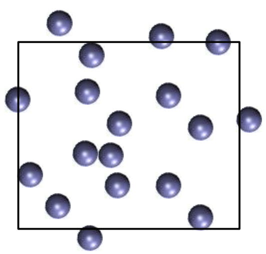
Imperial College London

Coupling The Discrete Element Method to Computational Fluid Dynamics

By
Edward Smith
and
Catherine O'Sullivan

Anderson and Jackson (1967)



Assumes

Cell >> particle

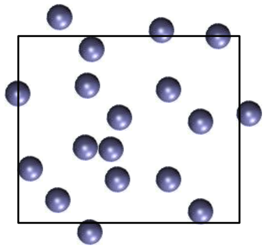
ε Porosity

$$\boldsymbol{F}^C = \sum_i \boldsymbol{f}_i^C$$ Force

Anderson and Jackson (1967)

- A Porous form of the Navier-Stokes Equations

$$\frac{\partial \rho \varepsilon \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \varepsilon \boldsymbol{u}) = -\varepsilon \boldsymbol{\nabla} P + \boldsymbol{\nabla} \cdot (\varepsilon \boldsymbol{\tau}) + \varepsilon \rho g - \boldsymbol{F}^C$$



Assumes

Cell >> particle

ε Porosity

$\boldsymbol{F}^C = \sum_i \boldsymbol{f}_i^C$ Force

- Force on particles: Added mass, Lubrication, Buoyancy, Drag forces (with empirical correlations), etc

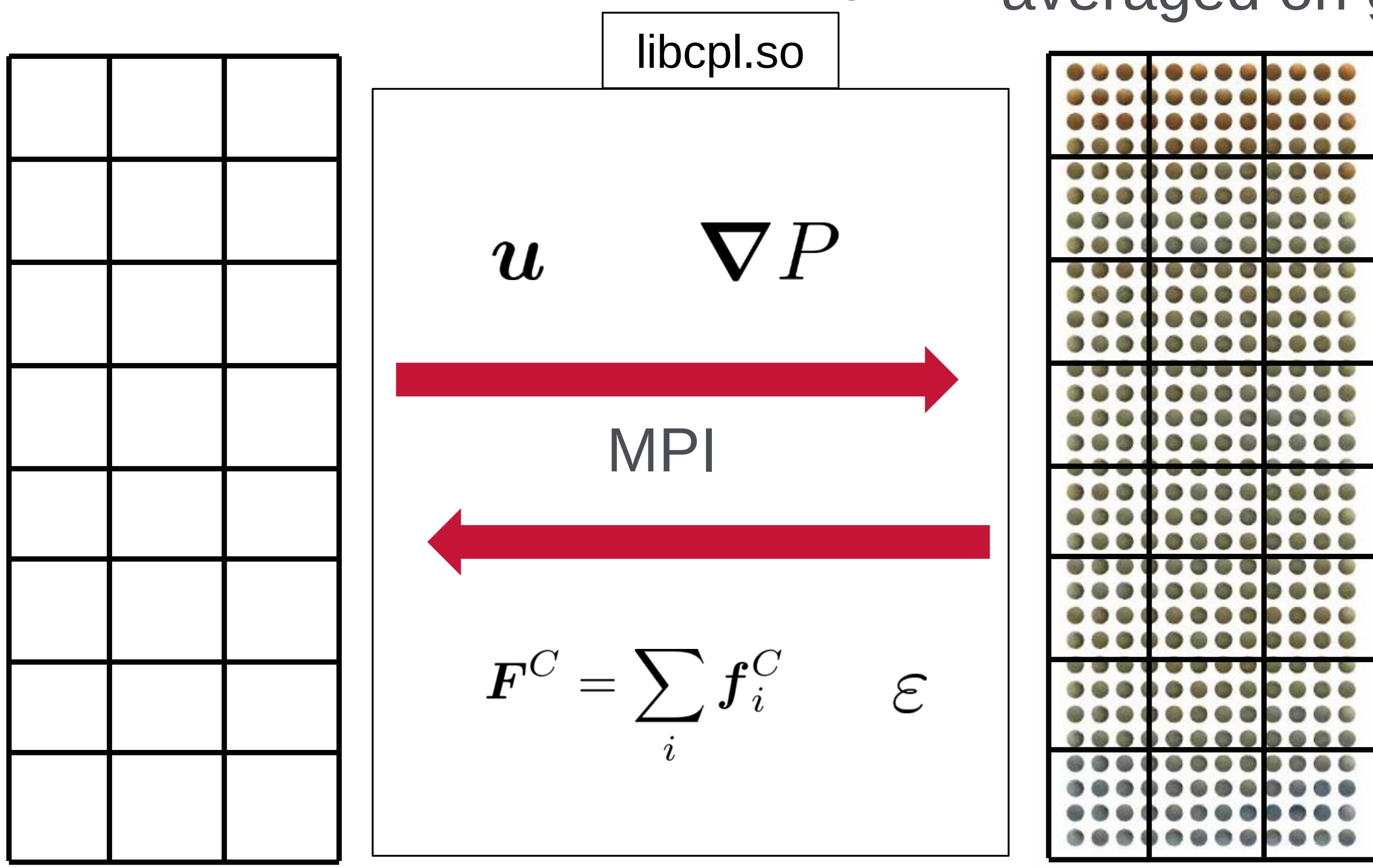
$$m_i \ddot{\boldsymbol{r}}_i = \sum_{i,j} \boldsymbol{f}_{ij} + \boldsymbol{f}_i^C$$

A Tale of Two Grids

CFD solved on grid

CPL library

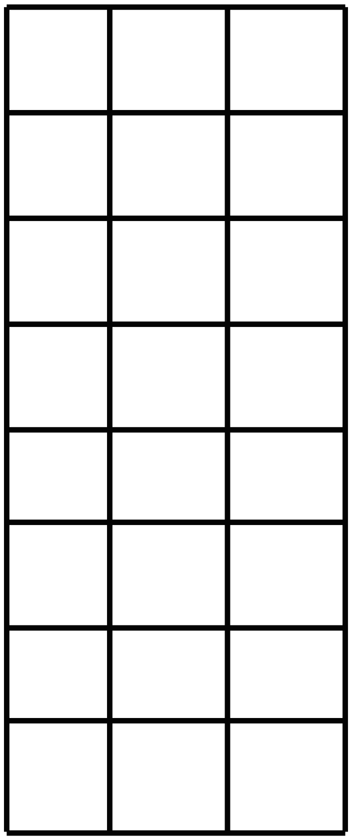
Particles
averaged on grid



Imperial College London

A Tale of Two Grids

Any CFD e.g. OpenFOAM



CPL library

libcpl.so

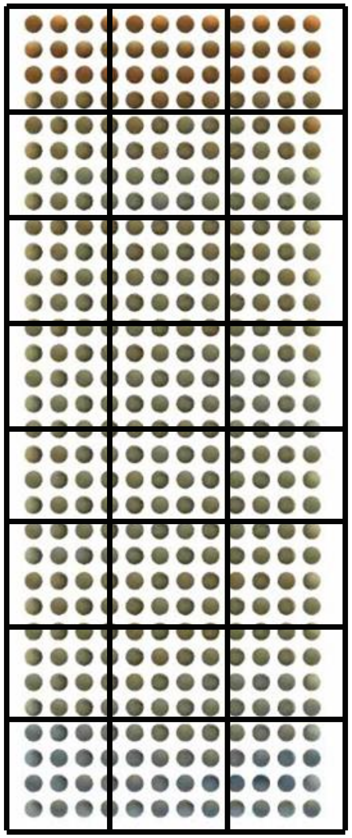
CPL_init(COMM, realm)

CPL_setup(cells, domain, proc_topology)

CPL_send(U, P) → CPL_recv(U, P)

CPL_recv(F_{sum},e) ← CPL_send(F_{sum},e)

Any DEM e.g. LAMMPS



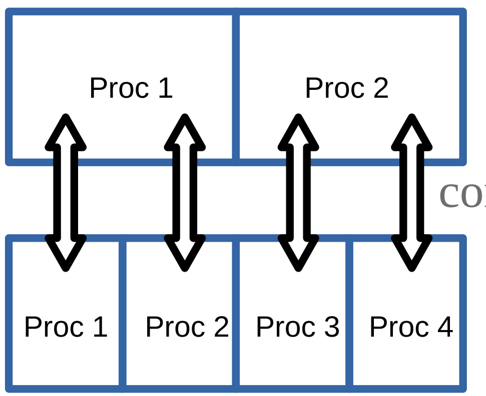
Two codes sharing a communicator

mpiexec -n 4 ./cfd.exe : –n 48 ./dem.exe

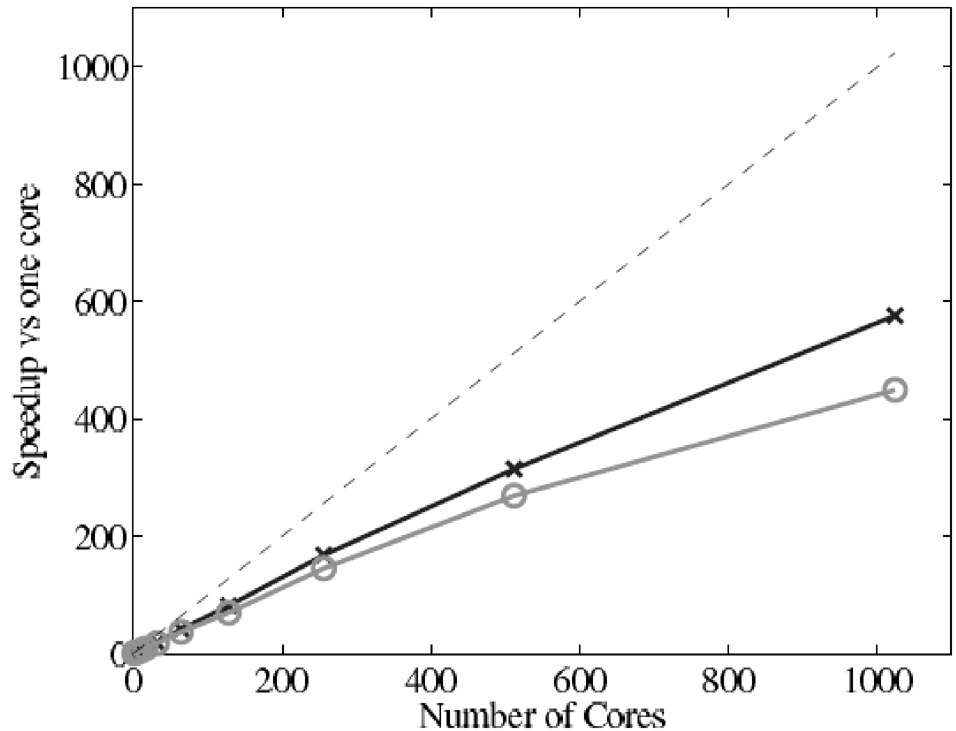
Imperial College London

CPL library

- Developed for linking of particle and continuum code
- Previous focus on scalability (for supercomputers)



All MPI communications are local



(a) Parallel speedup of the MD solver only (×), coupled code (○) against the ideal speedup (−−)

- Current focus on reliability and ease of use
- Maintains separate scope of each code by linking shared library

Weak scaling	
- Particle only	x
- Particle Coupled	o

- Testing the basic units of code

```
TEST_F(CPL_Force_Test, test_CPL_array_size) {
    int nd = 9;
    int icell = 3;
    int jcell = 3;
    int kcell = 3;
    CPL::ndArray<double> buf;
    int shape[4] = {nd, icell, jcell, kcell};
    buf.resize (4, shape);

    //Test sizes and shapes
    ASSERT_EQ(buf.size(), nd*icell*jcell*kcell);
    ASSERT_EQ(buf.shape(0), nd);
    ASSERT_EQ(buf.shape(1), icell);
    ASSERT_EQ(buf.shape(2), jcell);
    ASSERT_EQ(buf.shape(3), kcell);

};
```

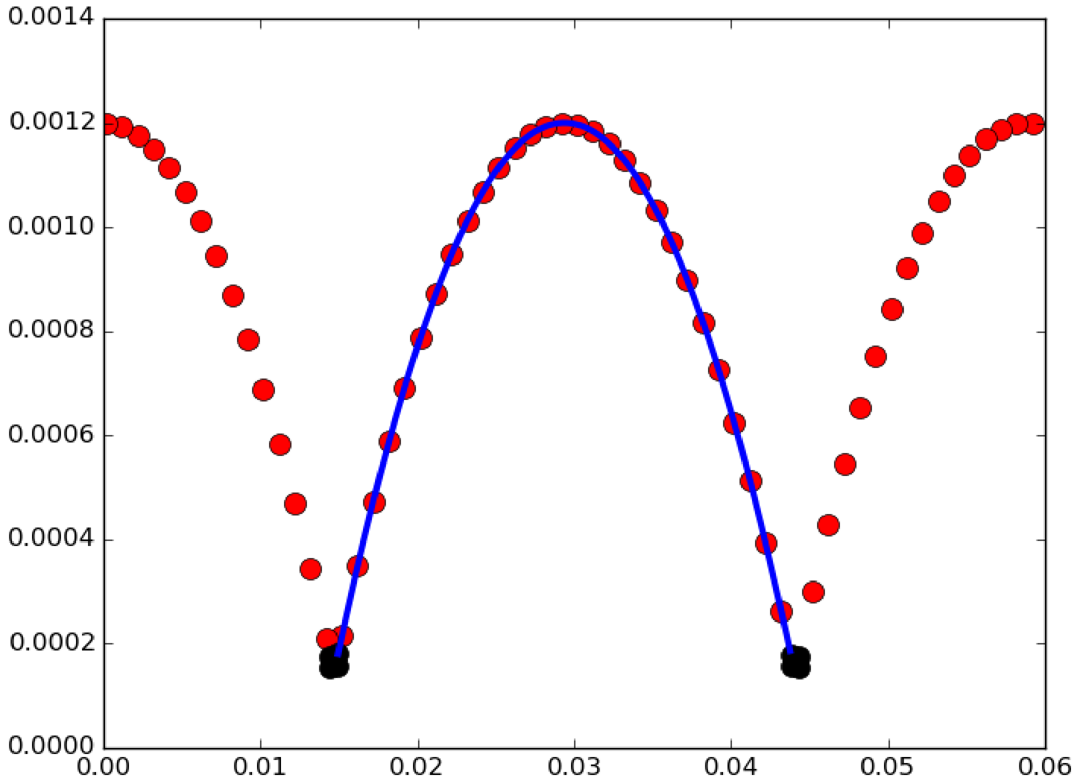
GitHub

Travis CI

Software Sustainability Institute

Validation (LAMMPS)

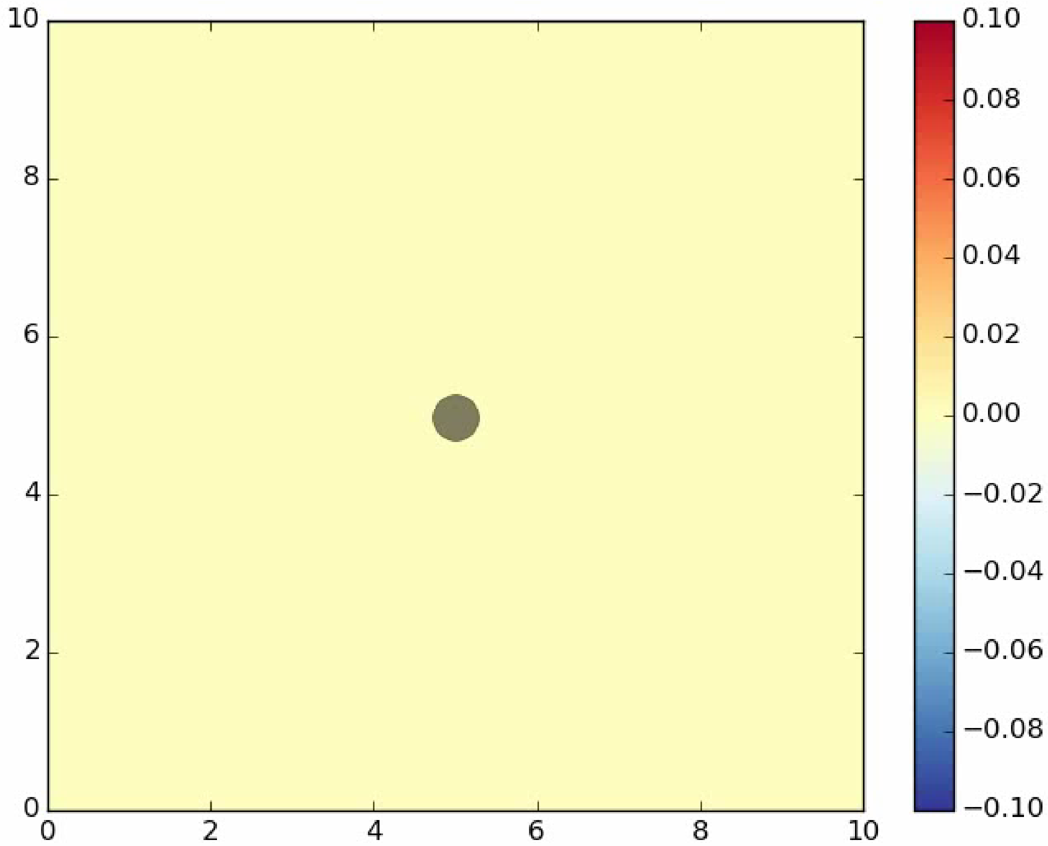
- Particle bouncing on a wall



Imperial College London

Validation (OpenFOAM)

- Particle moving through a fluid



Imperial College London

Summary

- We are coupling two separate codes to run together
  - Computational Fluid Dynamics
  - Discrete Element Method

- Build codes separately and exchange all information as average fields through shared library (CPL library)

- This is good because it:
  - Allows separate testing of both codes
  - Maintains scope of both codes
  - Promotes optimal scaling