Science and
Technology
Facilities Council

UKAEA NEPTUNE Mathematical Support Project

Hussam Al Dass[1], Niall Bootland[1], Tyrone Rees[1], Andrew Sunderland[2], Sue Thorne[2]

[1]UKRI-STFC Rutherford Appleton Laboratory
[2]UKRI-STFC Hartree Centre

Who we are

Hussam

STFC

Rutherford Appleton Laboratory (RAL)

Hartree Centre

Sue

Numerical Linear Algebra

Data analytics

AI

HPC

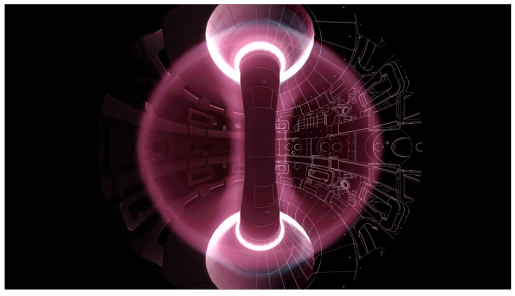
Large-scale optimisation

Niall

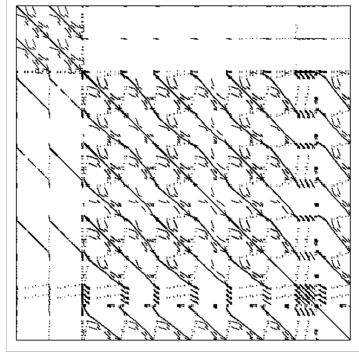
Tyrone

UKRI Science and Technology Facilities Council

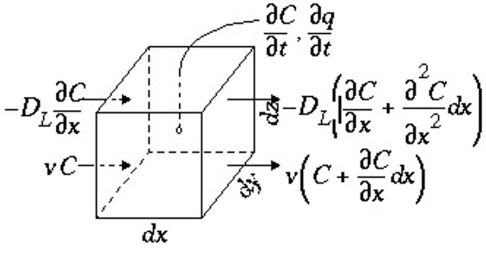
The big picture

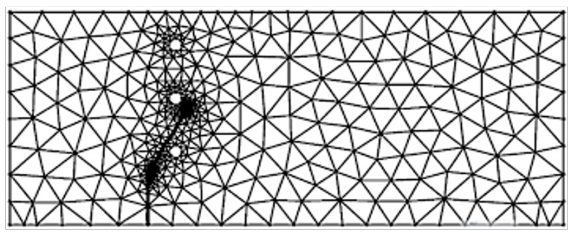


$$\frac{du_1}{dx} = \frac{-3u_1 + 4u_2 - u_3}{2h}$$
$$\frac{du_i}{dx} = \frac{-u_{i+1} + u_{i+1}}{2h},\ i = 2(1)N-2$$
$$\frac{du_{N-1}}{dx} = \frac{u_{N-3} - 4u_{N-2} + 3u_{N-1}}{2h}.$$



Mathematical Model → Discretisation → Solve Linear Systems

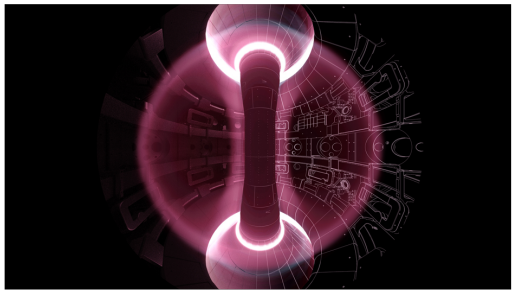




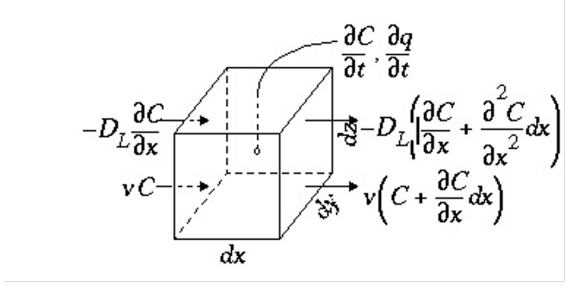
```
procedure pcg(A, Θ^k, b̄)
1    Compute Δy_0 as a guess of Δy^k;
2    r_0 := b̄ − AΘ^k A^⊤ Δy_0;
3    solve M z_0 = r_0;
4    p_0 := z_0;
5    i := 0;
6    do stopping criterion not satisfied →
7        q_i := AΘ^k A^⊤ p_i;
8        α_i := z_i^⊤ r_i / p_i^⊤ q_i;
9        Δy_{i+1} := Δy_i + α_i p_i;
10       r_{i+1} := r_i − α_i q_i;
11       solve M z_{i+1} = r_{i+1};
12       β_i := z_{i+1}^⊤ r_{i+1} / z_i^⊤ r_i;
13       p_{i+1} := z_{i+1} + β_i p_i;
14       i := i + 1
15   od;
16   Δy^k := Δy_i
end pcg;
```

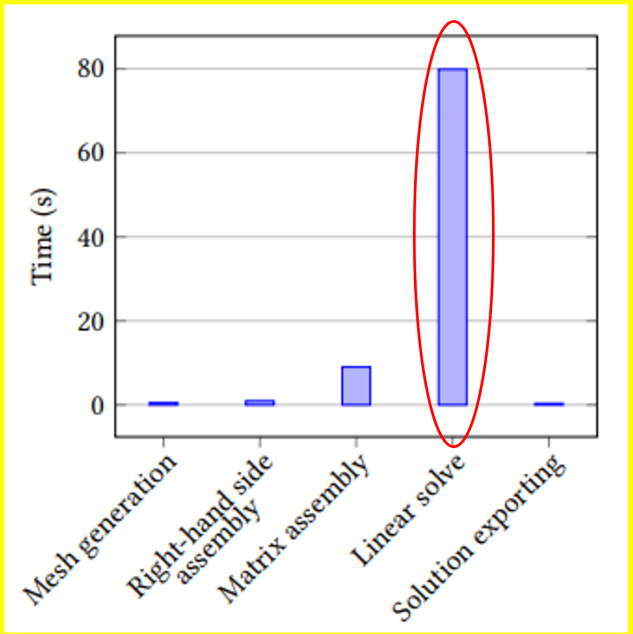
Science and Technology Facilities Council

The big picture

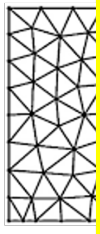


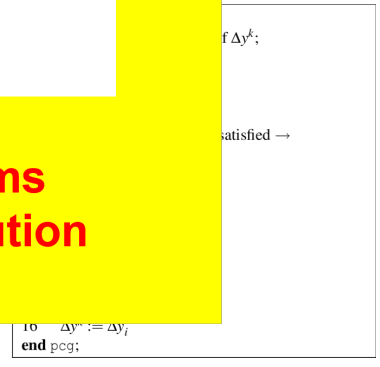
Mathematical Model

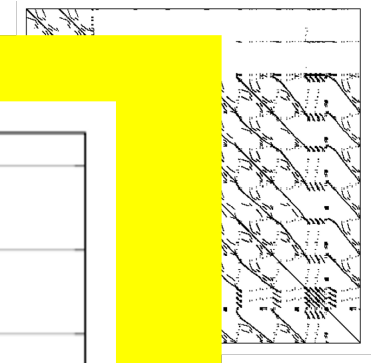




Solving linear systems dominates time to solution







inear
ms

Science and Technology Facilities Council

Time dependent solvers

We consider fully IRK methods

- high order accuracy
- require solving classic LS as in backward Euler

Solving linear systems – preconditioning

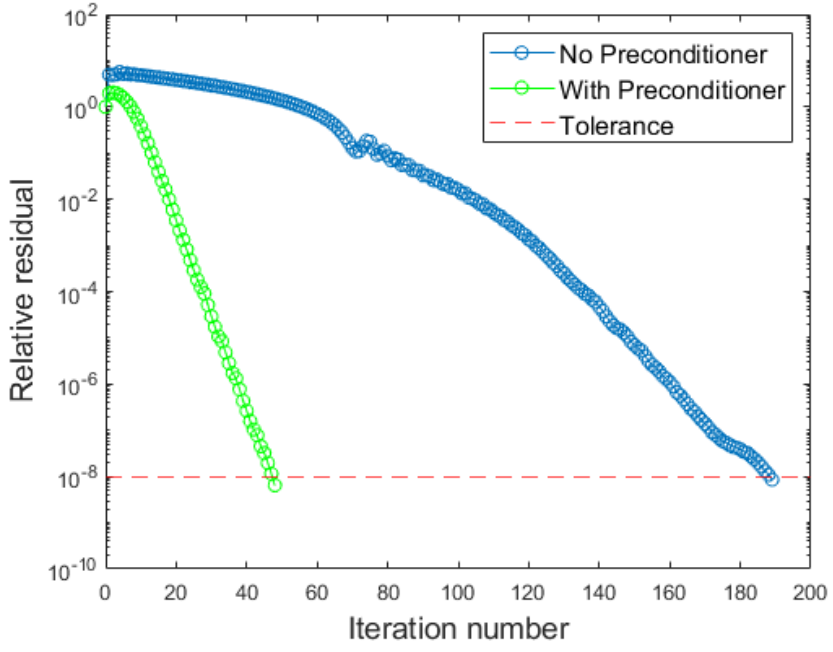
- Linear systems from discretised PDE models are typically very large and sparse
- Iterative methods are best suited to solve them
- However, properties degrade for real problems of interest (ill-conditioning prohibits convergence)
- Poor convergence avoided by having a good **preconditioner**

Solve

$$Ax = b$$

vs

$$\mathbf{P}Ax = \mathbf{P}b$$



Science and Technology Facilities Council

High-order finite elements

Solving is particularly challenging for high-order finite elements

Higher p gives better accuracy (exponential vs linear in # DOFs) but increased cost

	Complexity	Low order ($p = 1$)	High order ($p = 8$)
DOFs per element	$\mathcal{O}(p^d)$	~ 1	$\sim 8^d$
# of couplings	$\mathcal{O}(p^d)$	~ 1	$\sim 8^d$
non-zeros (memory)	$\mathcal{O}(p^{2d})$	~ 1	$\sim 64^d$
matrix-free memory	$\mathcal{O}(p^d)$	~ 1	$\sim 8^d$
matrix-free action	$\mathcal{O}(dp^{d+1})$	$\sim d$	$\sim d8^{d+1}$

Preconditioners for HO FEM

Leverage well-established preconditioners such as Algebraic Multigrid and Domain Decomposition methods.
Should avoid as possible assembling the HO operator matrix
Options

- use p-hierarchy of grids
- use the low-order operator matrix assembled on a refined mesh

If assembled, static condensation should be used

Synthetic results

Diffusion

Tests on orders up to $p = 13$, anisotropy (coef $10^{-6} - 1$, mesh $0.3 - 1$)

- Homogeneous diffusion: pMG scales perfectly
- Anisotropic diffusion (coef) : LOR AMG scales perfectly
- Anisotropic diffusion (mesh) : LOR AMG scales perfectly
- Anisotropic diffusion (coef&mesh): tough

Advection-diffusion

Tests on orders up to $p = 8$, viscosity $10^{-4} - 1$

- LOR AMG scales well

Numerical experiments

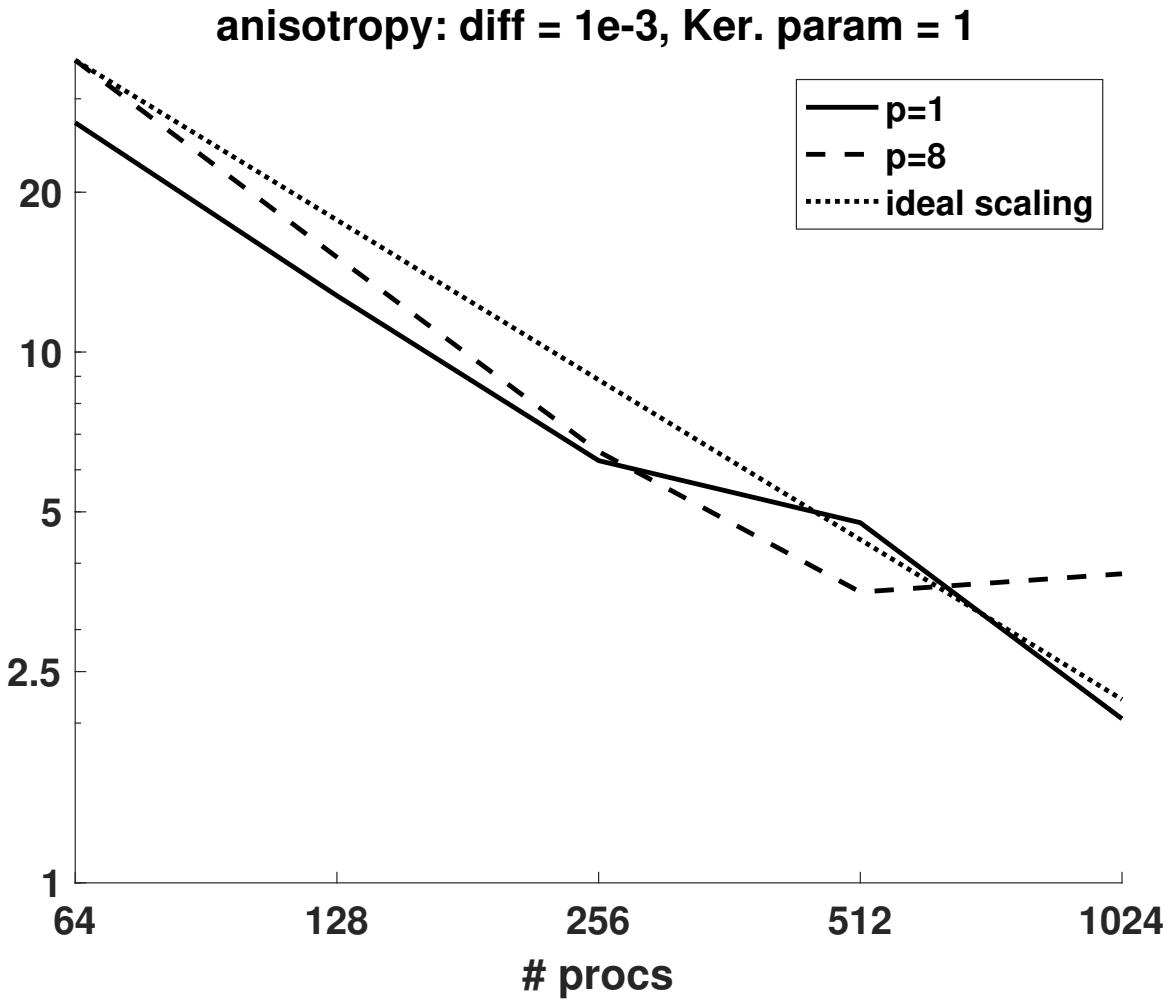


Figure: #DOFs $\approx$ 67M. # iter$_{p=1}$ = 11. # iter$_{p=8}$ = 15

Science and Technology Facilities Council

Numerical experiments

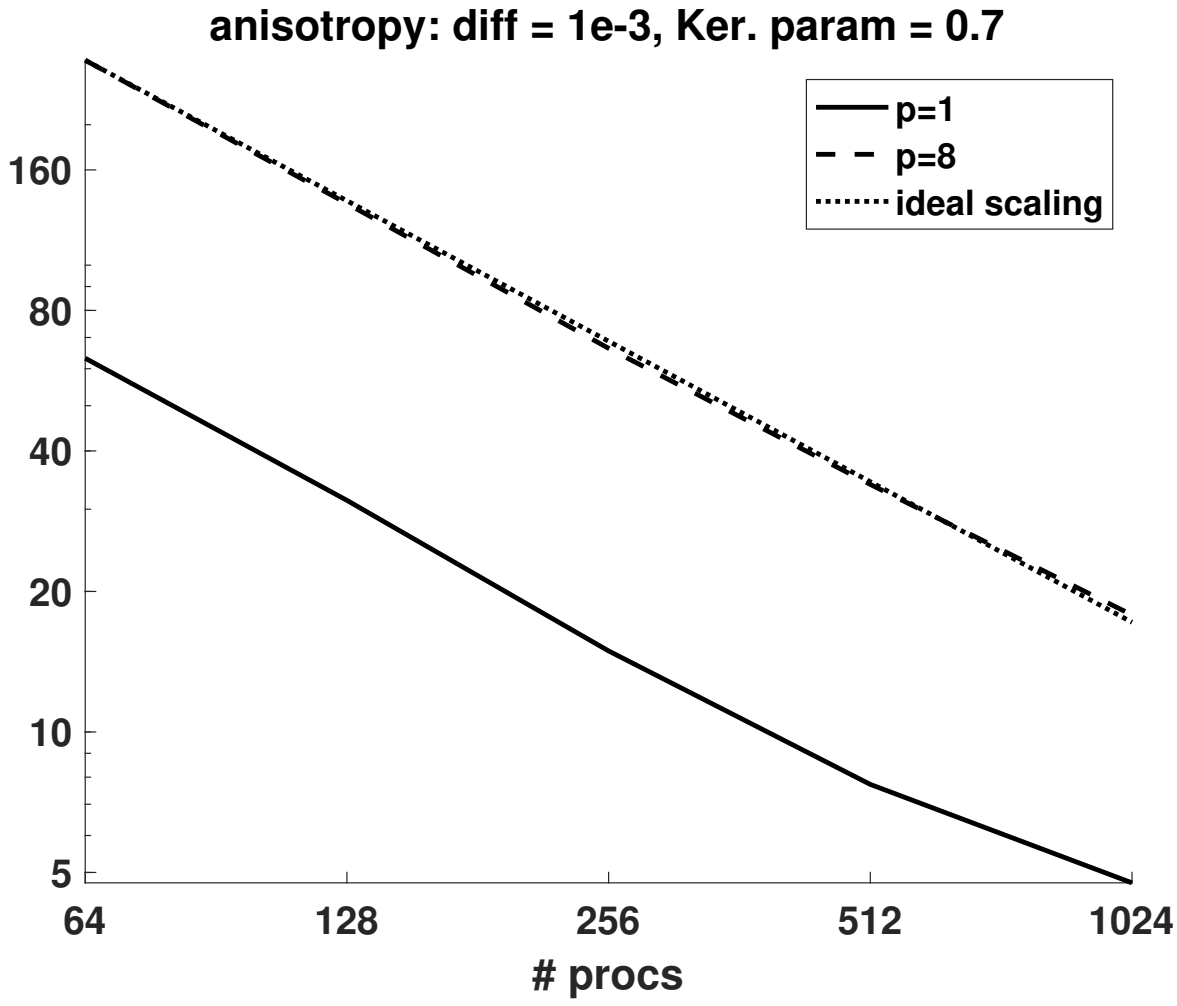


Figure: #DOFs $\approx$ 67M. # iter$_{p=1}$ = 53. # iter$_{p=8}$ = 317

Science and Technology Facilities Council

Numerical experiments

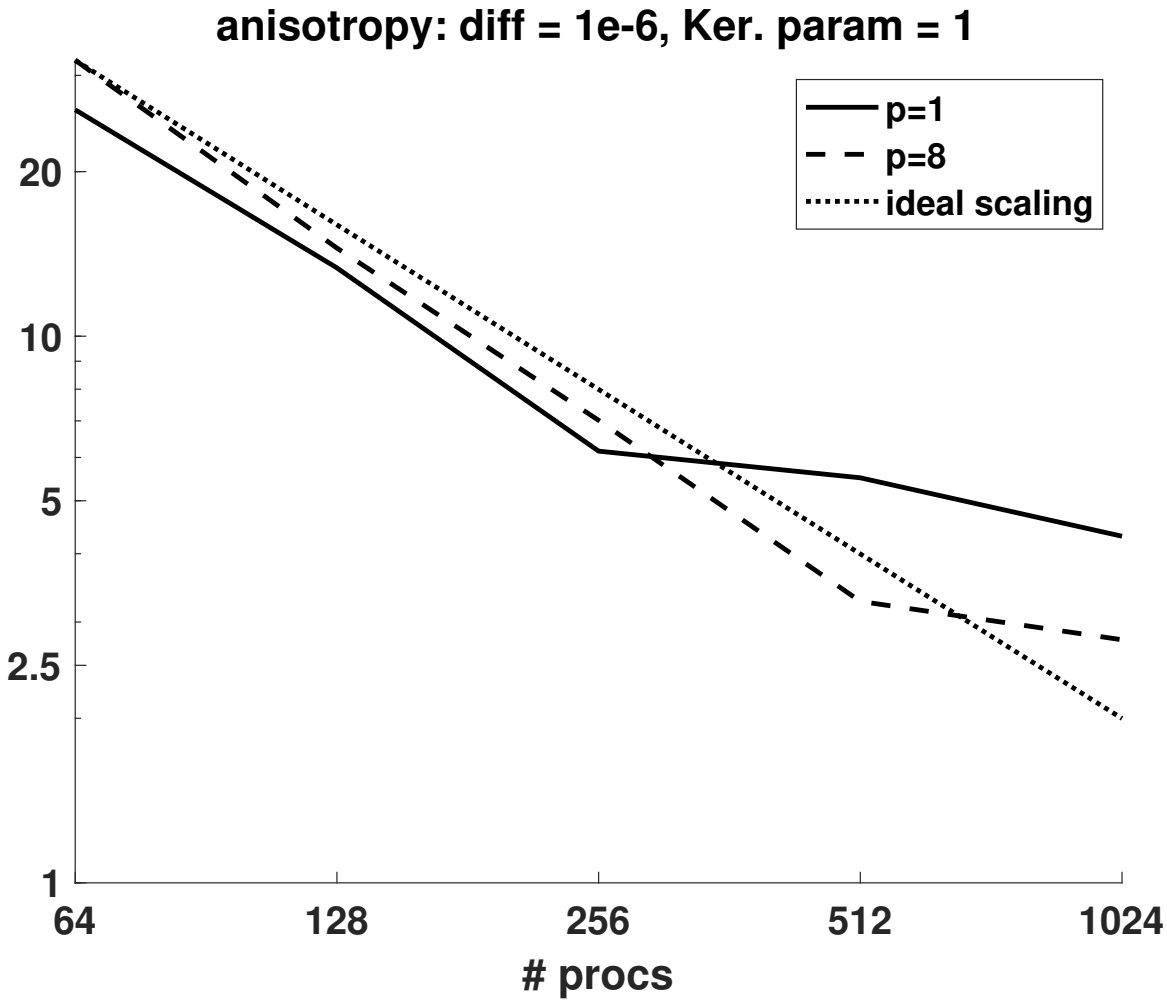


Figure: #DOFs $\approx$ 67M. # iter$_{p=1}$ = 11. # iter$_{p=7}$ = 17

Science and Technology Facilities Council

Numerical experiments

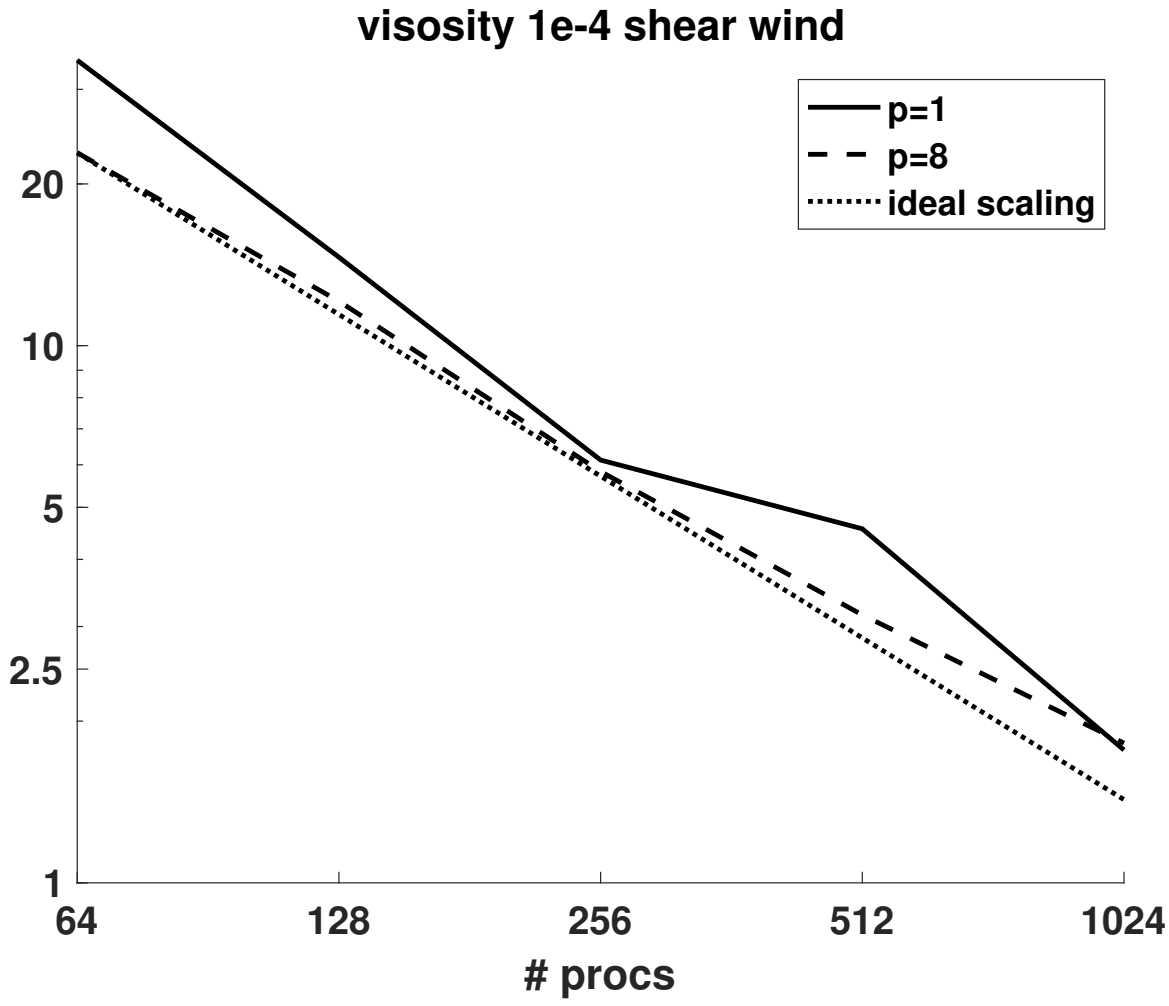


Figure: #DOFs $\approx$ 67M. # iter$_{p=1}$ = 20. # iter$_{p=8}$ = 17

Science and Technology Facilities Council

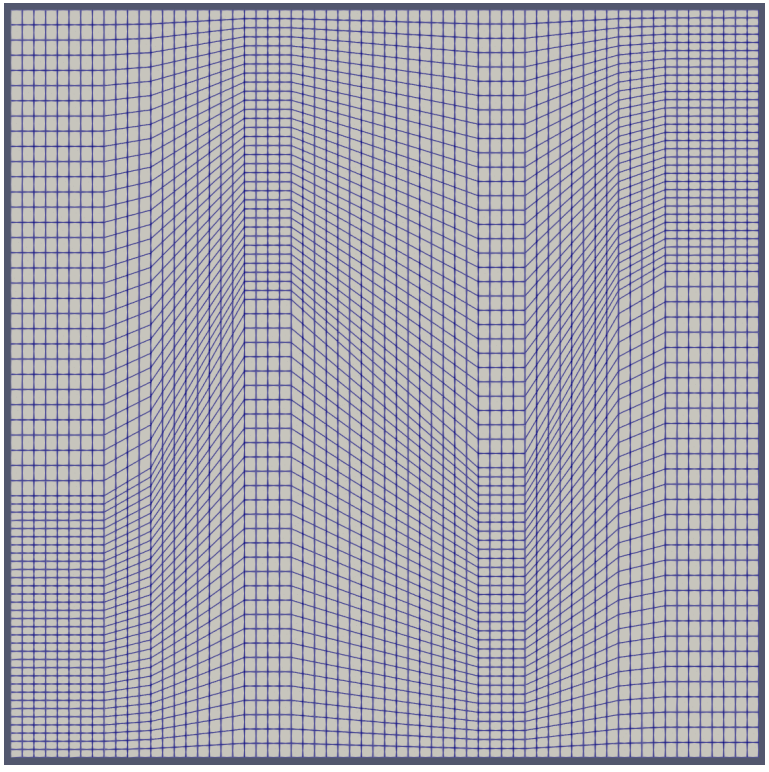
Code coupling Fluid–Particle

- Suggested Spatial Hybridization (SpH) Approach
- Identified several lightweight and efficient coupling software (MUI, CWIPI, preCICE)

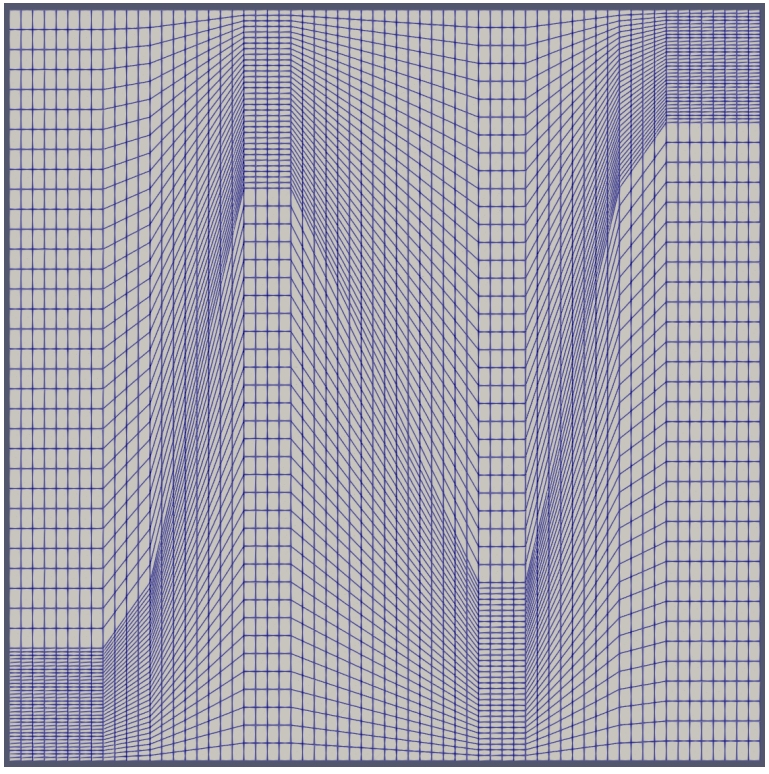
Since NESO particle-in-cell solver not yet available in Nektar++, current proxyapp plan is to represent particle-in-cell solver by sampling a travelling sine wave, the fluid model by an advection equation, in Nektar++

- 1-D model with variable overlap region
- At each timestep combine particle contribution with existing density using an indicator function

Kershaw mesh



(a) Ker. param: 0.7



(b) Ker. param: 0.3

Figure: Anisotropic mesh

Science and Technology Facilities Council