Dynogrid

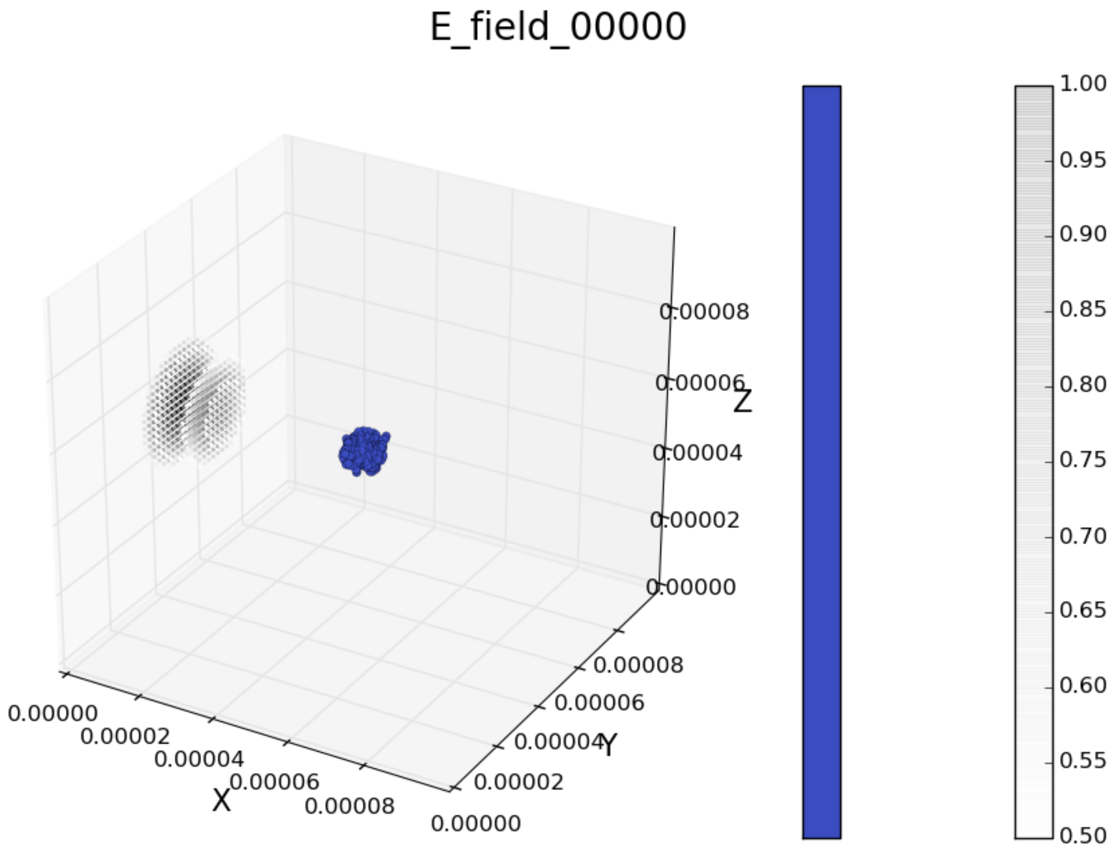
Particle in Cell Code with Adaptive 3D Grid and Dynamic Load Balancing

Tuesday, April 14th, 2015

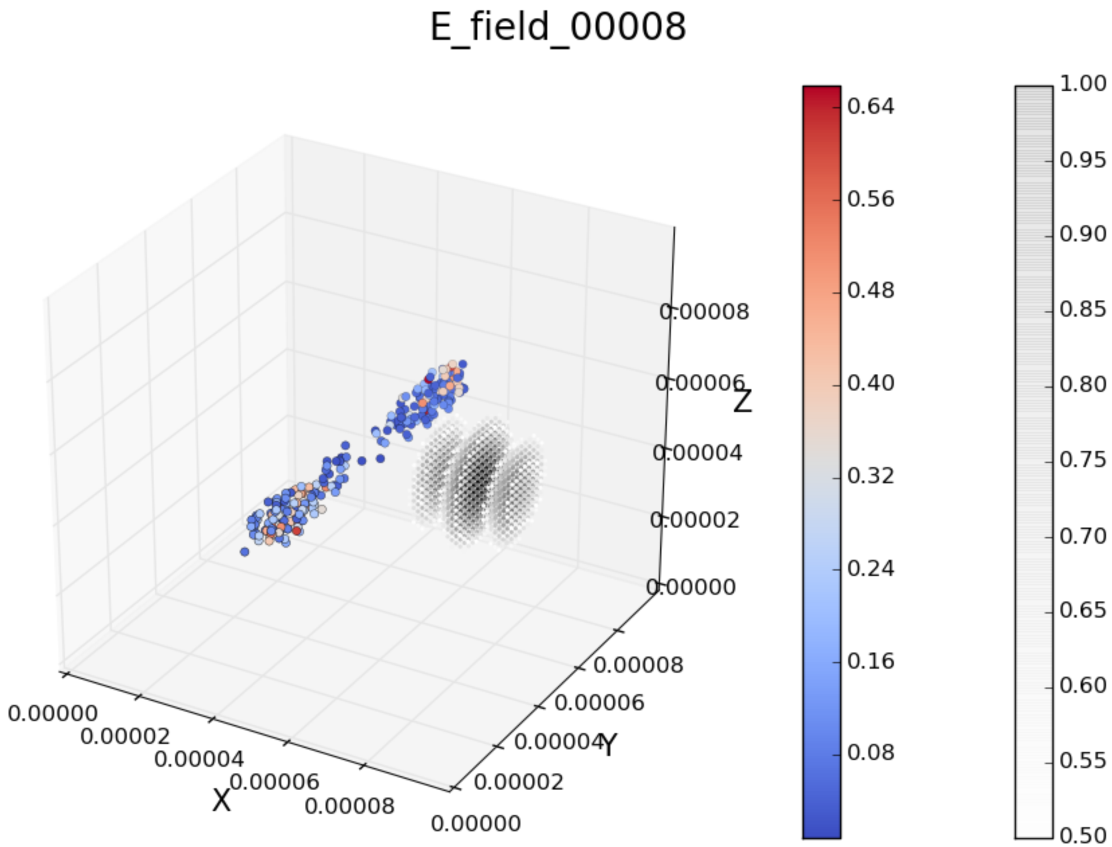
Mark Sholte
Scott Luedtke
Max Porter
Joel Iventosch

Preliminary Simulation

Initial


E_field_00000
X
0.00000
0.00002
0.00004
0.00006
0.00008
Y
0.00000
0.00002
0.00004
0.00006
0.00008
Z
0.00000
0.00002
0.00004
0.00006
0.00008
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50

Final


E_field_00008
0.00008
0.00006
Z
0.00004
0.00002
0.00000
0.00008
0.00006
0.00004
Y
0.00002
0.00000
0.00000
0.00002
0.00004
X
0.00006
0.00008
0.64
0.56
0.48
0.40
0.32
0.24
0.16
0.08
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50

Problem Definition (Theory)

- Ultimate problem is achieving optimal load balancing
- Since distributed memory (due to problem size), only communicating with nearest neighbors is reasonable
- Takes $p^{1/3}$ iterations for information to disseminate ($p^{1/3}$ = # procs in one dimension in 3D)

Problem Definition (Specifics)

- Start with a 3-dimensional grid
  - If D=Finest grid dimension needed…
    - Static grid: D^3 grid points
    - Dynamic: D^3 x (small factor) grid points
  - However: dynamic grid makes load balancing very tricky
- Problem: Load balancing
  - Distributed memory
  - Balancing both grid points (dynamic) and particles (moving)
  - Need small All2All comm, lots of (intelligent) neighbor comm

Complexity Estimates

- N = number of grid points
- M = number of particles
- Sequential case:
  - Complexity = O(N+M)
- Parallel case:
  - Complexity = $O((N+M)/p + (L+M/p^{2/3}b) + (L \log p + (n L + p^{2/3}/b) p^{1/3}))$
    - Second group of terms = swapping particles between processors as the particles move
    - Third group of terms = Load balancing, i.e. intelligently Give or Take grid points

High-level Pseudocode

Load Balancer Pseudocode

```
void Balance(){
Reduce(work);
Broadcast(total_work);
target_work = total_work / p;
while(any node not close to average)
  propensity = work - target_work;
  if (propensity < 0)
    Tell_neighbors(take, propensity);
  else
    recieve_from_right(take/give, propensity);
    recieve_from_left(take/give, propensity);
    analyize_give_left_or_right();
    give(propensity);
}
```

Goals

- Achieve optimum load balancing
- Optimize load balancer
- Build MPI communication routines
- Implement adaptive grid
- Possibly add more accurate physics – there is lots of flexibility here, depending on the time available