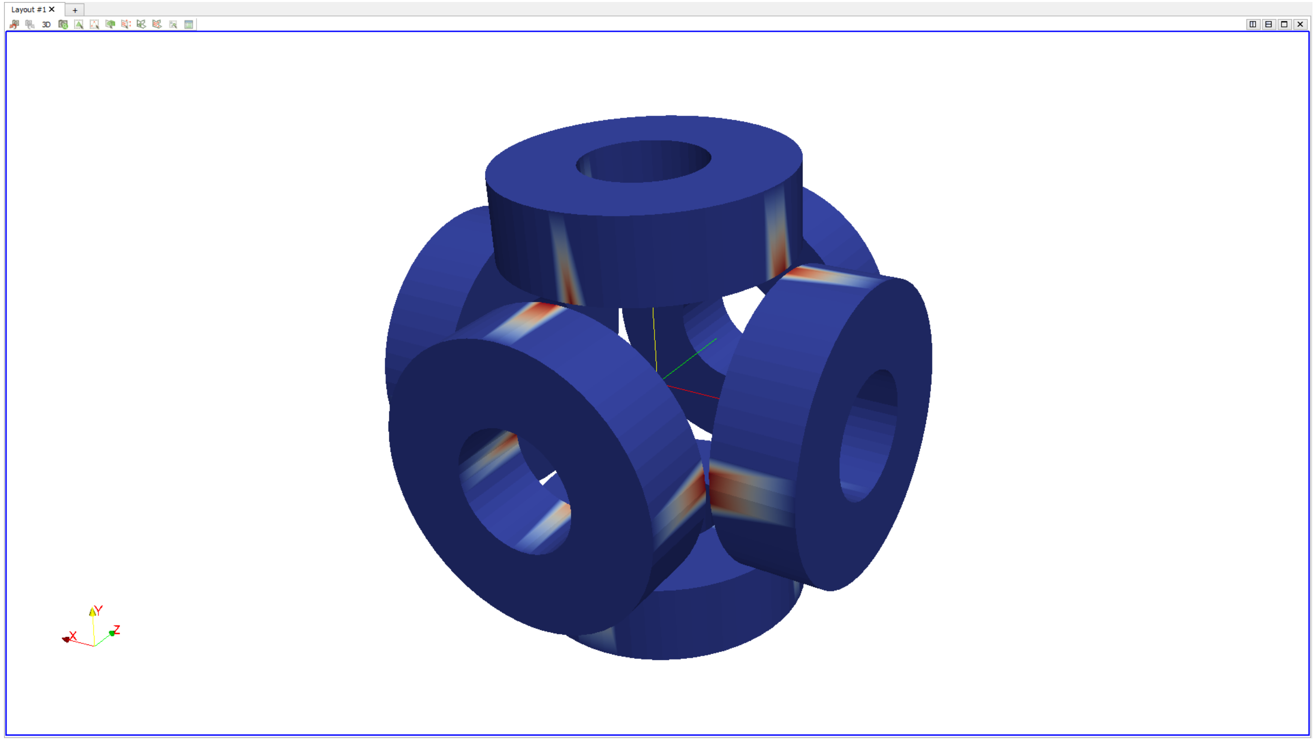
Interested in using PlasmaCloud? Let us know.

d.baker@convsci.com

info@convsci.com

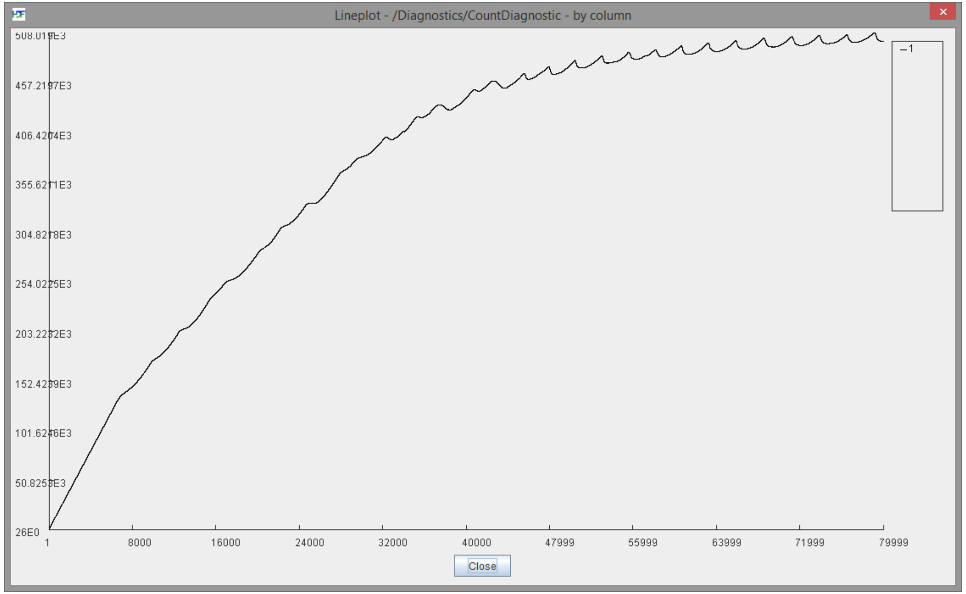
Extra Slides

Extra Slides- Non conformal coils

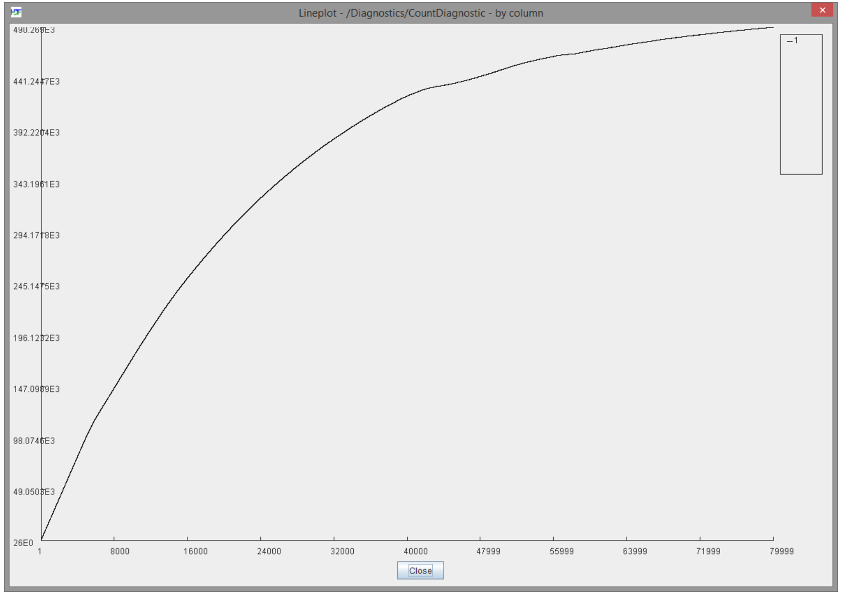


Electron Injection Study: Analysis

Oscillations in highly collimated beam



Et = 0

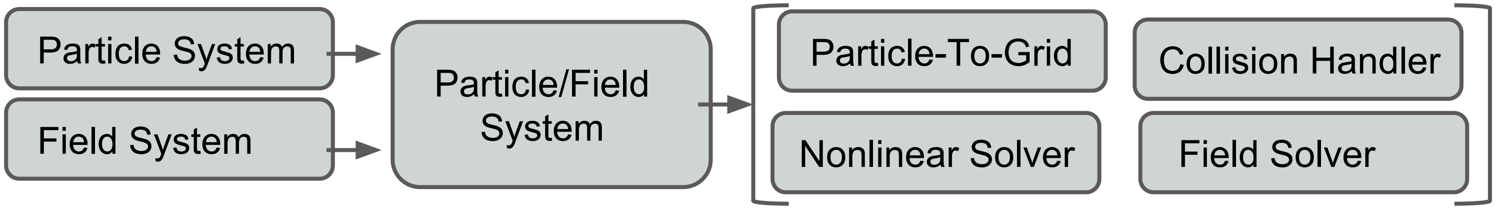


Et = 0.1

SimCode: Modular PIC System

Explicit Mode:

Implicit Mode:



GPU Data Details

Practical particle count limitation on K520 grid GPU: 5E8 Particles with 128x128x128

GPU memory overhead per particle

Particles

Total: 55 bytes

- float vx[], 4 bytes
- float vy[], 4 bytes
- float vz[], 4 bytes
- float x[], 4 bytes
- float y[], 4 bytes
- float z[], 4 bytes
- uint16 s[], 2 bytes
- int32 binId[], 4 bytes
- bool exit[], 1 byte
- float fcopy[], 4 bytes
- int32 icopy[], 4 bytes

Only when collision = 1

- float x2[], 4 bytes
- float y2[], 4 bytes
- float z2[], 4 bytes
- float dtprev[], 4 bytes

GPU memory overhead per node

Grid **Total: 40 bytes**

- float Ex[], 4 bytes
- float Ey[], 4 bytes
- float Ez[], 4 bytes
- float Bx[], 4 bytes
- float By[], 4 bytes
- float Bz[], 4 bytes
- float Jx[], 4 bytes
- float Jy[], 4 bytes
- float Jz[], 4 bytes
- float rho[], 4 bytes

GPU memory overhead per bin

- int32 spatial[], 4 bytes
- int32 start[], 4 bytes
- int32 end[], 4 bytes
- int32 gstart[], 4 bytes
- int32 gend[], 4 bytes

- A few dozen bytes for species info, grid parameters etc.
- Geometry!

References

Plasma Physics via Computer Simulation (Series in Plasma Physics) (01 October 2004) by C. K. Birdsall, A. B. Langdon

Kafafy, R.; Wang, J., "A Hybrid Grid Immersed Finite Element Particle-in-Cell Algorithm for Modeling Spacecraft-Plasma Interactions," Plasma Science, IEEE Transactions on , vol.34, no.5, pp.2114,2124, Oct. 2006

Payne, J.E., 2012. Implementation and performance evaluation of a GPU particle-in-cell code. Cambridge, MA: Massachusetts Institute of Technology

G. Chen, L. Chacón, and D. C. Barnes. 2012. An efficient mixed-precision, hybrid CPU-GPU implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm. J. Comput. Phys. 231, 16 (June 2012), 5374-5388.

S. Markidis, G. Lapenta, The energy conserving particle-in-cell method, Journal of Computational Physics 230 (18) (2011) 7037-7052.

Lapenta, Giovanni and Markidis, Stefano, Particle acceleration and energy conservation in particle in cell simulations,
Physics of Plasmas 18, 072101 (2011)