Future Directions

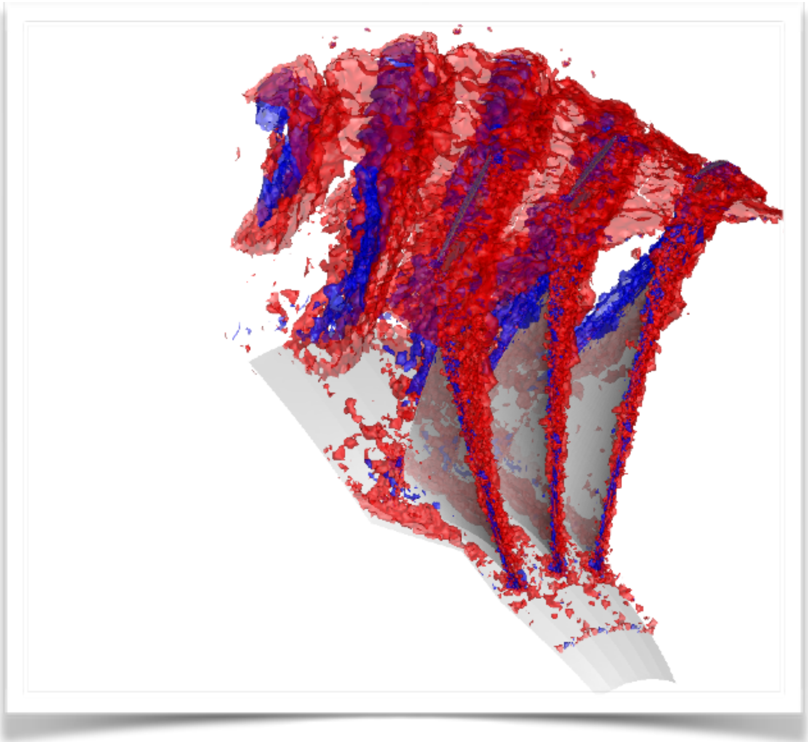
In the pipeline:

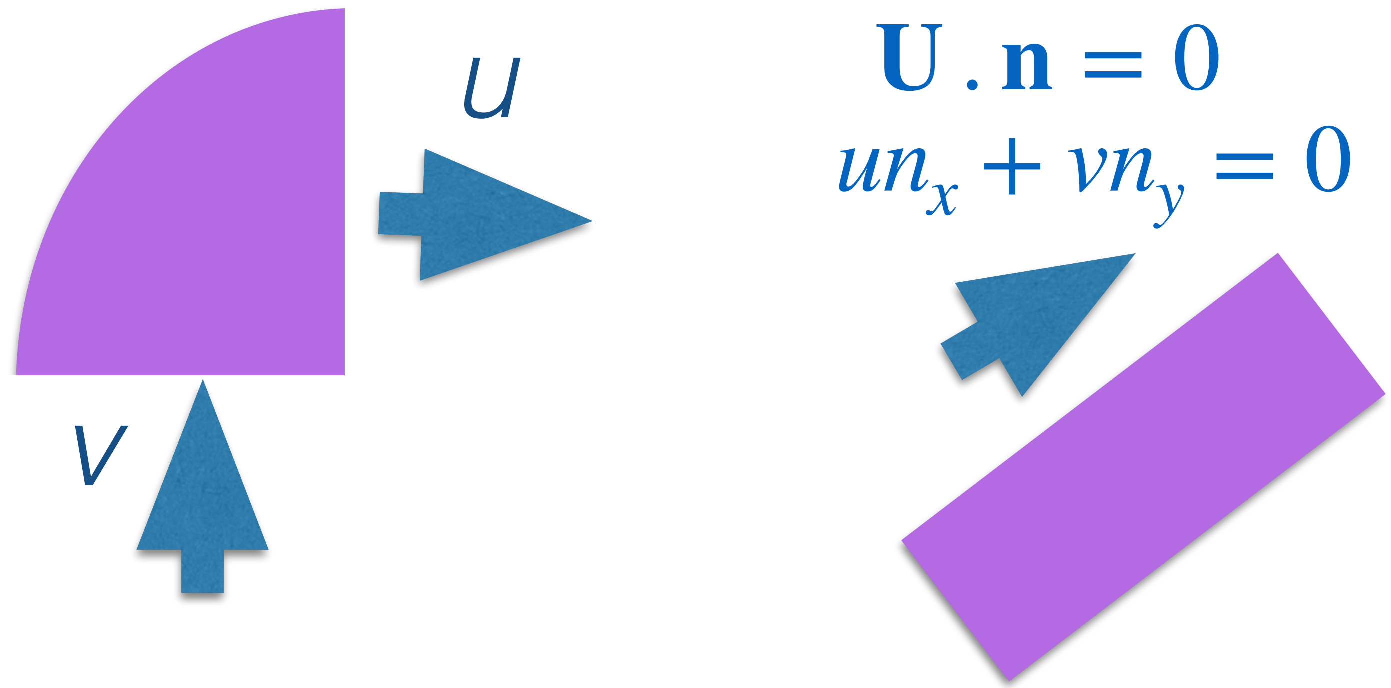
- Library
- Solver Capabilities
- Outreach

In the pipeline: Library

- Vector expansions
  - Multiregions refactoring
  - Linear solvers
- Acceleration
- Fault Tolerance

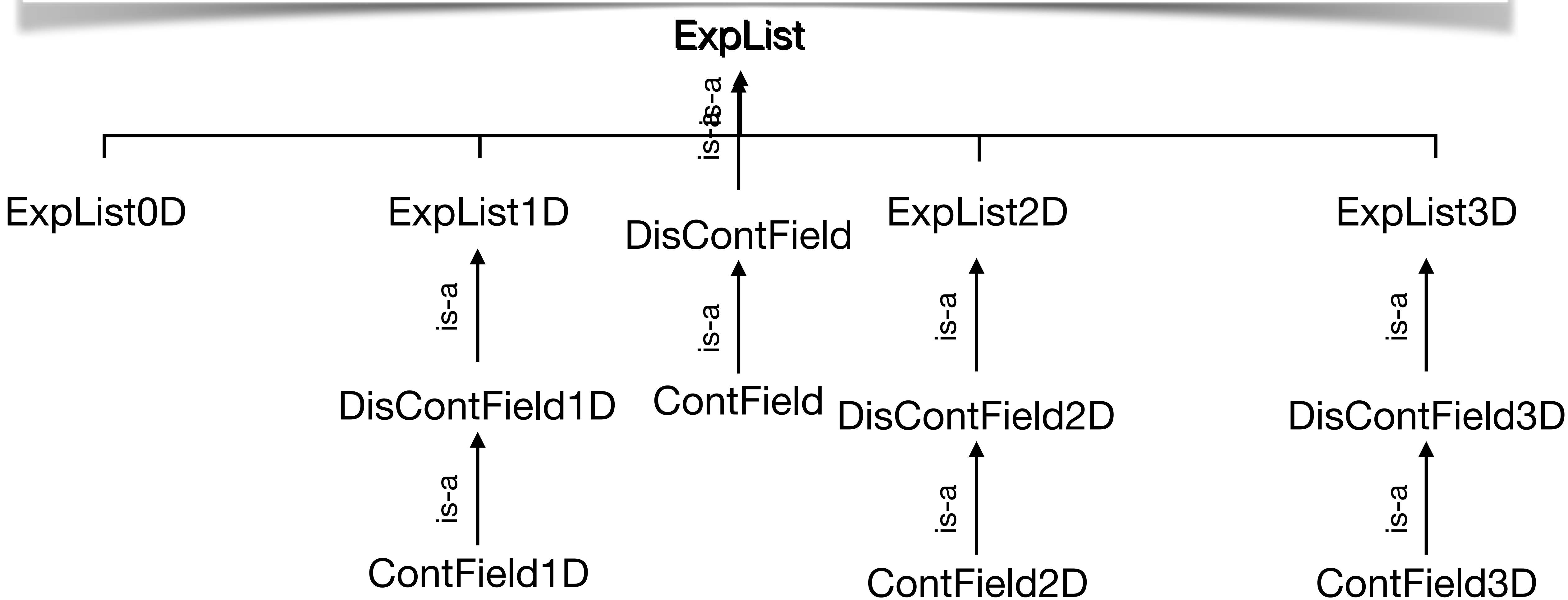
Vector Expansions





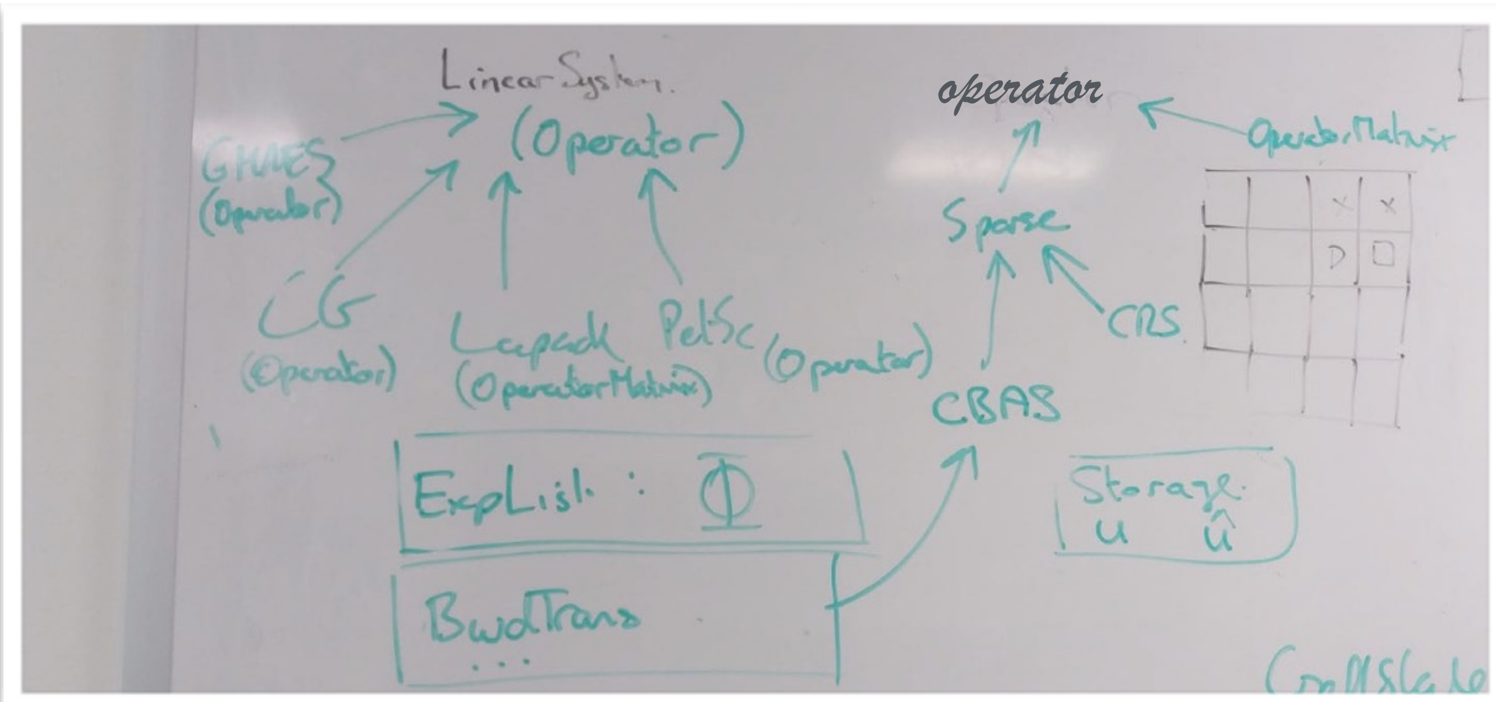
- Rotated periodic, no permeability BCs couple components of vector field.
- Scalar variables (pressure) already implemented.
- Vector variables (velocity) most easily done in iterative solver
- Would allow possibility of propellor, wind turbine modelling too.

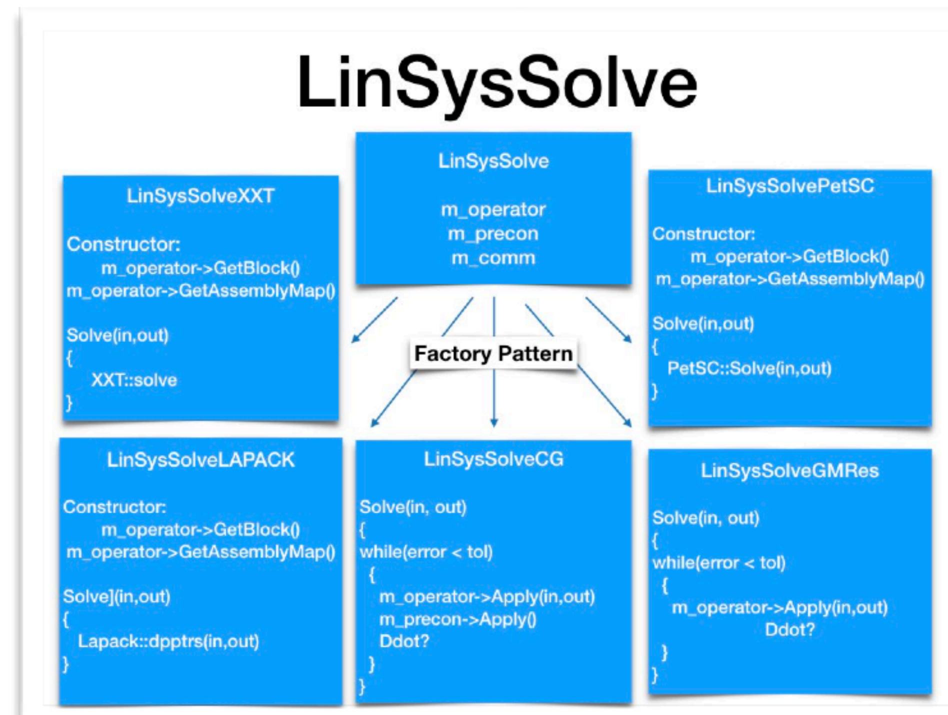
Multiregions restoring

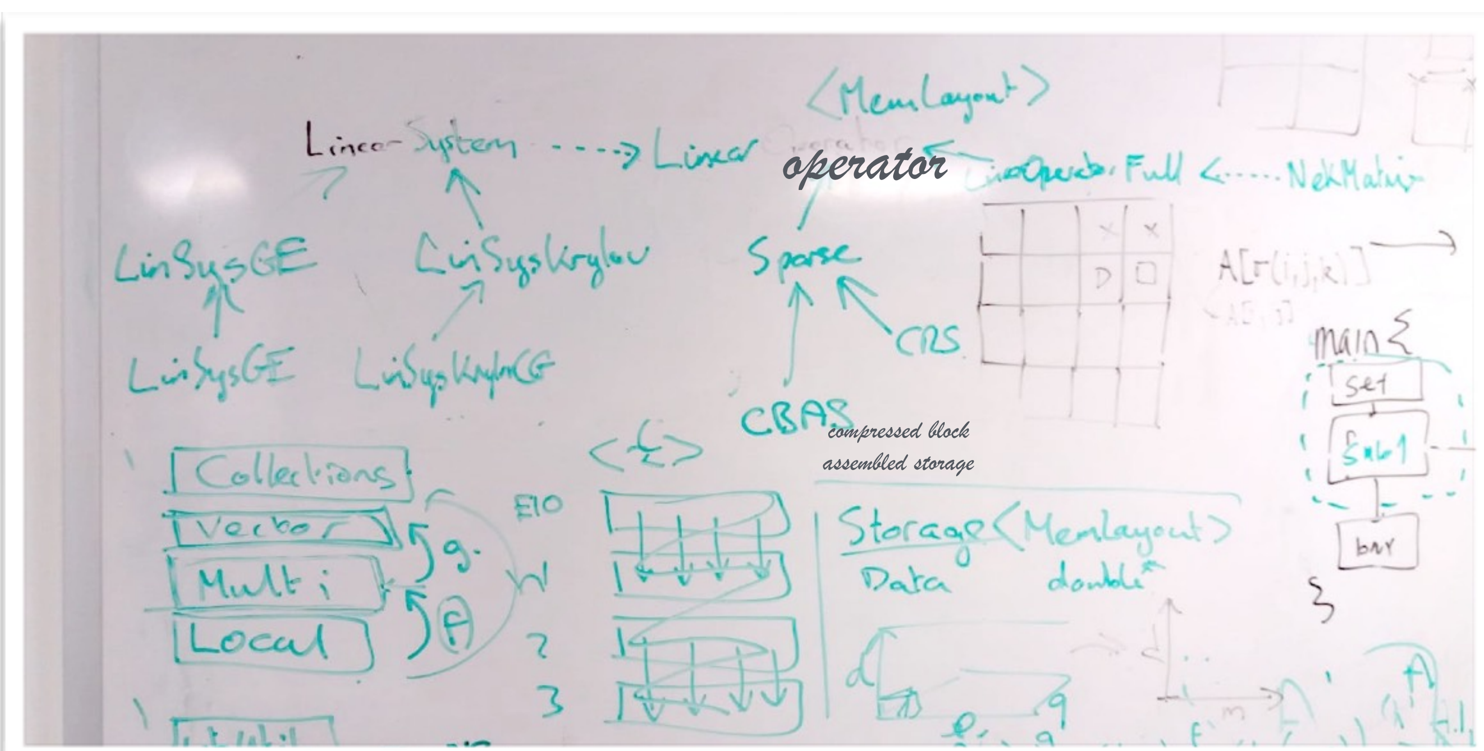


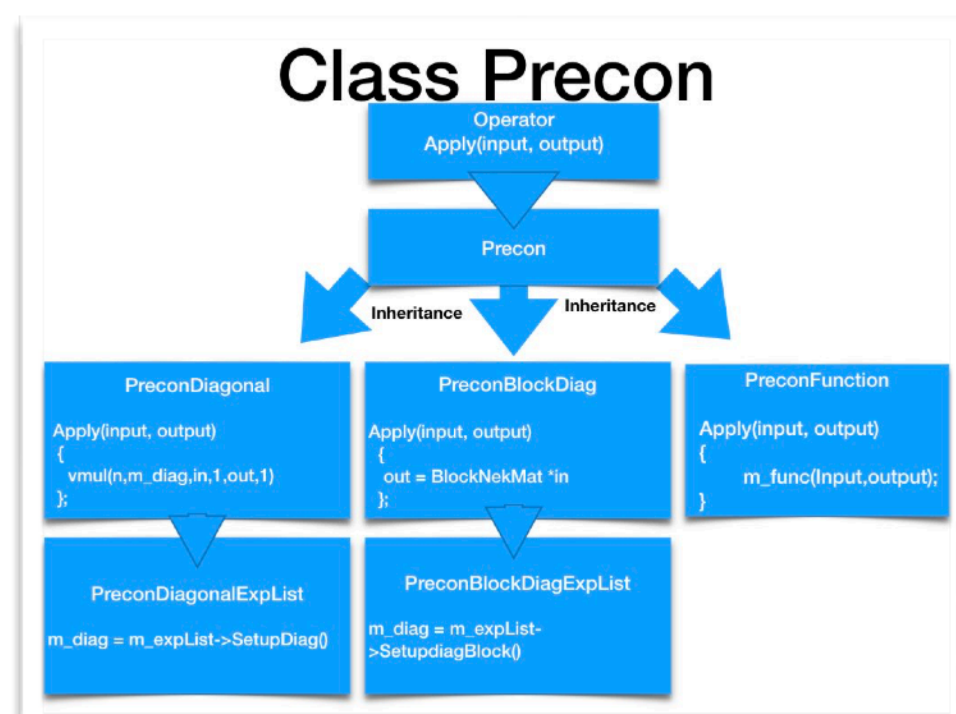
- MultiRegions: ExpList simplified
- constructors reduced from 36 to 6.
- Some changes within SolverUtils

Linear Algebra Solver

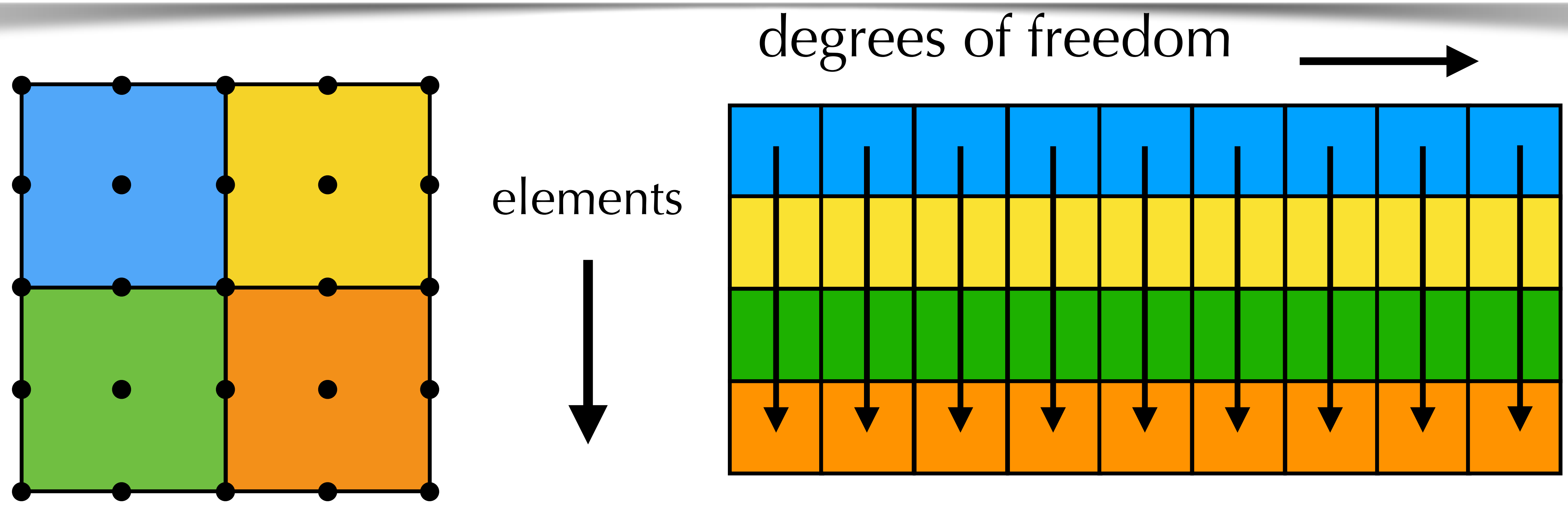


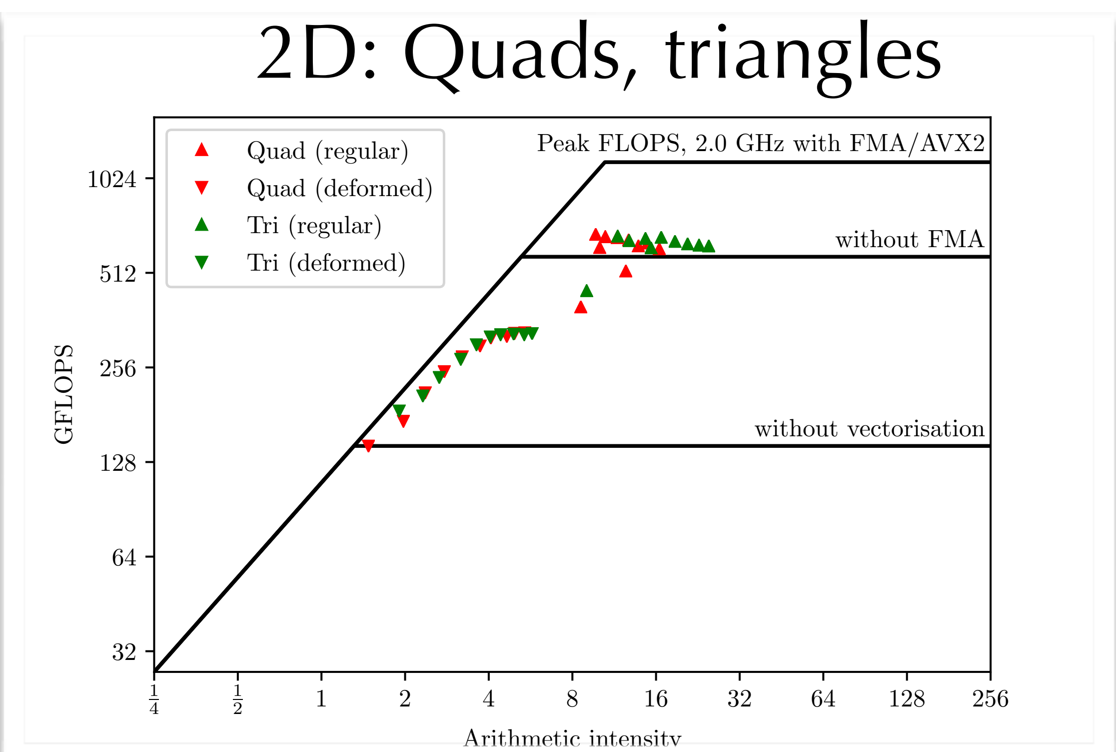


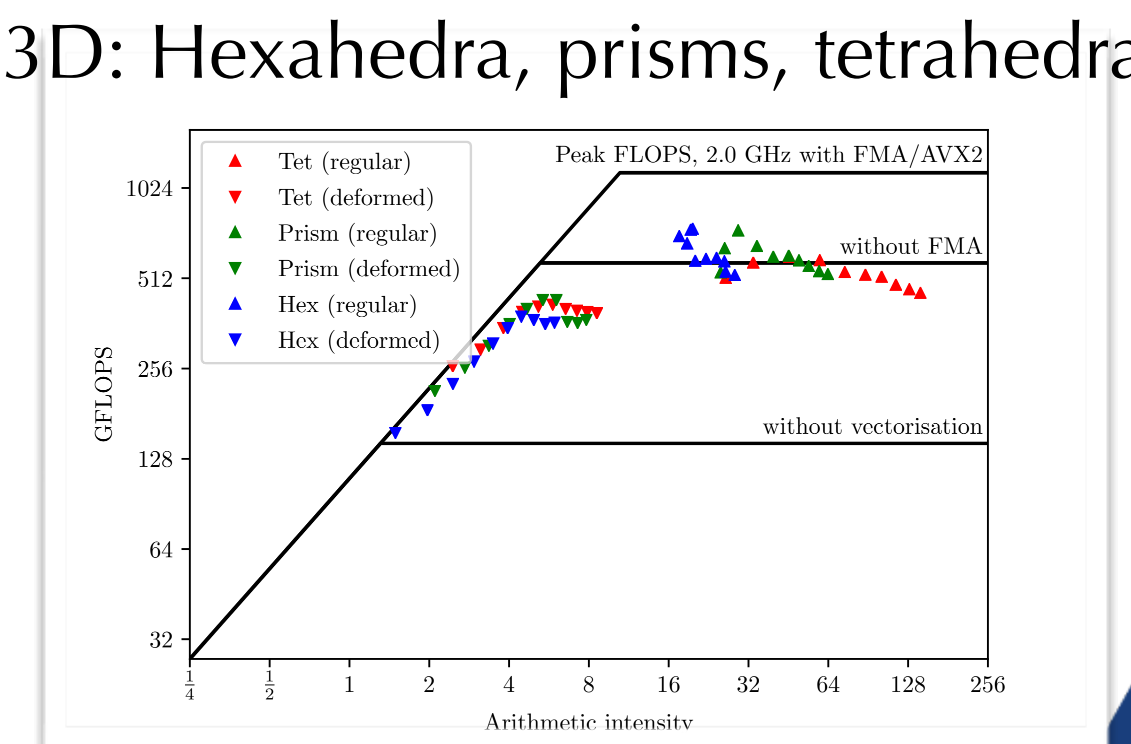




Acceleration/Memory Layout





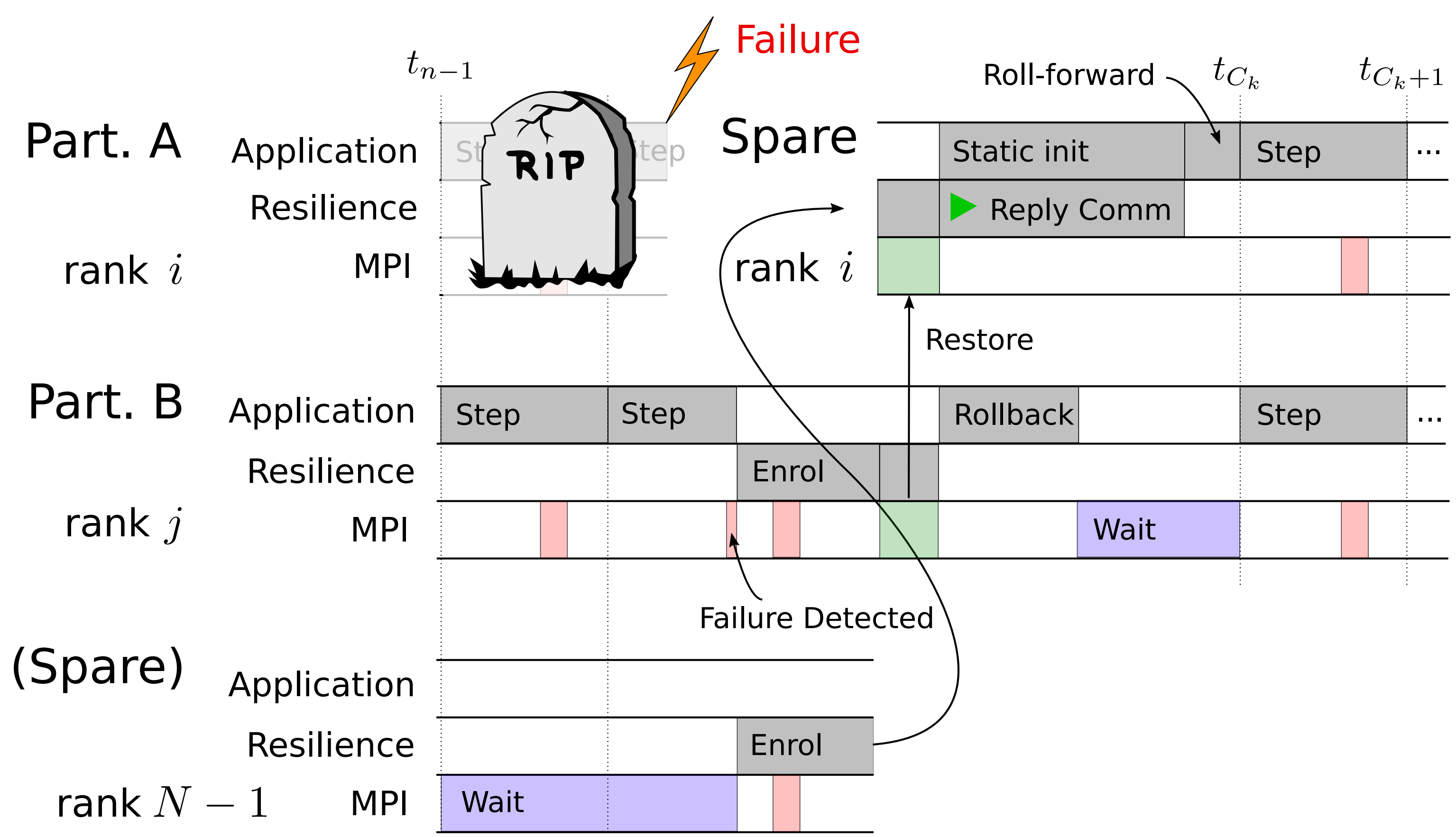


Moxey, Amici, Kirby "Efficient matrix-free high-order finite element evaluation for simplical elements" Under review SIAM J.Sci Comp,

Fault Tolerance/Resilience

Example	PF	Mean time to interrupt	
Titan	27.11	173 hours	[1]
Blue Waters (CPU-only)	5.66	8.6 hours	[2]
Tianhe-2 (8k nodes)	17.30	2 hours	[3]
(Exascale)	1000	$<$ 1 hour ?	[4]

Algorithm: Recovery



Cantwell C, Nielsen A, 2018, A minimally intrusive low-memory approach to resilience for existing transient solvers, *Journal of Scientific Computing*, Pages: 1-17

HDF5 Geometry

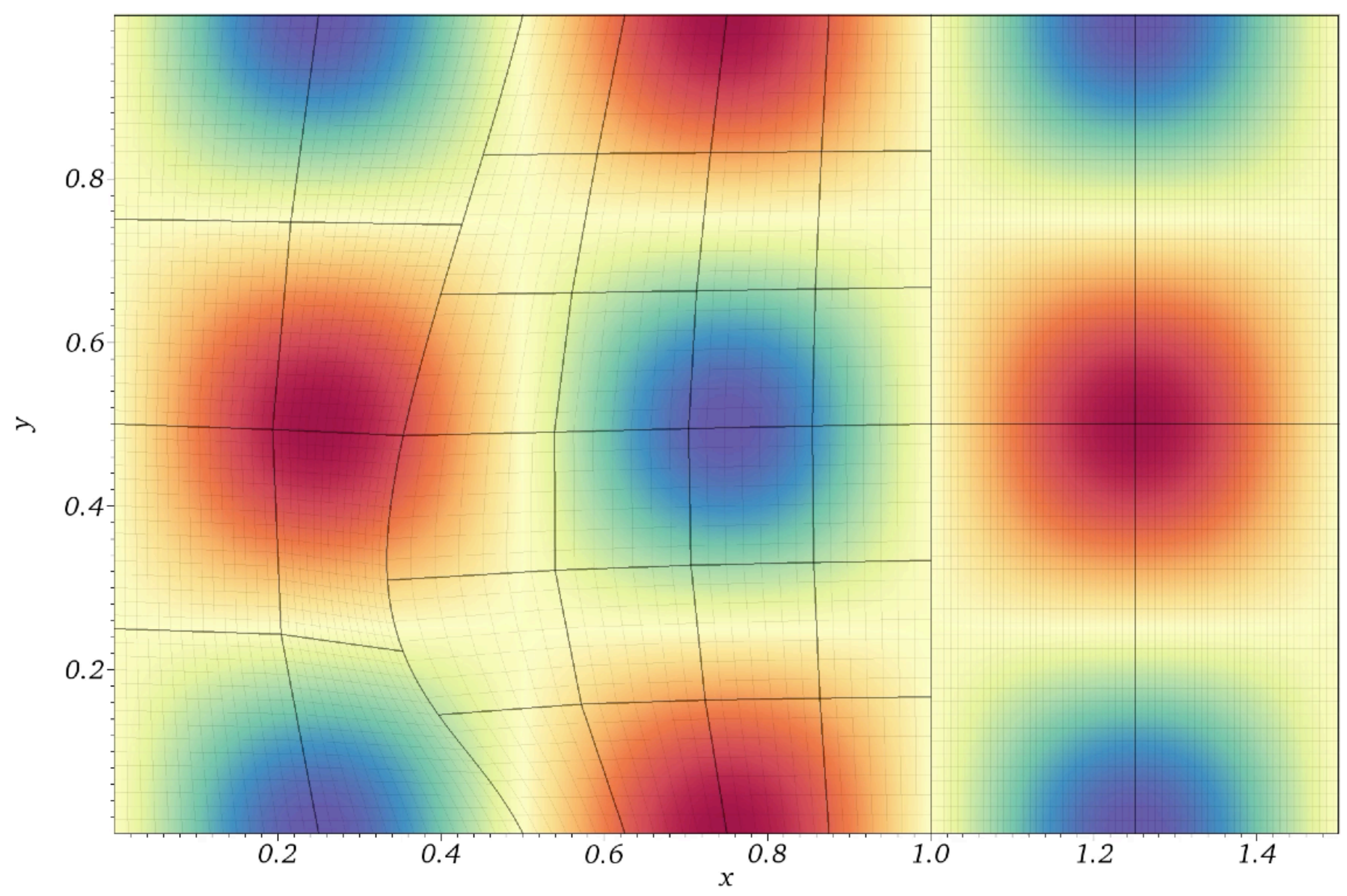
- Had severe limitations on big meshes: > 10K partitions, 10M elements
- Key bottleneck is xml format
  - Slow/conflicted reading
  - Partition then requires a write
- Nek 5.0 has introduced binary based hdf5 format
  - Parallel partitioning ptscotch
  - Maintained xml backwards compatibility
- **Intent to move to hdf5 as default so please consider enabling on your compilation**

In the pipeline: Capabilities

- Sliding meshes
  - *(Session 4, Edward Laughton)*
- Implicit solver
  - *(Session 4, Zhenguo Yan)*
- NekMesh
  - (*Session 3, Joaquim Peiro*)

Sliding Mesh

Edward Laughton (Exeter): Non-conformal mesh interfaces in 2D with the discontinuous Galerkin method



Implicit solver

Zhenguo Yan (Imperial): Development of implicit compressible flow solver in Nektar++

Jacobian-free Newton Krylov method (JFNK)

We need to solve

$$\mathbf{N}\left(\mathbf{u}^{n+1,m}\right) = \mathbf{u}^{n+1,m} - \mathbf{S}_m - \alpha_{mm}\mathbf{F}_m = \mathbf{0} \qquad (\mathbf{u}^{n+1,m,0} = \mathbf{S}_m), \qquad (12)$$

JFNK method is used for solving the nonlinear system

- Newton method: solving nonlinear system iteratively

$$\left(\frac{\partial \mathbf{N}}{\partial \mathbf{u}}\mathbf{P}^{-1}\right)\mathbf{P} \triangle \mathbf{u}^{n+1,m,l} = -\mathbf{N}\left(\mathbf{u}^{n+1,m,l}\right) \qquad (13)$$

- Jacobian-free, linearize nonlinear equation in each Newton iteration

$$\frac{\partial \mathbf{N}}{\partial \mathbf{u}} \cdot \mathbf{q} = \frac{\mathbf{N}(\mathbf{u}^{n+1,m,l} + \epsilon_{JF}\mathbf{q}) - \mathbf{N}(\mathbf{u}^{n+1,m,l})}{\epsilon_{JF}}, \quad \mathbf{N}(\mathbf{u}^{n+1,m,l}) \ \textit{stored} \qquad (14)$$

- Krylov method to solve linear system:
  GMRES, restarted every 30 iterations
- Preconditioner, approximate block Jacobi inversion

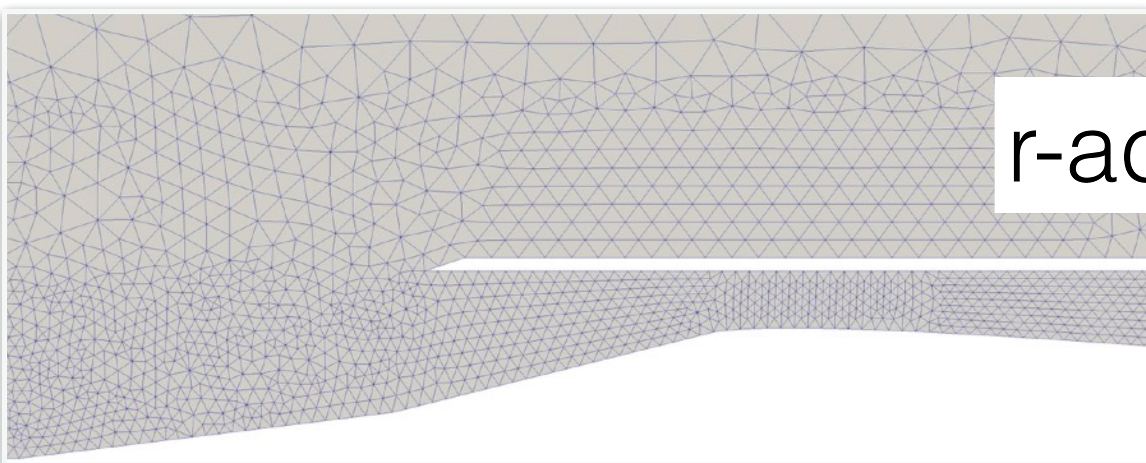
$$\hat{\mathbf{q}}^{k+1} = \mathbf{D}^{-1}\left(\mathbf{q} - (\mathbf{L} + \mathbf{U})\,\hat{\mathbf{q}}^{k}\right), \quad = \mathbf{D}^{-1} \ \textit{stored} \qquad (15)$$

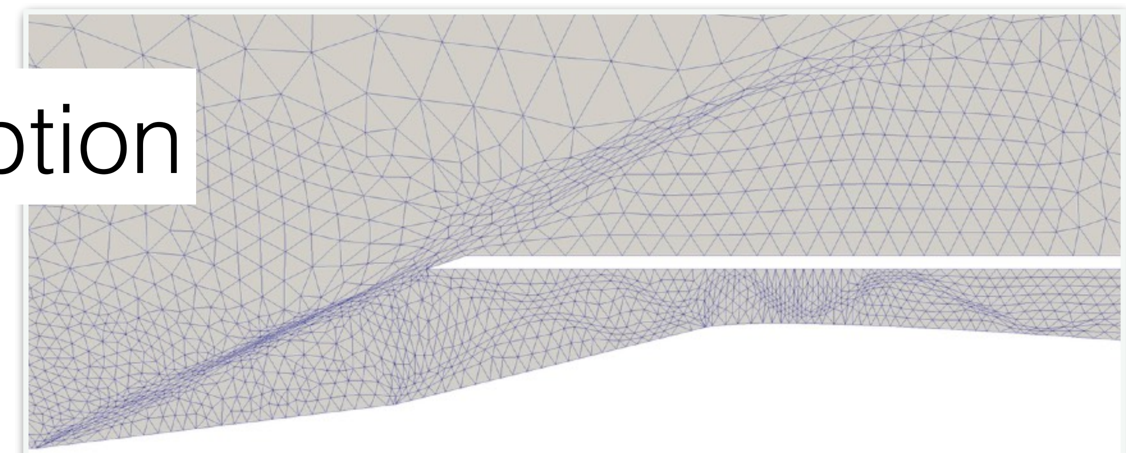
Cost to run 2.5 time units (x_{sh}/u_{inf})

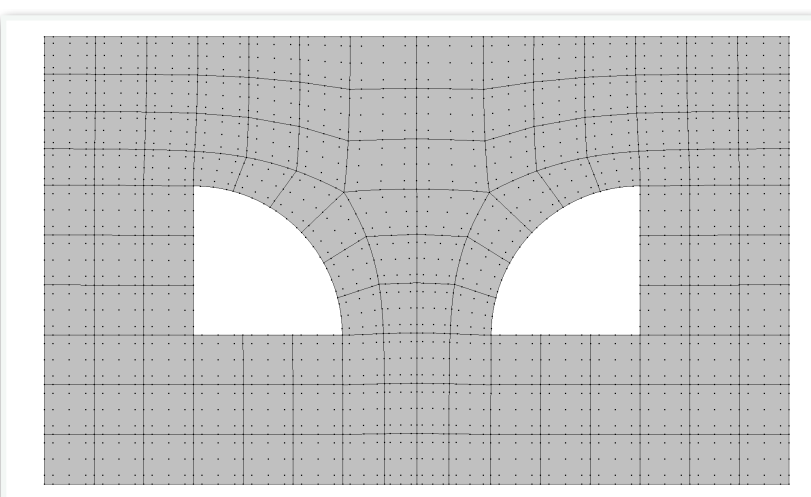
	AV			
	RK2	DIRK2		
$\triangle t$	6.64e-5	1.13e-3	5.56e-3	1.13e-2
CFL	0.05	1	5	10
CPUh	10.7	12.4	4.14	3.19
speed-up		0.86	2.58	3.35

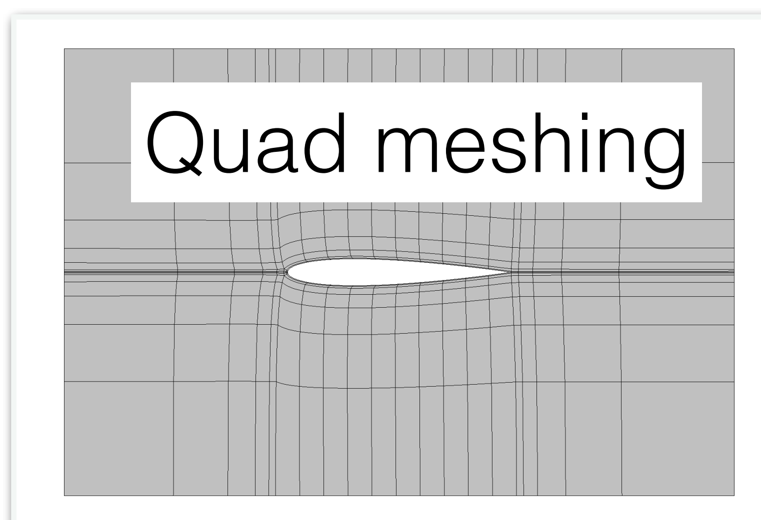
NekMesh

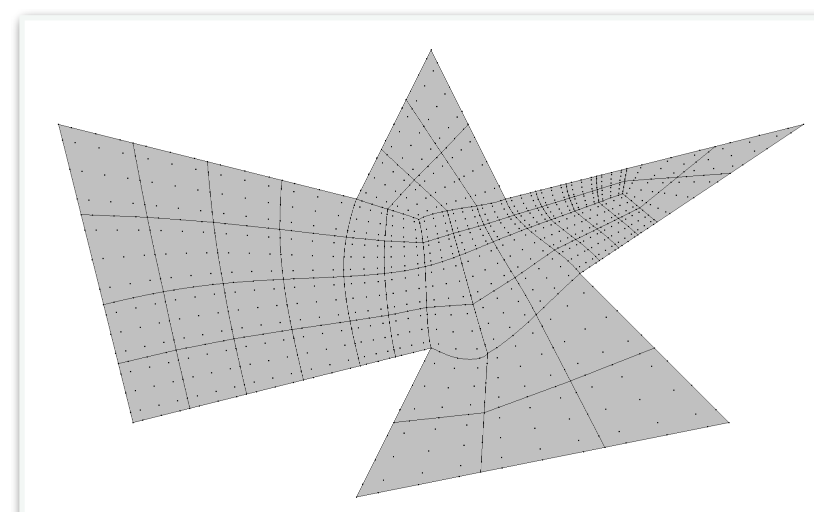
Joaquim Peiró (Imperial): *NekMesh:* An open-source high-order mesh generator







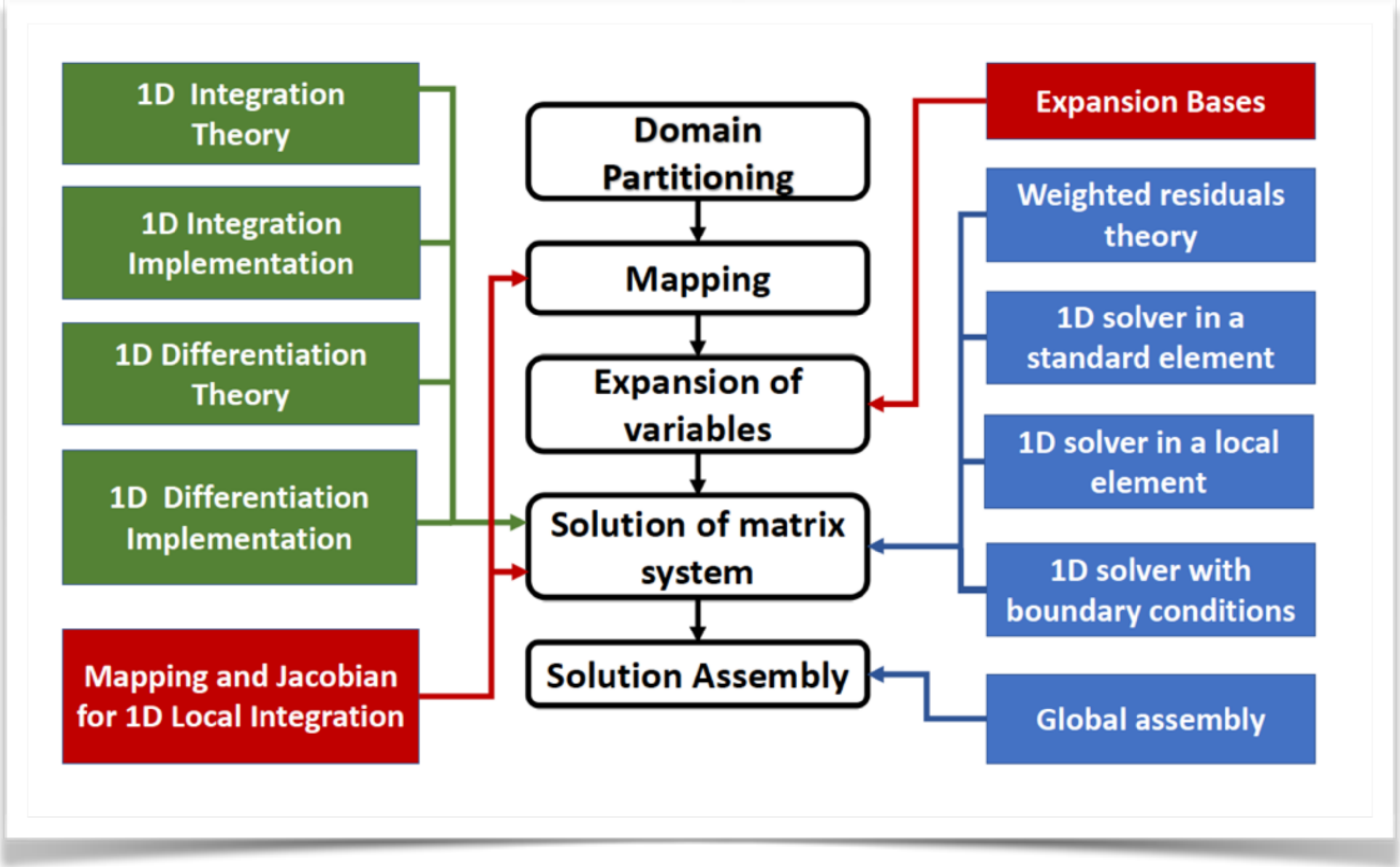




In the pipeline: Outreach

Tuesday afternoon: Jupyter notebooks & the Python interface

Python → Jupyter tutorials



Educational tutorials about fundamentals of a spectral/hp element method

Any comments?