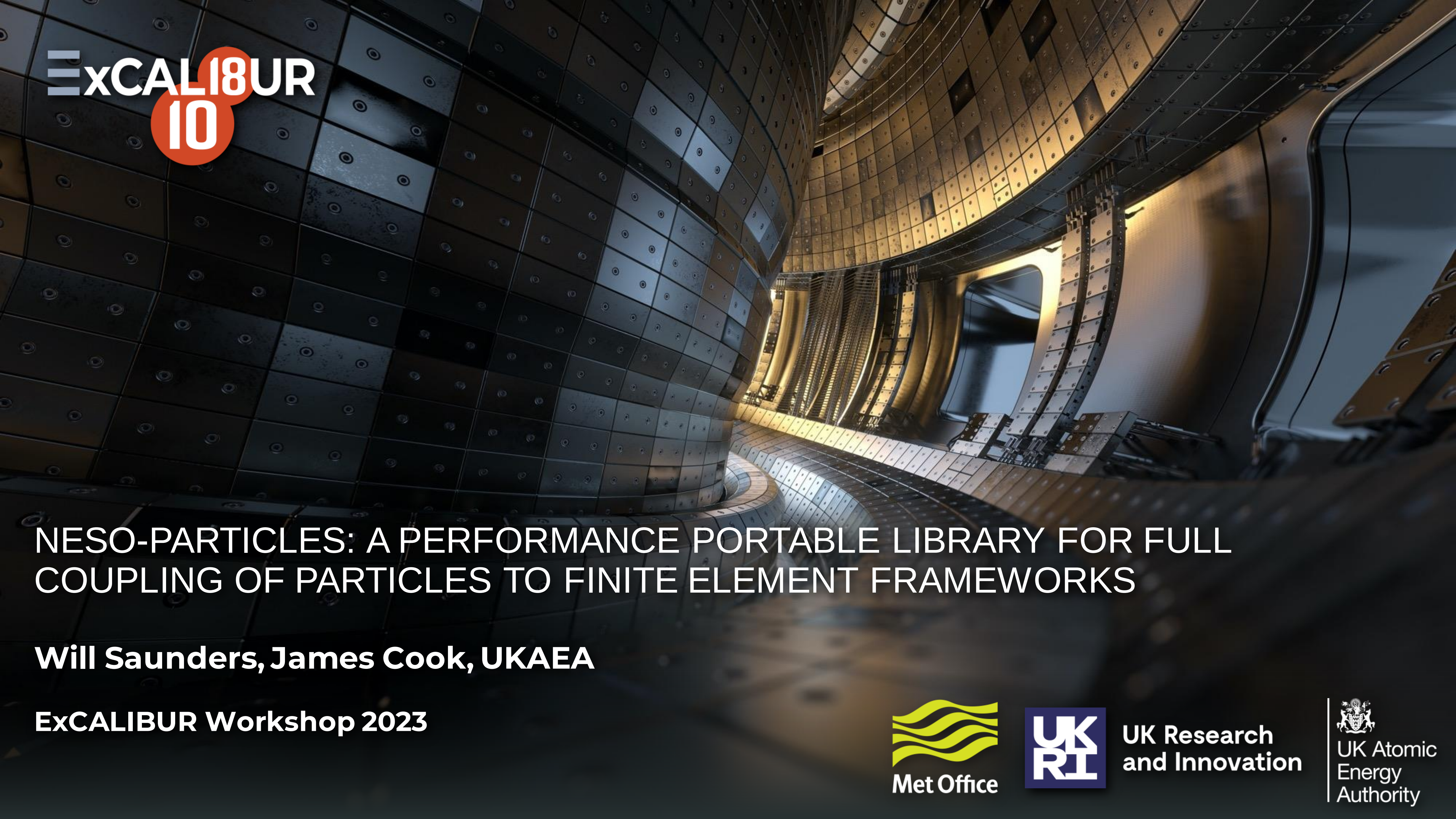


ExCALIBUR 10

NESO-PARTICLES: A PERFORMANCE PORTABLE LIBRARY FOR FULL COUPLING OF PARTICLES TO FINITE ELEMENT FRAMEWORKS

Will Saunders, James Cook, UKAEA

ExCALIBUR Workshop 2023

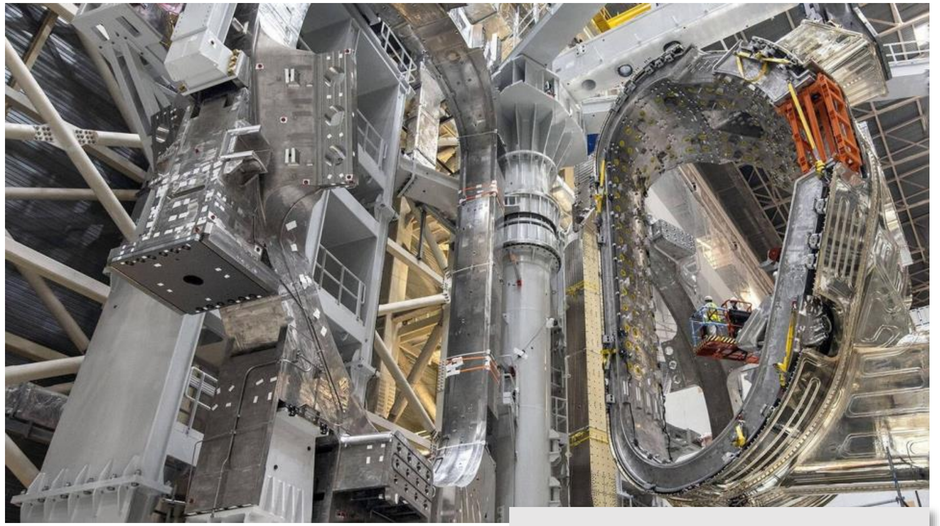
Met Office

UK Research and Innovation

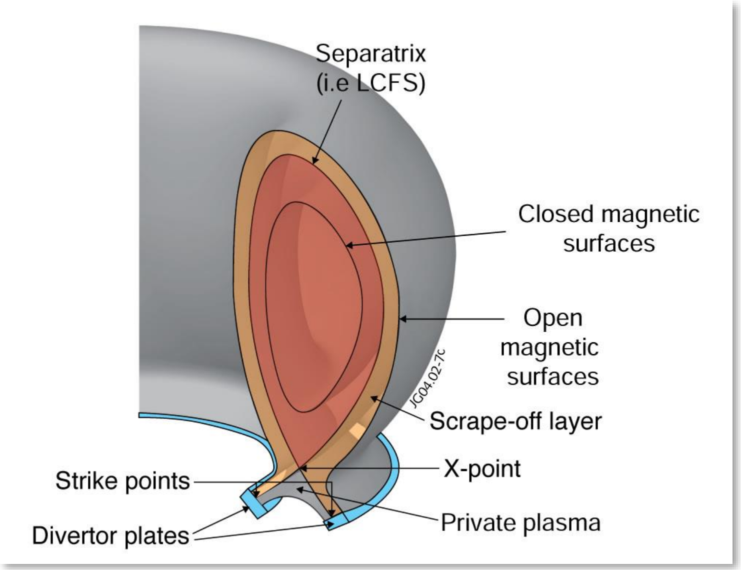
UK Atomic Energy Authority

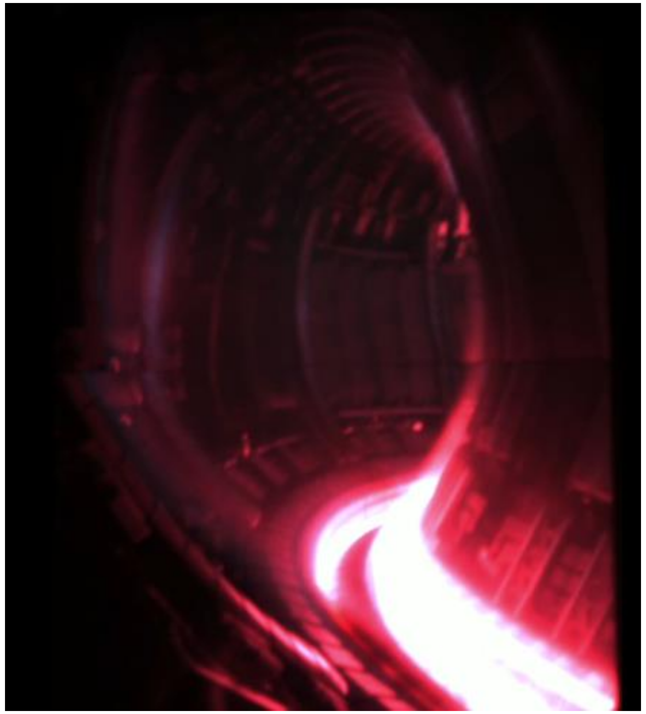
ExCALIBUR NEPTUNE

- Modelling the plasma edge/exhaust.
- A long-established exascale grand-challenge multi-physics, multi-scale problem.
- Complexity: turbulence, many species, atomic physics, etc.
- Kinetic effects: out-of-thermal equilibrium matter (few collisions), requires coupled fluid and particles.

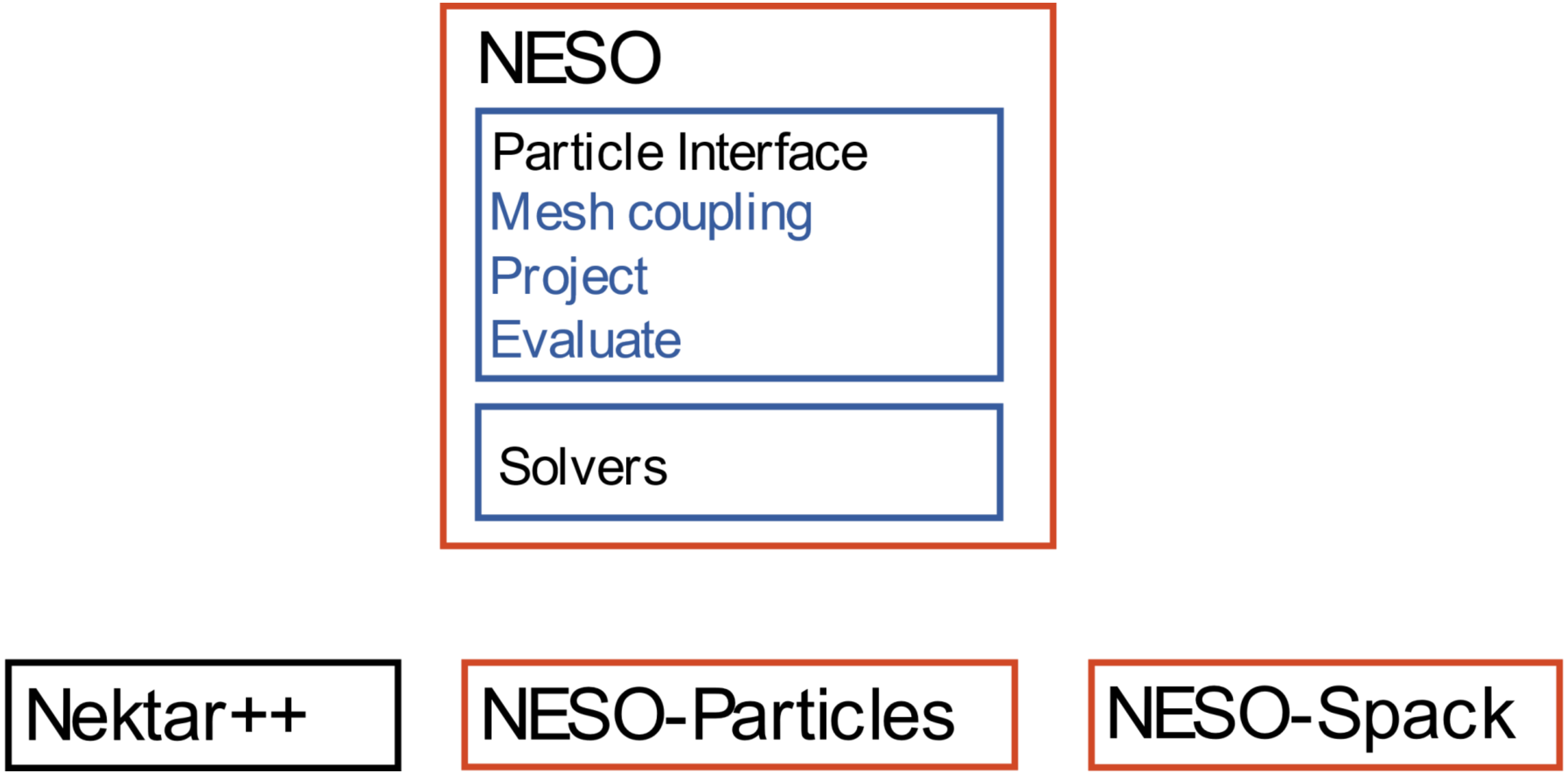


ITER Magnet Coils





Core Components

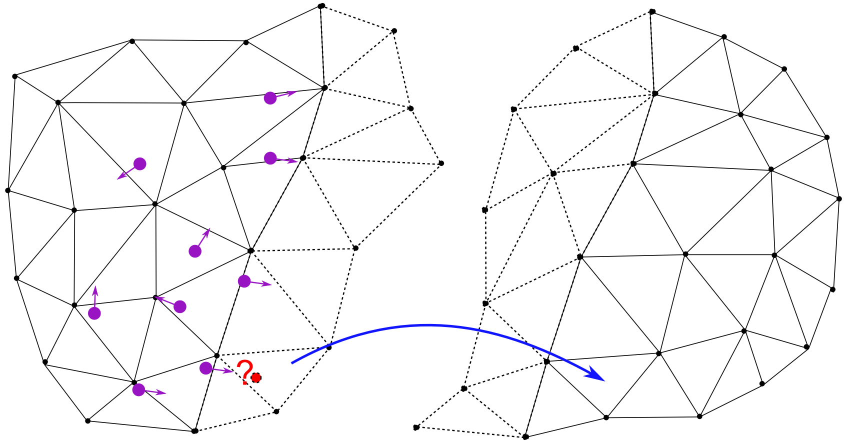


NESO-Particles: Core Components

- SYCL Particle framework
- Unstructured meshes

- Particle data communication
  1. Highly directional plasma flow (along field lines)
  2. Fast neutral flow (typically global and omnidirectional)
  3. Unstructured high-order mesh

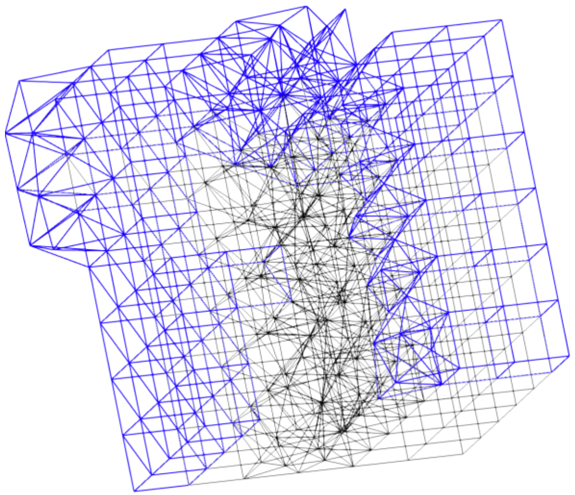
- Particle Based Operations/Data structures
  1. Particle properties – position, velocity, charge, id...
  2. Loops over particles
  3. Degrees of Freedom (DOF) – Particle Loops
  4. Particle – Particle Loops



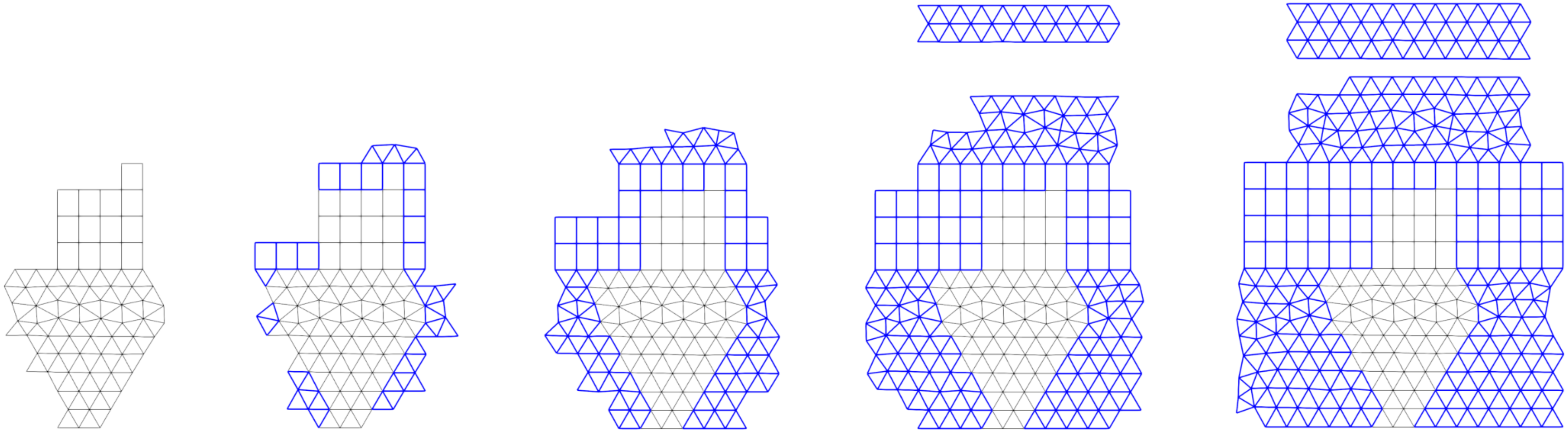
Combination of halo regions plus a global move method.

NESO-Particles: Halos

3D halo building example.



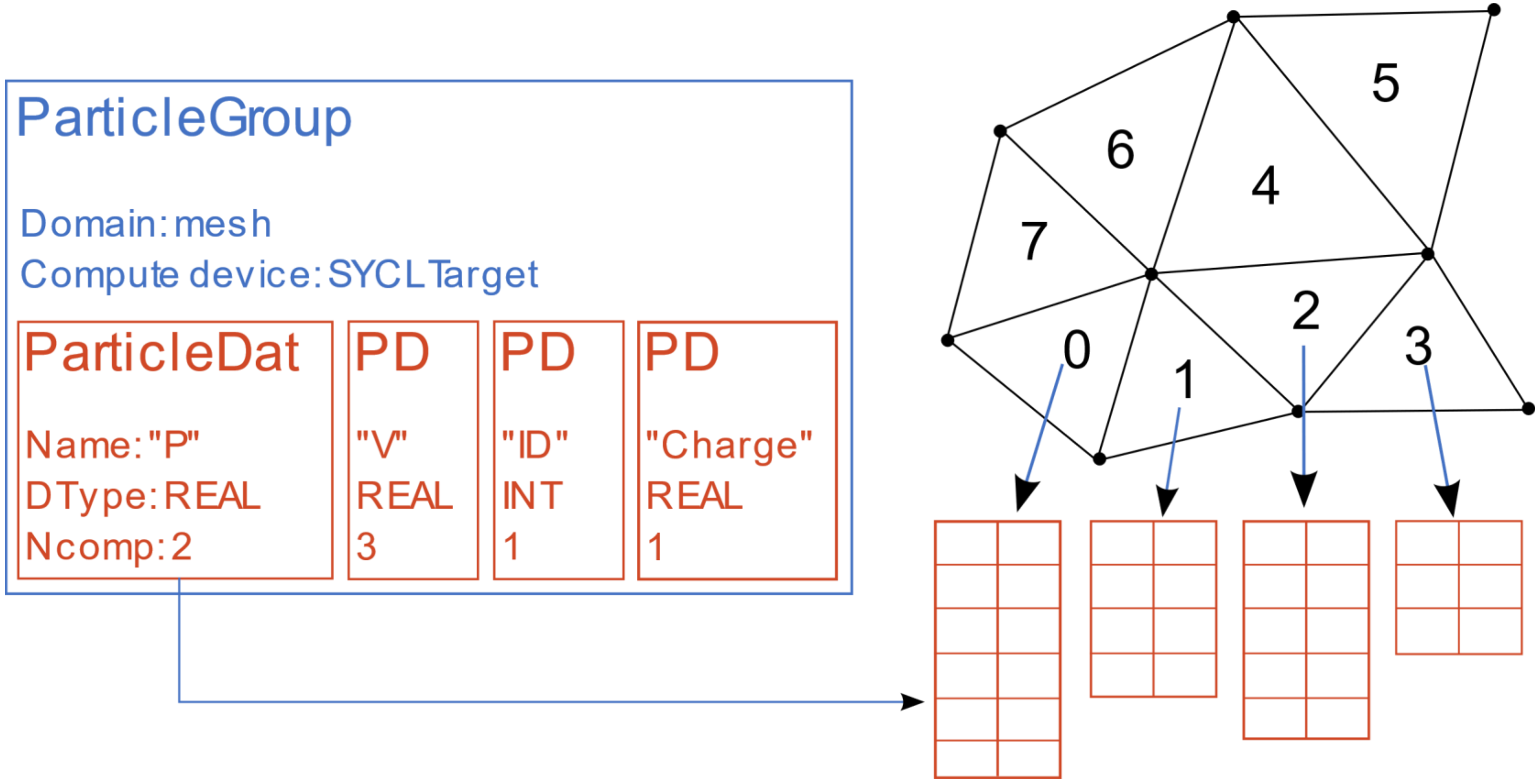
- Halos enable local communication patterns.
- Larger halo – more likely particle is communicated via local method.
- Can choose wider halo widths with faster particle movement.



Particle Data

ParticleGroup, ParticleDat

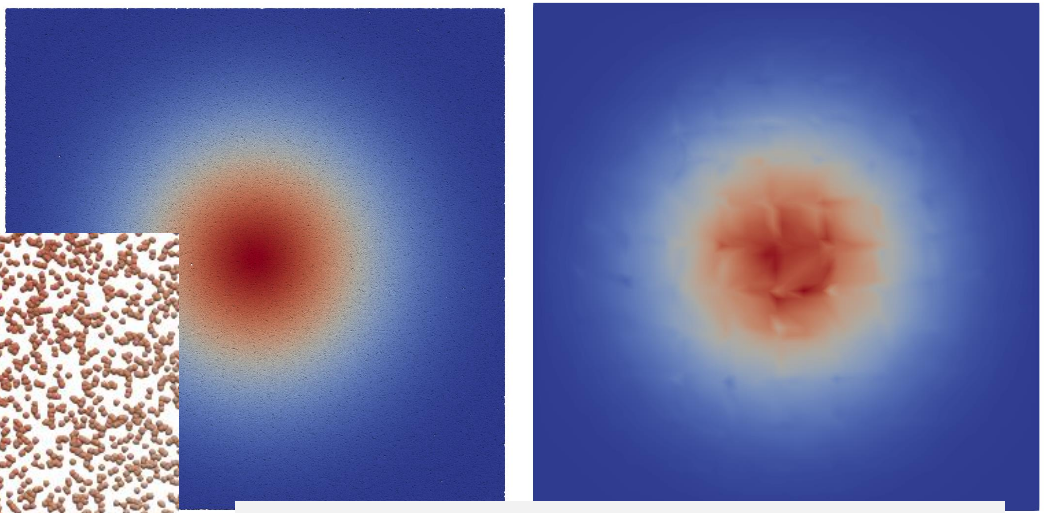
- Combines the: mesh, compute device and particle data.
- Implements particle bookkeeping – cells and MPI ranks.
- General particle properties, e.g. charge, mass, weight, velocity.

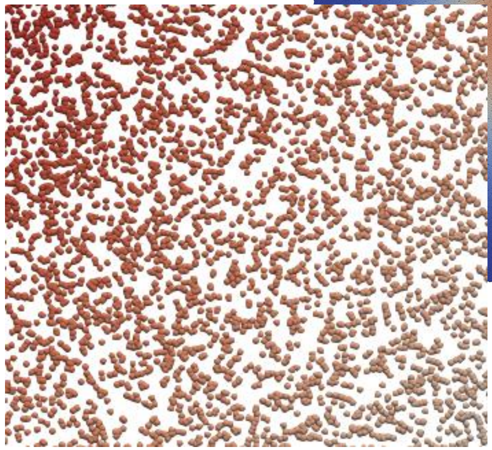


NESO

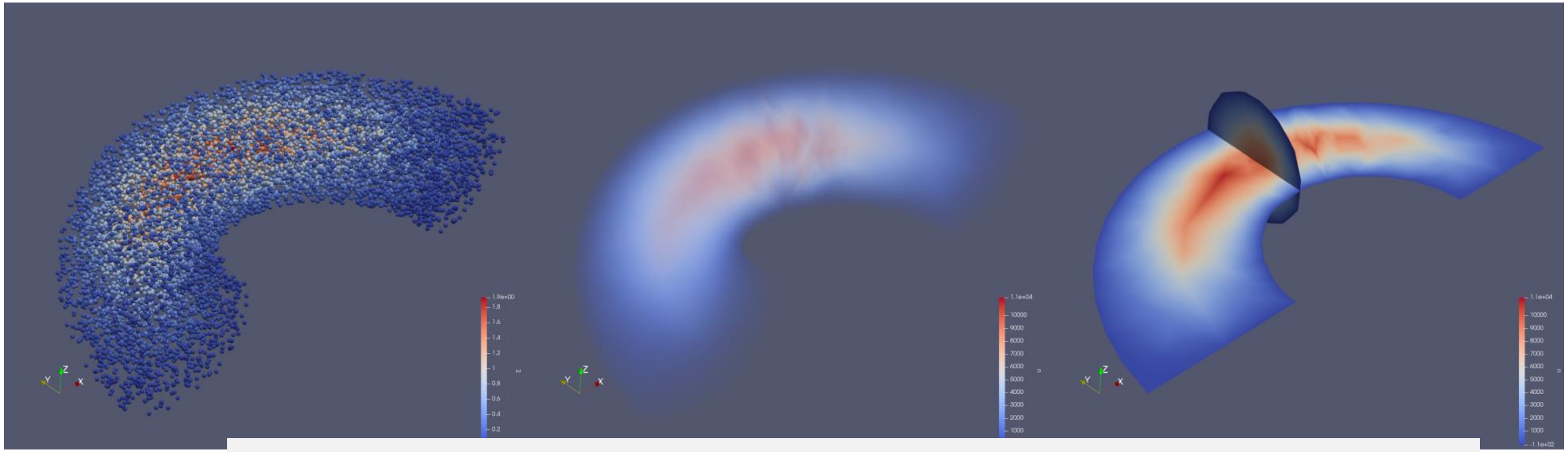
Coupling Finite Element Method and Particles

- Coupling from particles to FEM via L2 Galerkin projection.
- FEM to particles is point evaluation.
- Extends to complex geometry.





- Uniformly distributed positions
- Gaussian distributed weights

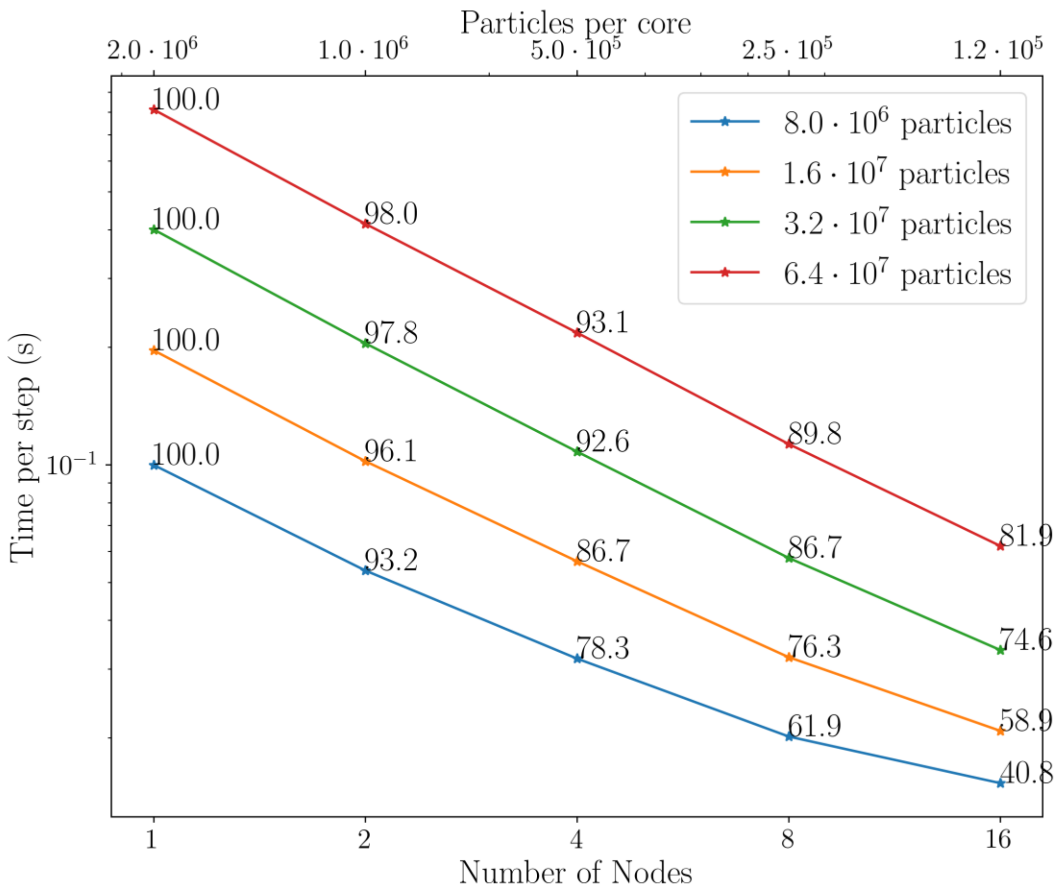


3D projection example on a half torus constructed with Hexahedrons

The End

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

Initial Profiling/Scaling



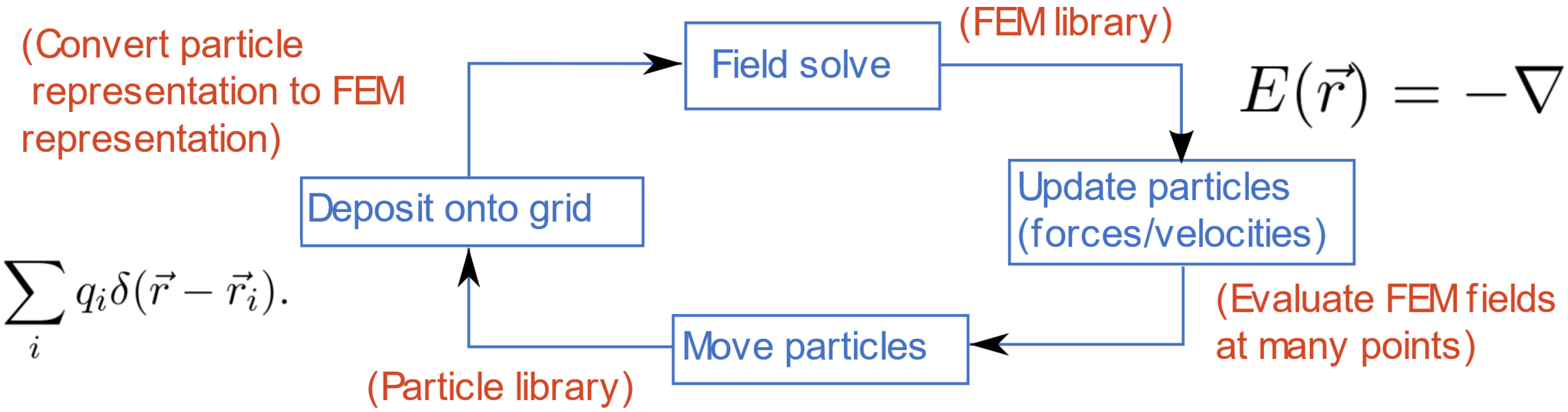
- Two-stream heavily biased towards particle work over finite element work.
- Strong scaling limit approximately 100k particles/core.

ExCALIBUR

PIC Loop

Overview

$$\Delta\phi(\vec{r}) = -\rho(\vec{r})$$



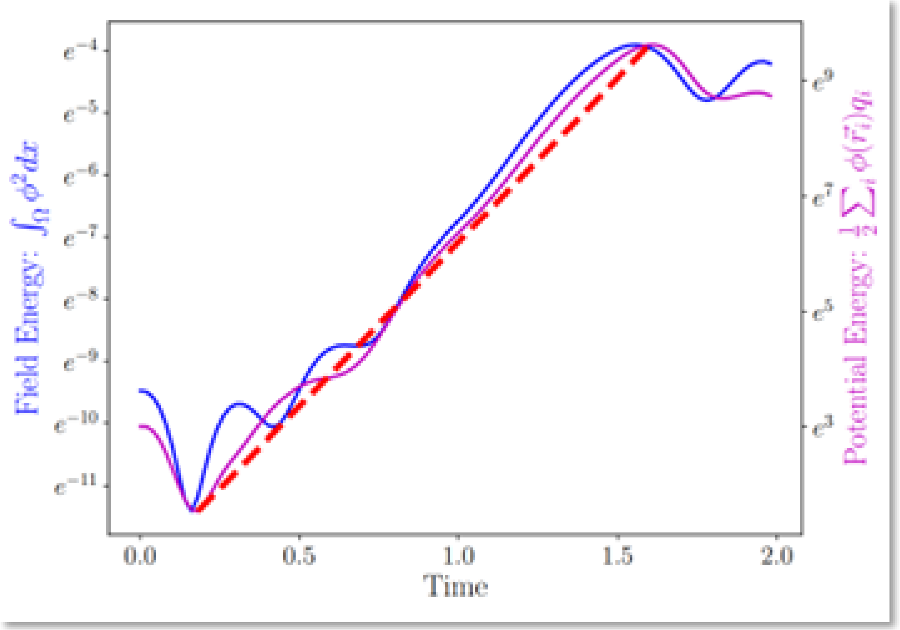
$$E(\vec{r}) = -\nabla\phi(\vec{r})$$

$$\rho(\vec{r}) = \sum_i q_i \delta(\vec{r} - \vec{r}_i).$$

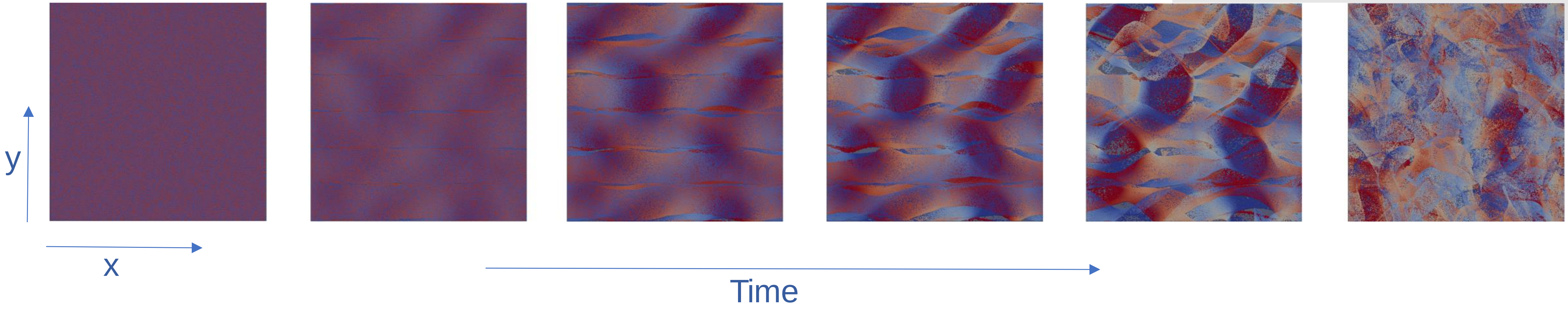
- Integration done with suitable integrator (Velocity-Verlet).
- Synopsis – More involved (and useful) schemes may combine steps.
- PIC schemes exist that conserve quantities of interest, e.g. charge(mass), energy and momentum.
- Loop till convergence/end time.

Two Stream Instability

NESO [1]	• Test implementations integrating particle capabilities and FEM. • Can be built using Spack package manager. • 2D2V electrostatic particle-in-cell solver. • Nektar++ provides Poisson solve.
Tests	• Linear growth rates of unstable modes. • Energy conservation.



Instability growth rate vs theory



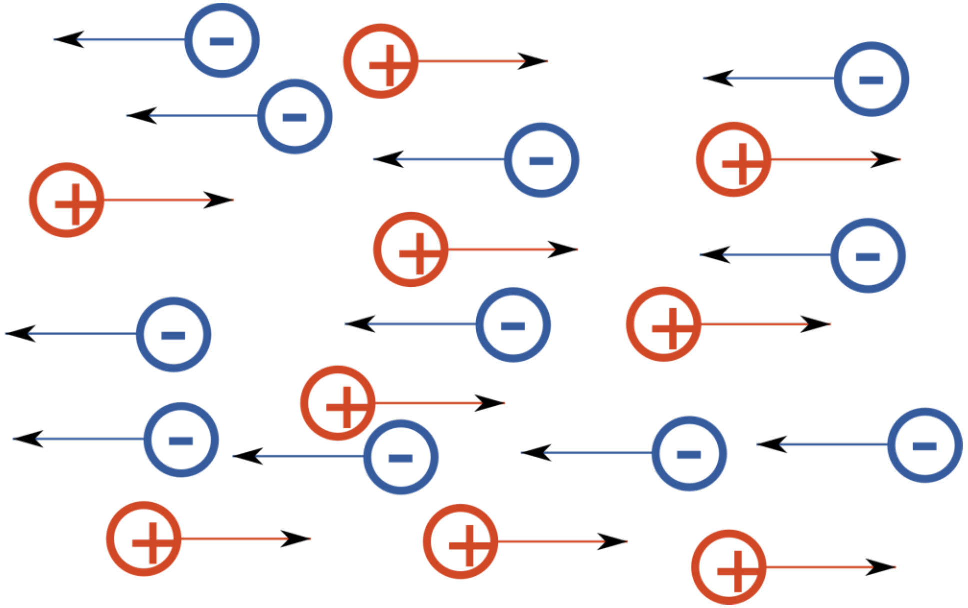
Time evolution (left to right) of 512k interacting particles. Colour is y-velocity.

1. https://github.com/ExCALIBUR-NEPTUNE/NESO

Two Stream Instability

Motivational Example

- Periodic Boundary Conditions
- Overall charge neutral system
- Electrostatic interactions through a mesh representation (not point to point Coulomb interactions)
- Initial velocities are +1/-1 in x
- Unstable initial conditions

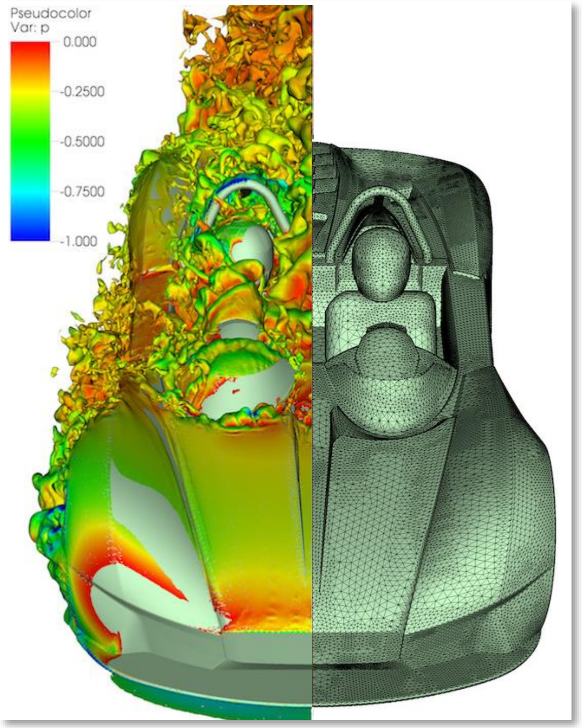


NESO Field Solve: Nektar++

NEKTAR++
SPECTRAL/HP ELEMENT FRAMEWORK

- Arbitrary convergence order *p.(*Error h^p (element size h)).
- Arithmetic intensity – increased number of operations on same data - counters HPC data movement bottleneck.
- Flow preferentially along field lines.
- Good support for complicated geometries, curved elements.

Structure	• Set of libraries. • C++ code with MPI parallelism for CPUs. • Refactoring for performance portability / GPUs / C++17.
Provenance	• Proven scaling to c.100k cores. • Well-tested code. • Established community of developers / users.
Benefit	*Good complex geometry support.*



CFD simulation of Elemental RP1 track car.

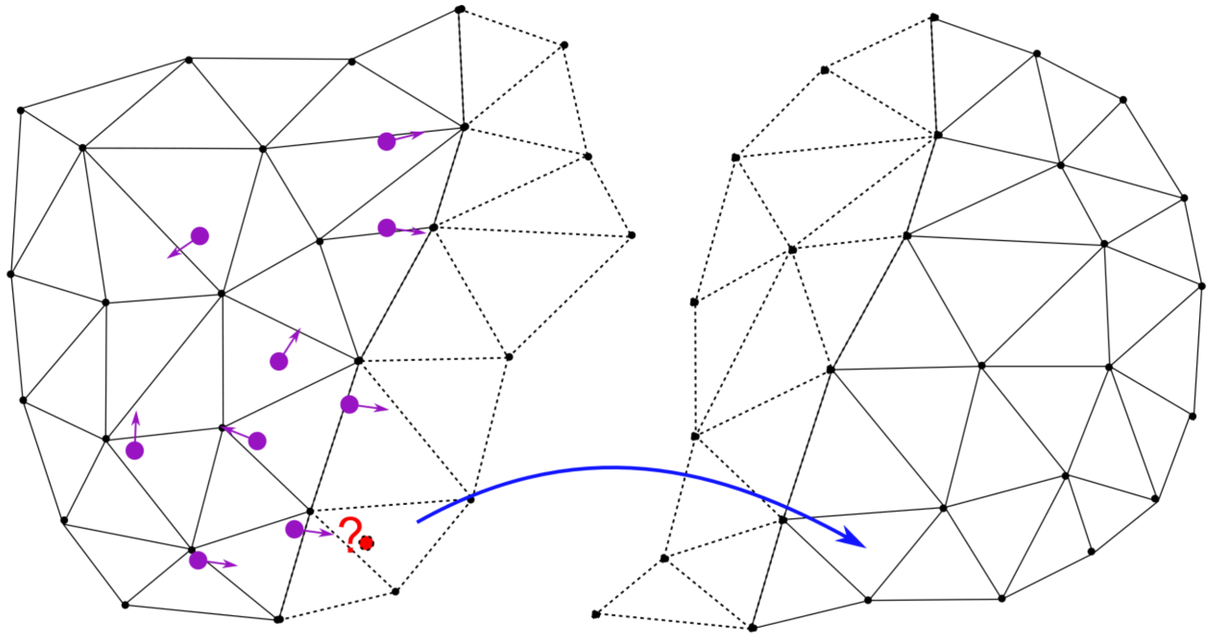
1. https://www.nektar.info
D. Moxey (King's College London); C.D. Cantwell, S.J. Sherwin (Imperial College London)

ExCALIBUR

NESO-Particles

Global Particle Movement

- Anywhere to Anywhere particle movement supported (2D and 3D).
- Implemented with halos + coarse grid.
- Tuneable local communication via variable halo width.

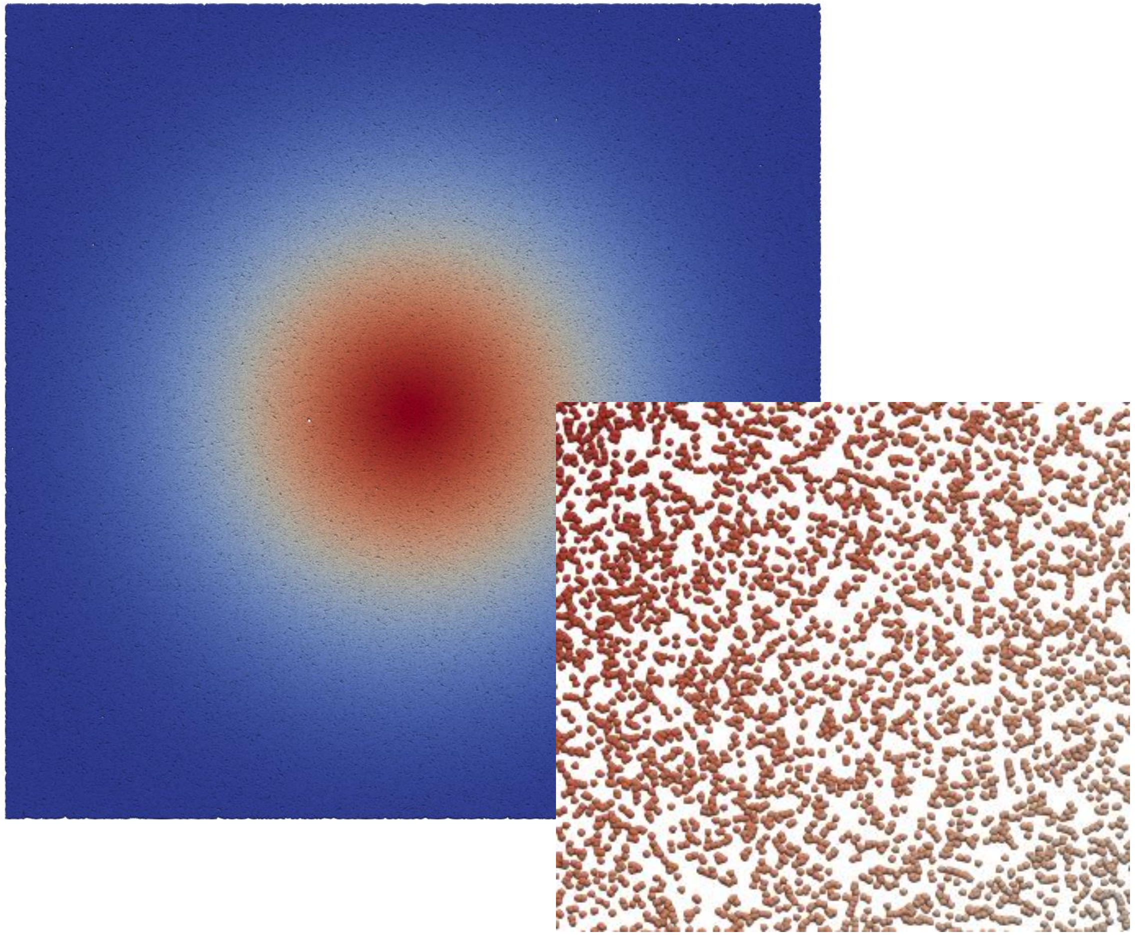


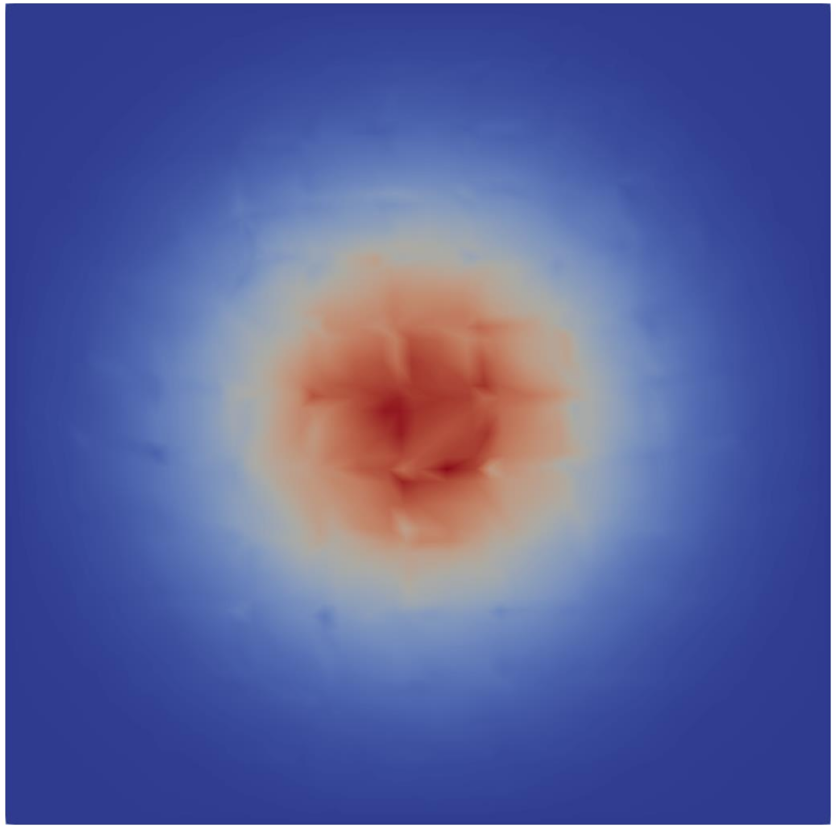
NESO:

- Bins particles into 2D and 3D elements.
- Caches reference positions (projection/evaluation).

Projection Example

500K particles, 10x10 Quadrilateral mesh





- Uniformly distributed positions
- Gaussian distributed weights

Projection

L2 Galerkin Projection

For particles indexed by i,

$$\hat{\rho}(\vec{r}) = \sum_i q_i \delta(\vec{r} - \vec{r}_i)$$

- Particle representation

seek a function ρ such that

$$\rho(\vec{r}) = \sum_{j=1} \alpha_j \psi_j$$

- Finite element representation.

where

$$\langle \rho - \hat{\rho}, \psi_j \rangle = 0 \ \forall \ j.$$

$$M\vec{\alpha} = \vec{\Psi},$$

Projection

L2 Galerkin Projection

$$M\vec{\alpha} = \vec{\Psi},$$

$$(\vec{\Psi})_j = \langle \hat{\rho}, \psi_j \rangle$$

$$= \sum_i q_i \int_\Omega \delta(\vec{r} - \vec{r}_i)\psi_j d\vec{x}$$

$$= \sum_i q_i \psi_j(\vec{r}_i)$$

- Dirac delta particle shape - No quadrature.

- Require evaluation of each basis function at each particle location.
- Implemented as SYCL kernels
- Static polymorphism (CRTP) for basis function types.
- CRTP as virtual functions are not device callable
- Given basis functions and DOFs – function evaluation is "easy".

Summary

Current status:

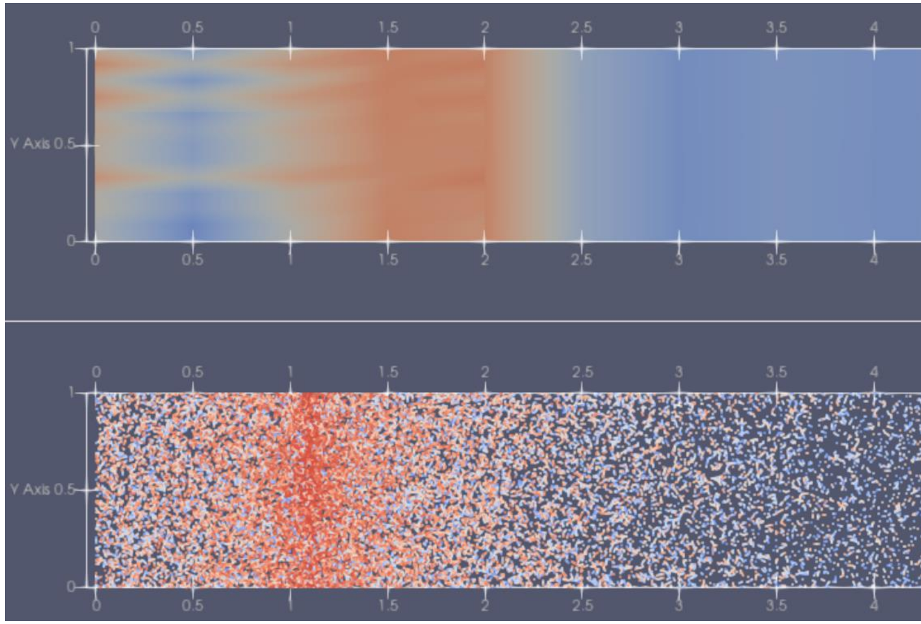
- Efficient particle coupling between finite elements and particles (project/evaluate).
- MPI+SYCL implementation (CPU + GPU execution).

In progress:

- Implementation of plasma turbulence models:
  1. Fluid approximation of plasma
  2. Kinetic Neutral species (particles)
  3. Plasma-Neutral coupling through project/evaluate
  4. Testing implementation using plasma turbulence problems

Continuous:

- Cycle of profile and improve implementations on CPU/GPU architectures.



ParticleLoop

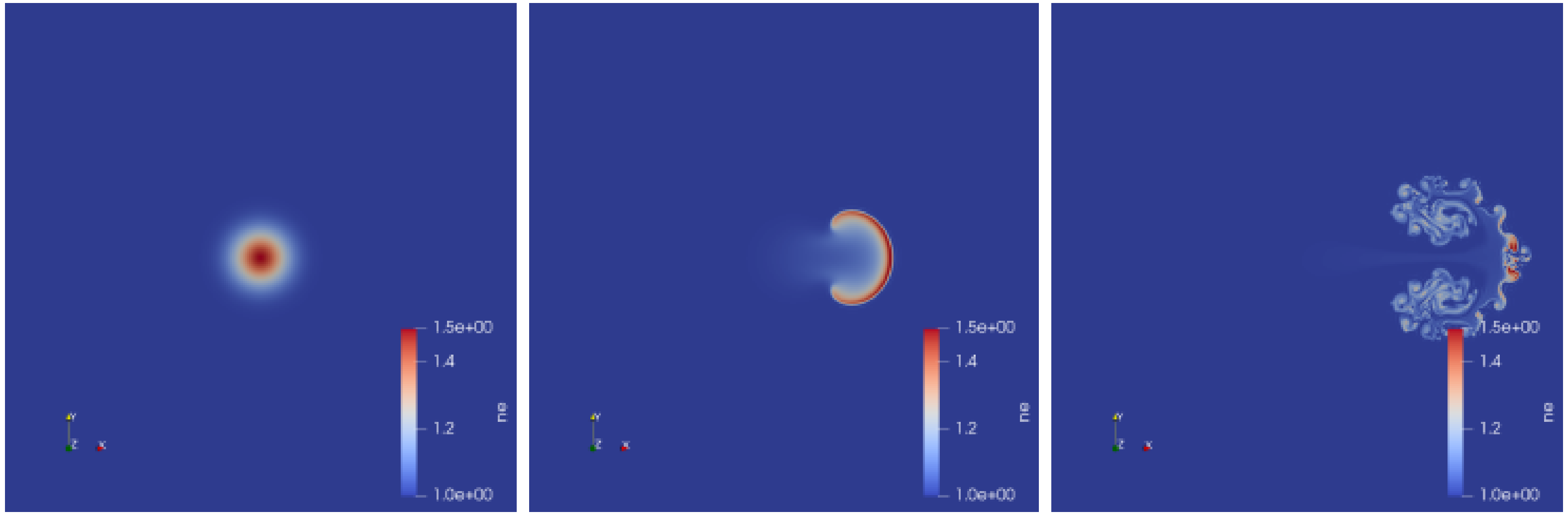
Key:
- (standard) SYCL
- NESO-Particles API
- User Kernel

```
auto k_P = (*particle_group)[Sym<REAL>("POSITION")]->cell_dat.device_ptr();
auto k_V = (*particle_group)[Sym<REAL>("VELOCITY")]->cell_dat.device_ptr();

auto pl_iter_range = ...; auto pl_stride = ...; auto pl_npart_cell = …;

sycl_target->queue
    .submit([&](sycl::handler &cgh) {
      cgh.parallel_for<>(
          sycl::range<1>(pl_iter_range), [=](sycl::id<1> idx) {
            NESO_PARTICLES_KERNEL_START
            const INT cellx = NESO_PARTICLES_KERNEL_CELL;
            const INT layerx = NESO_PARTICLES_KERNEL_LAYER;

            k_P[cellx][0][layerx] += 0.001 * k_V[cellx][0][layerx];
            k_P[cellx][1][layerx] += 0.001 * k_V[cellx][1][layerx];

            NESO_PARTICLES_KERNEL_END
          });
    })
    .wait_and_throw();
```

KERNEL_START/END are macros for CPU/GPU loop ordering

Next steps: 2D3V plasma proxyapp

2D plasma turbulence with neutral particle source terms	• Tight-coupled integration of the spectral / hp and particles. • Kinetic neutral species in plasma background. • Due by end Mar 2023.
Plasma turbulence in *Nektar++*	*Nektar++* [1] implementation of equations from existing *Hermes-3* code (finite difference) [2].
Neutral particles	Neutral particles do not feel confining magnetic field, but ionize as they interact with plasma – source terms in fluid equations (= coupling).



(a) $t = 0$ (b) $t = 2$ (c) $t = 4$

1. https://github.com/ExCALIBUR-NEPTUNE/nektar-driftplane 2. https://github.com/bendudson/hermes-3

SYCL Experience

- Thoughts:
  1. CI – want to retain portability across SYCL implementations / hardware
  2. Developer/user environments – code needs to run on the pseudorandom environments in the wild.
  3. SYCL_EXTERNAL – optional in standard. Nice to be able to write device functions.
  4. 2020 spec significantly improves usability (64bit atomics, usm)

- Works out-of-the-box:
  1. Profiling, vtune/nvprof
  2. Composition with MPI

Code Generation

- Kernel and loop structure are captured (can perform higher level optimisations).
- Execution method is now tuneable and not the concern of the domain specialist (separation of concerns)
- Scope to alter how kernels perform more complex operations (RNG, special functions) on different hardware.

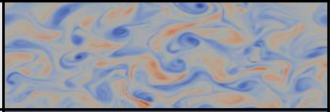
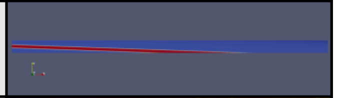
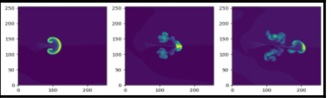
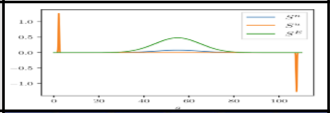
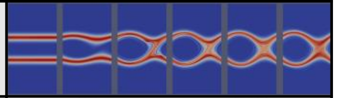
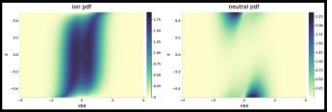
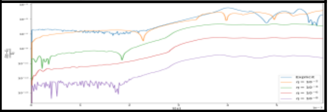
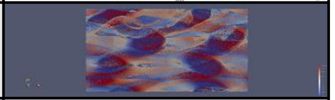
- Requires good abstractions:
    1. ParticleLoops, field deposition/evaluation are first steps.
    2. Neutral physics/molecular models are complex.

Code Generation – possible solution?

```
P = ParticleSymbol(..., "P"); V = ParticleSymbol(..., "V")
dt = Constant(0.001)

@kernel_inline
def dot_product_3d(a1, a2, a3, b1, b2, b3):
    return (a1 * b1) + (a2 * b2) + (a3 * b3)

@kernel_inline
def l2_squared_3d(a1, a2, a3):
    return dot_product_3d(a1, a2, a3, a1, a2, a3)

# Looping structure is captured
px = ParticleLoop()

@kernel
def k_euler(P, V):
    for dx in range(2):
        P[px, dx] = P[px, dx] + dt * V[px, dx]
    ke = l2_squared_3d(V[px,0], V[px,1], V[px,2])

Loop(k_euler, P,  V)
```

```
for (int dx = 0; dx < 2; dx+=1)
{
  P[neso_cellx][dx][neso_layerx] =
P[neso_cellx][dx][neso_layerx] +
0.001*V[neso_cellx][dx][neso_layerx];
}
auto a1_0 =
V[neso_cellx][0][neso_layerx];
auto a2_1 =
V[neso_cellx][1][neso_layerx];
auto a3_2 =
V[neso_cellx][2][neso_layerx];
auto a1_0_3 = a1_0;
auto a2_1_4 = a2_1;
auto a3_2_5 = a3_2;
auto b1_3_6 = a1_0;
auto b2_4_7 = a2_1;
auto b3_5_8 = a3_2;
auto ke = a1_0_3*b1_3_6 +
a2_1_4*b2_4_7 + a3_2_5*b3_5_8;
```

Proxyapps inventory

Proxyapp	Framework	Language	Comments	Sample output
nektar-driftwave	*Nektar++*	C++	2D Hasegawa-Wakatani equations	
nektar-diffusion	*Nektar++*	C++	strongly anisotropic diffusion	
vertical natural convection in spectral / hp, 2D and 3D	*Nektar++*	C++	incompressible Navier-Stokes with buoyancy	
2D plasma turbulence equations in spectral / hp	*Nektar++*	C++	*Hermes-3* equation system	
1D fluid solver with UQ and realistic boundary conditions	*Nektar++*	C++	1D model of scrape-off layer	
Vlasov-Poisson kinetic solver in spectral / hp	*Nektar++*	C++	due Dec 2022	
moment-kinetics	new code (Univ. Oxford)	Julia	moment-kinetic gyro-averaged code	
minepoch	*EPOCH* (Univ. Warwick)	Fortran	used for testing particle implementations	
electrostatic PIC proxyapp	NESO-Particles	C++ / SYCL	due Dec 2022	
2D3V coupled fluids-neutral particles proxyapp	NESO-Particles	C++ / SYCL	due Mar 2023	coming soon

Community overview

UKAEA TEAM	Rob Akers, Wayne Arter, Matthew Barton, James Cook, John Omotani, Joseph Parker, Owen Parry, Will Saunders, Ed Threlfall.
UKRI GRANTS	• University of Exeter (VVUQ, surrogate models): Peter Challenor, Tim Dodwell, Louise Kimpton. • King's College London (Nektar++): Mashy Green, David Moxey. • Imperial College London (Nektar++): Chris Cantwell, Bin Liu, Spencer Sherwin. • University of Oxford: Michael Barnes, Patrick Farrell, Michael Hardman. • STFC Hartree Centre: Vasil Alexandrov, Hussam al-Daas, Tyrone Rees, Emre Sahin, Andrew Sunderland, Sue Thorne. • University College London (VVUQ): Kevin Bronik, Peter Coveney, Matt Graham, Serge Guillas, Tuomas Koskela, Yiming Yang. • University of Warwick (DSLs): Gihan Mudalige. • University of York (plasma physics, support & coordination, DSLs): David Dickinson, Ed Higgins, Chris Ridgers, Steven Wright.
ALUMNI	• University of Oxford: Felix Parra-Diaz. • University of Warwick (EPOCH): Ben McMillan, Tom Goffrey. • University of York: Ben Dudson.
OUTPUT (INC. CODE)	• Proxyapps code (MIT licence): see repositories on https://github.com/ExCALIBUR-NEPTUNE (some, inc. NESO and NESO-Particles, are public). • Large body of supporting documents and reports : https://github.com/ExCALIBUR-NEPTUNE/Documents (currently private). • Developer website in development.

Participation welcomed!