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.

\[ \mathbf{U} \cdot \mathbf{n} = 0 \]
\[ un_x + vn_y = 0 \]
Multiregions restoring

- MultiRegions: ExpList simplified
- constructors reduced from 36 to 6.
- Some changes within SolverUtils
Moxey, Amici, Kirby "Efficient matrix-free high-order finite element evaluation for simplical elements" Under review SIAM J.Sci Comp,
Fault Tolerance/Resilience

The need for Resilience at Exascale

- Barker et al, 2014
- Di Martino et al, 2015
- Chen et al, 2017
- Cappello et al, 2014

Increased probability of hardware failure: more CPUs, memory, disks

$$\text{Mean time to interrupt (MTTI) of a component}$$

Algorithm: Recovery

<table>
<thead>
<tr>
<th>Example</th>
<th>PF</th>
<th>Mean time to interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>27.11</td>
<td>173 hours [1]</td>
</tr>
<tr>
<td>Blue Waters (CPU-only)</td>
<td>5.66</td>
<td>8.6 hours [2]</td>
</tr>
<tr>
<td>Tianhe-2 (8k nodes)</td>
<td>17.30</td>
<td>2 hours [3]</td>
</tr>
<tr>
<td>(Exascale)</td>
<td>1000</td>
<td>&lt; 1 hour ? [4]</td>
</tr>
</tbody>
</table>

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

![Graph showing a 2D mesh with non-conformal interfaces.](image-url)
Zhenguo Yan (Imperial): Development of implicit compressible flow solver in Nektar++

Jacobian-free Newton Krylov method (JFNK)

We need to solve
\[ N\left( u^{n+1,m} \right) = u^{n+1,m} - S_m - \alpha_m F_m = 0 \quad (u^{n+1,m,0} = S_m), \]  

(12)

JFNK method is used for solving the nonlinear system

- Newton method: solving nonlinear system iteratively
  \[ \left( \frac{\partial N}{\partial u} \right) P \triangle u^{n+1,m,l} = -N\left( u^{n+1,m,l} \right) \]  
  
(13)

- Jacobian-free, linearize nonlinear equation in each Newton iteration
  \[ \frac{\partial N}{\partial u} \cdot q = \frac{N(u^{n+1,m,l} + \epsilon_J q) - N(u^{n+1,m,l})}{\epsilon_J}, \quad N(u^{n+1,m,l}) \text{ stored} \]  
  
(14)

- Krylov method to solve linear system: GMRES, restarted every 30 iterations
- Preconditioner, approximate block Jacobi inversion
  \[ \hat{q}^{k+1} = D^{-1} \left( q - (L + U) \hat{q}^k \right), \quad = D^{-1} \text{ stored} \]  
  
(15)

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

<table>
<thead>
<tr>
<th></th>
<th>RK2</th>
<th>AV</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta t )</td>
<td>6.64e-5</td>
<td>1.13e-3</td>
</tr>
<tr>
<td>CFL</td>
<td>0.05</td>
<td>1</td>
</tr>
<tr>
<td>CPUh</td>
<td>10.7</td>
<td>12.4</td>
</tr>
<tr>
<td>speed-up</td>
<td>0.86</td>
<td>2.58</td>
</tr>
</tbody>
</table>
NekMesh

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

- r-adaption
- Quad meshing
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?