Science and Technology Facilities Council

Hartree Centre

High-order FEM implementation in AMReX using PETSc

PETSc User Meeting – 6th June 2023

Karthik Chockalingham, **Alex Grant**, Xiaohu Guo

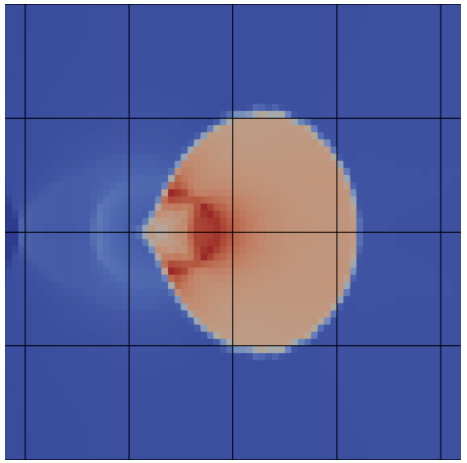
Summary

- Introduction
  - AMReX, motivation, use of AMReX data structures
- Implementing FEM in AMReX (amrPX)
  - Poisson – simplest case, refinement, higher order
  - Initial performance results
  - Time-dependent problems - heat equation, Navier-stokes
- Challenges and next steps

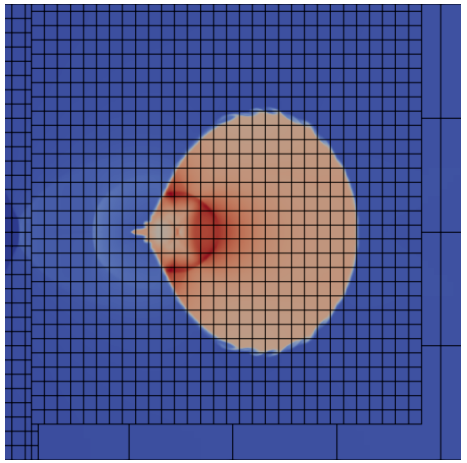
AMReX overview

AMReX is an ECP-funded C++ software framework to support the development of block-structured AMR applications

- Multiple levels at increasing spatial and temporal resolution, each solved independently
- Higher levels sit on top of lower levels
- Each level is comprised of a set of rectangular boxes
- Base level is fixed and covers the whole domain, higher levels change dynamically as the simulation evolves



Level 0

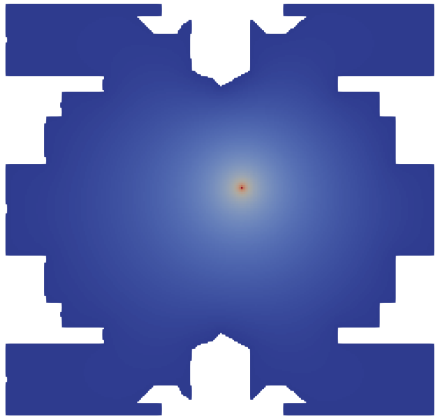


Level 1 (8x refinement)

Motivation for FEM in AMReX

- First-ever C++ implementation of FEM with AMReX
- Performance benefits of structured vs unstructured grids
- Collaboration with UKAEA on fusion research
  - Developing high-order FEM particle-in-cell for simulation of edge plasma physics
  - Developing embedded boundary functionality for complex geometries
  - Current UKAEA production codes have issues with locating particles on an unstructured grid
  - Using a structured grid makes locating particles easy

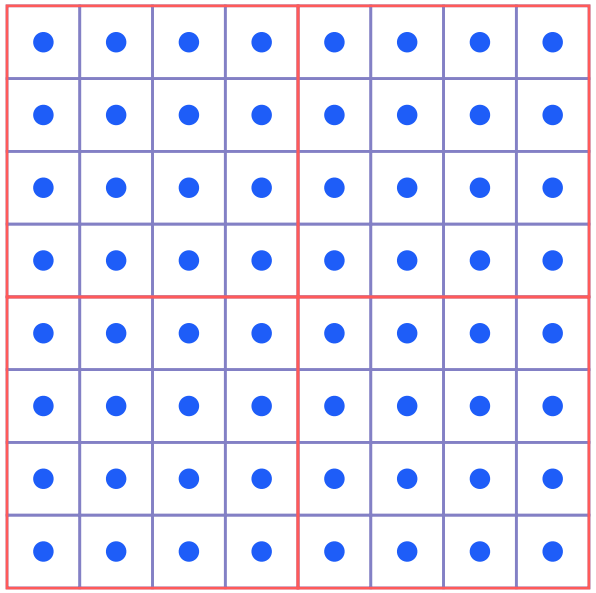
UK Atomic Energy Authority



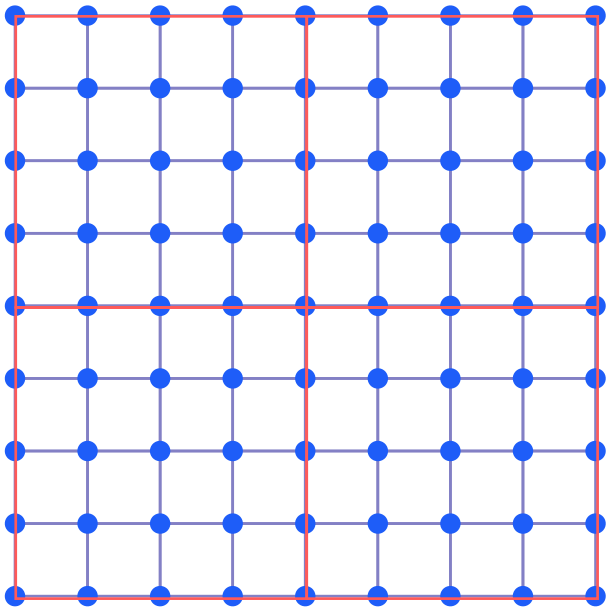
Science and Technology Facilities Council

Hartree Centre

AMReX data structures



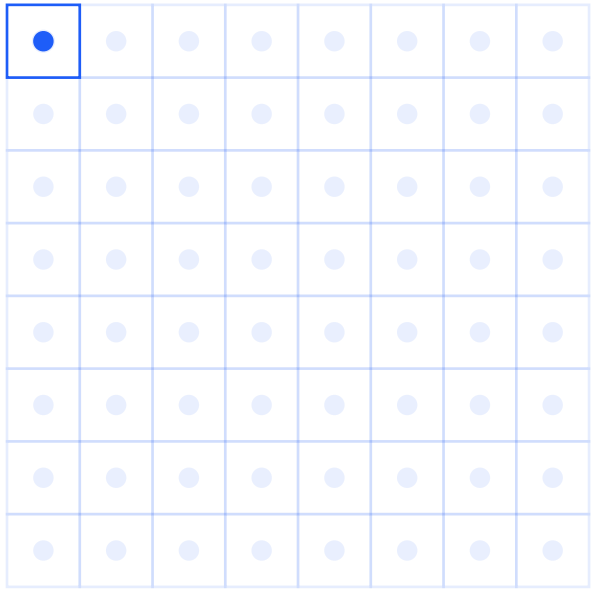
Cell-centered data



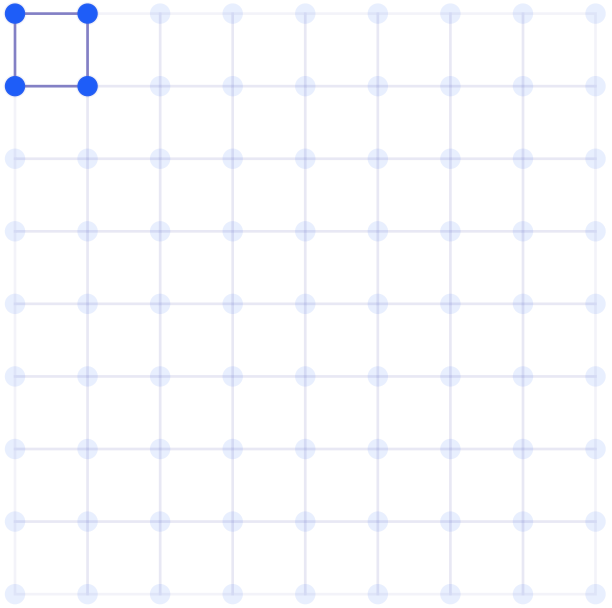
Nodal data

- AMReX has cell-centered or nodal data, split into boxes
- Each cell/node has unique index in each coordinate direction (0 to n-1, 0 to 2n-1), boxes are defined from low cell to high cell
- For FEM, we use cell-centered boxes to iterate over elements, nodal boxes to store data, interpolate between levels, indexing

AMReX data structures



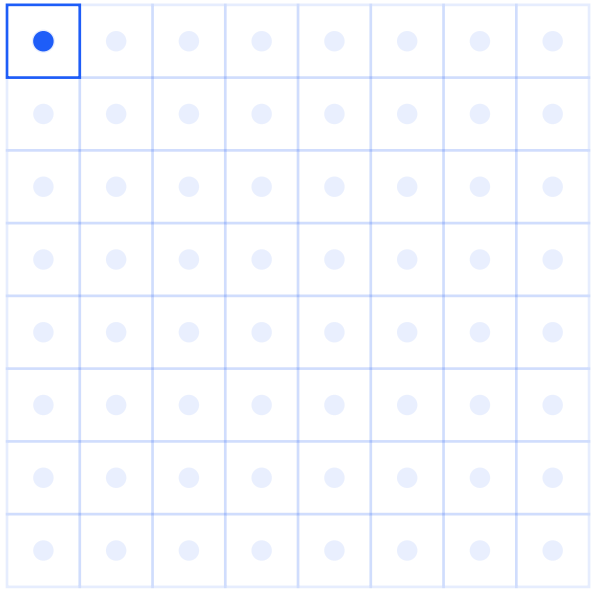
Cell-centered data



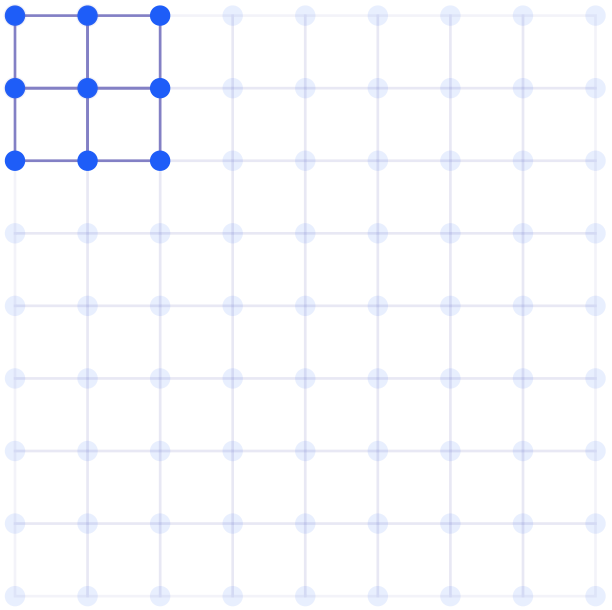
Nodal data

- For higher-order elements, can map cell-centered boxes to refined nodal data (up to the implementation, not AMReX)
- Node locations within the element might not be correct, but implementation doesn't need this, just needs to store the data

AMReX data structures



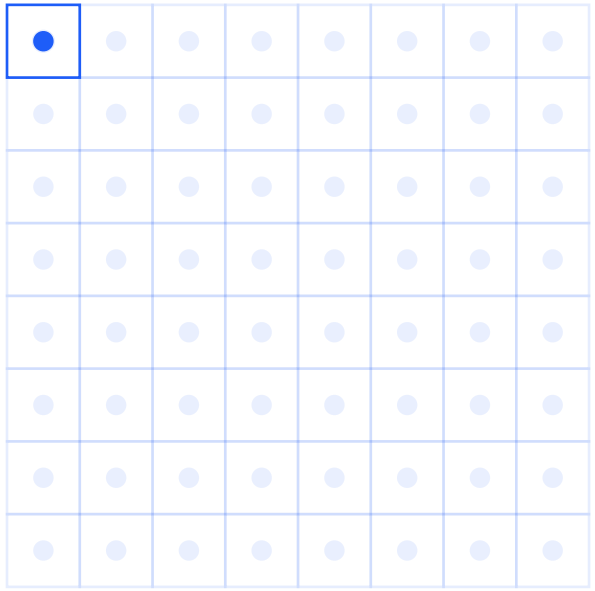
Cell-centered data



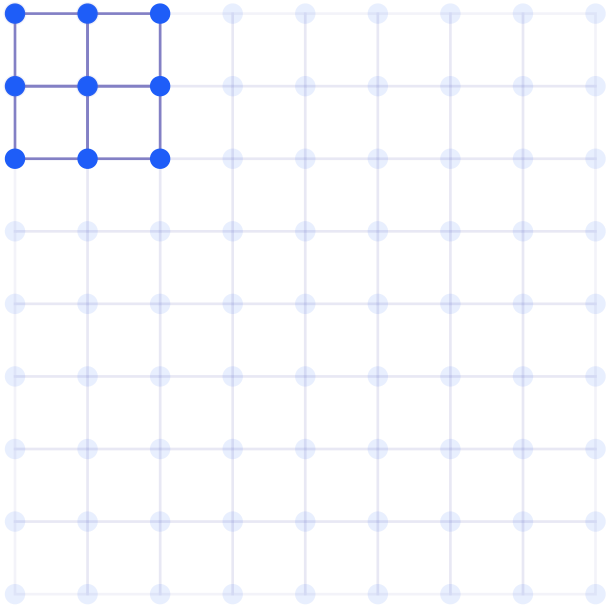
Nodal data

- For higher-order elements, can map cell-centered boxes to refined nodal data (up to the implementation, not AMReX)
- Node locations within the element might not be correct, but implementation doesn't need this, just needs to store the data

AMReX data structures



Cell-centered data



Nodal data

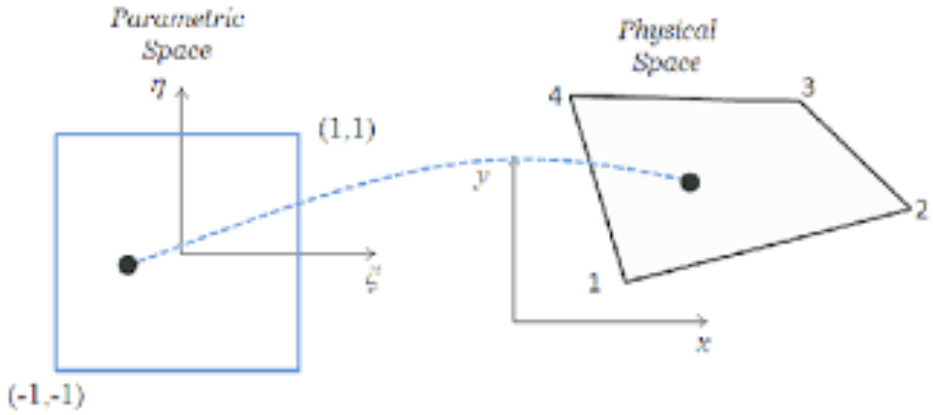
Algorithm 1 Element by element assembly of the stiffness matrix and the load vector

for each element $e \in E$ **do**
  for each degree of freedom $i \in M_e(e)$ **do**
    for each degree of freedom $j \in M_e(e)$ **do**
      $K_{ij} += \int_{\Omega_e} \kappa \nabla \phi_i \cdot \nabla \phi_j \, dV$
    end for
    $\bar{l}_i += \int_{\Omega_e} f \phi_i \, dV$
  end for
end for

Poisson problem

$$\frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial y^2} = -f \qquad \phi(x) = 0 \quad \forall \quad x \in \Gamma_D$$

$$\phi = \psi_I(x, y)\phi^I$$



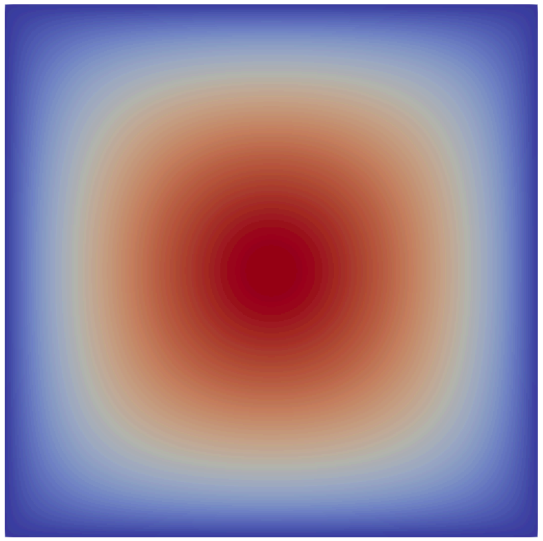
Coordinate transformation (only once per level)

$$k_{mn} = \int_{-1}^{1}\int_{-1}^{1} \left\{ \begin{array}{c} \frac{\partial \psi_m}{\partial \xi} \\ \frac{\partial \psi_m}{\partial \eta} \end{array} \right\} \left[J^{-1}\right]^{\top} [\mathbb{I}] \left[J^{-1}\right] \left\{ \begin{array}{c} \frac{\partial \psi \eta}{\partial \xi} \\ \frac{\partial \psi_n}{\partial \eta} \end{array} \right\} [[J] \mid d\xi d\eta$$

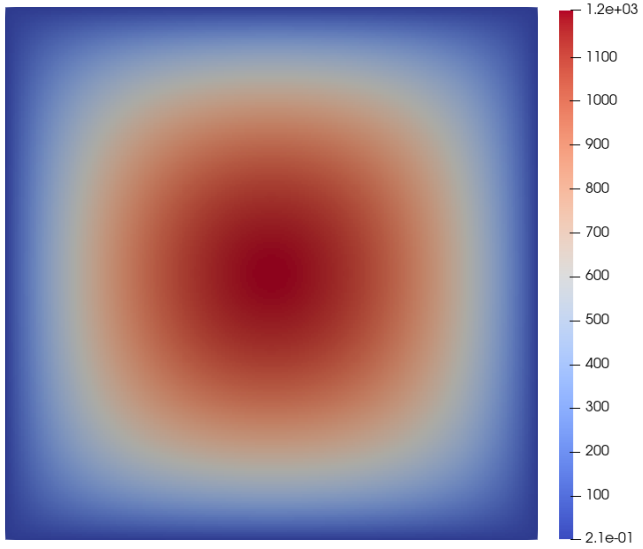
$$f_n = \int_{-1}^{1}\int_{-1}^{1} \{ \ f\psi \ \} [[J] \mid d\xi d\eta$$

$$k_{mn}\phi_n = f_m$$

Simplest case – linear, no refinement



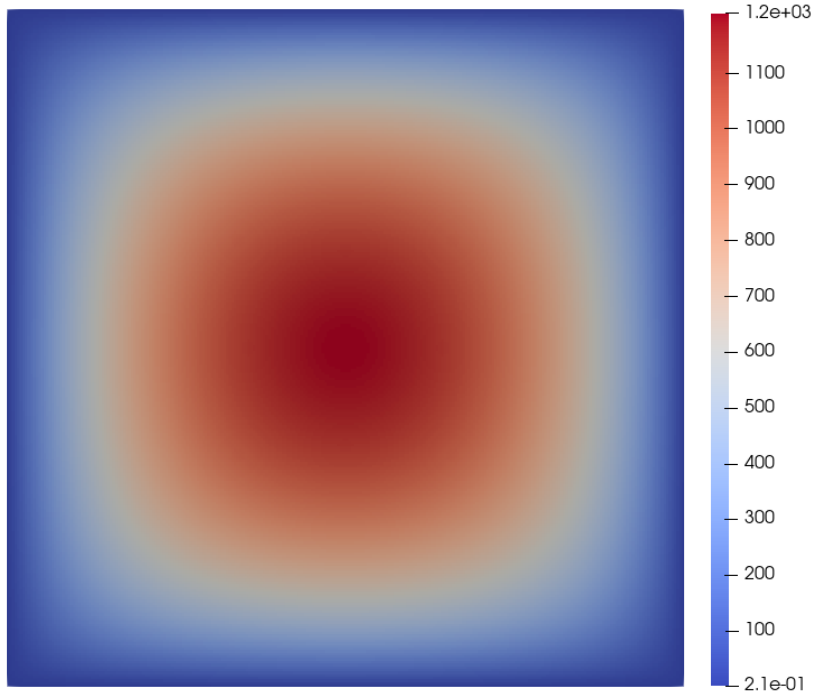
amrPX output



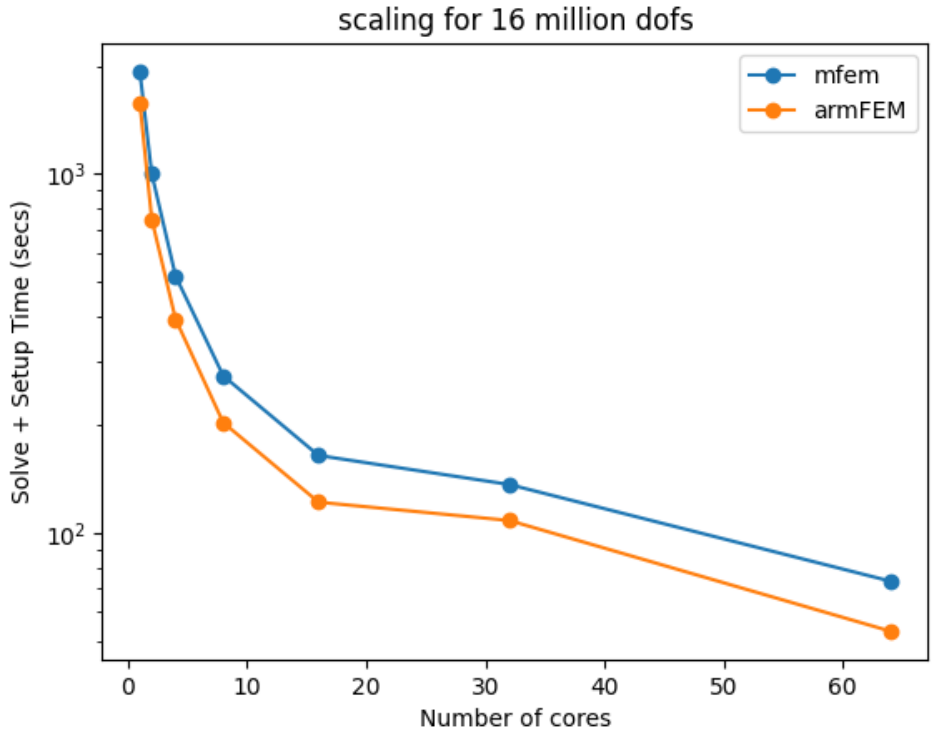
MFEM output

- Comparison with MFEM shows we are getting the correct result
- Covering entire domain means number of nodes is known, single solve
- Higher levels will not cover entire domain, need to be solved independently

Comparing amrFEM and mfem



Verified with mfem



amrFEM is up to 37% faster compared to mfem

Science and Technology Facilities Council

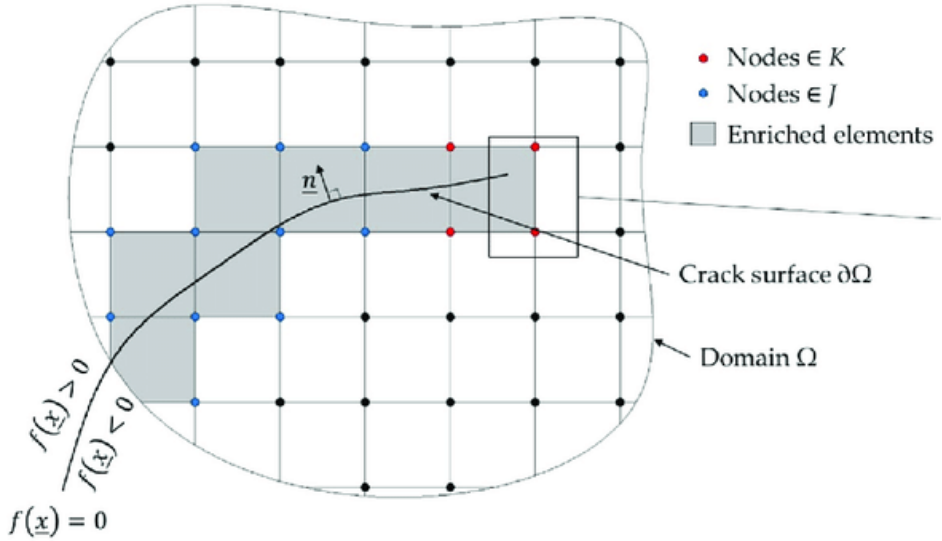
Hartree Centre

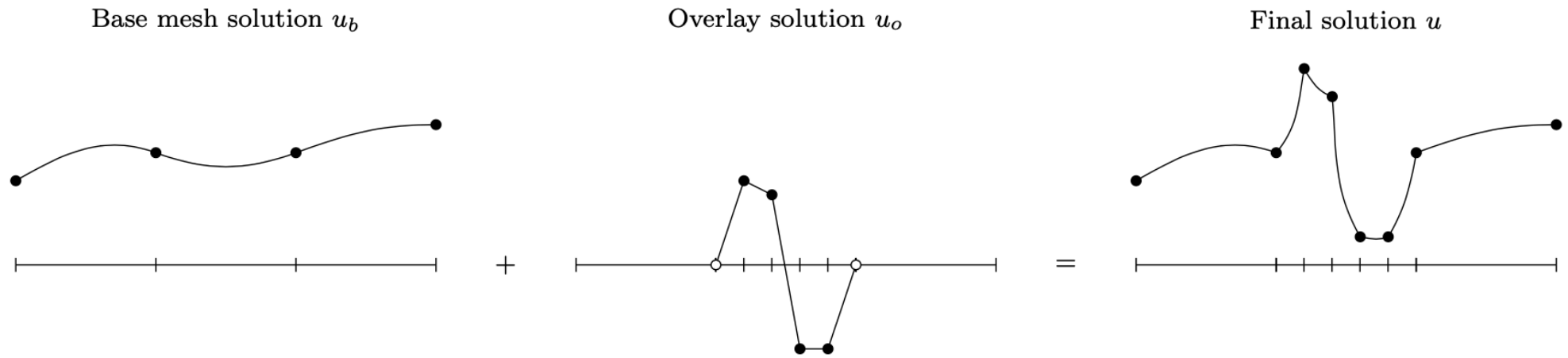
Majority of the computational cost saved in amrFEM is due all elements having the same dimension and straightforward indexing scheme of the elements.

Superposition

Principle of superposition is widely used to model fracture using XFEM

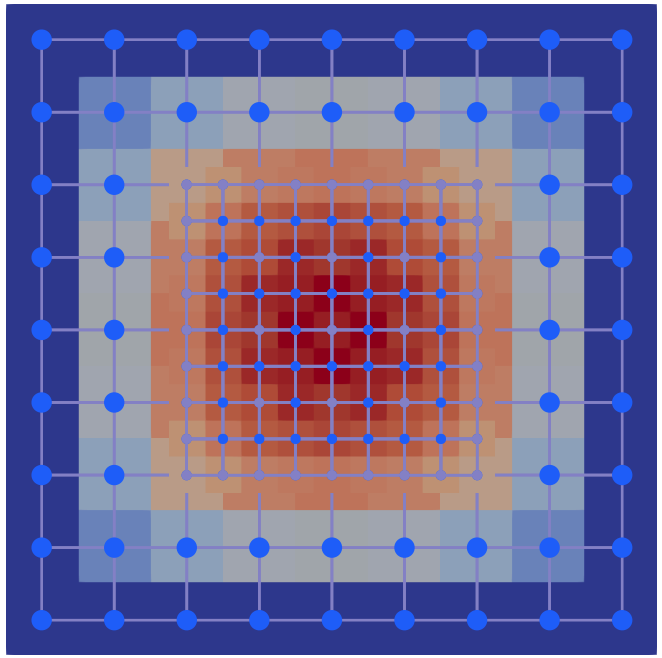
$$u(\boldsymbol{x}) = \sum_i N_i(\boldsymbol{x})\hat{u}_i + \sum_i N_i(\boldsymbol{x})\psi(\boldsymbol{x})\hat{a}_i.$$





De Cicco, Davide. Taheri, Farid, Applied Sciences (2018)
Nils Dietrisch Zander, Ph.D. dissertation, Technischen Universität München (2016)

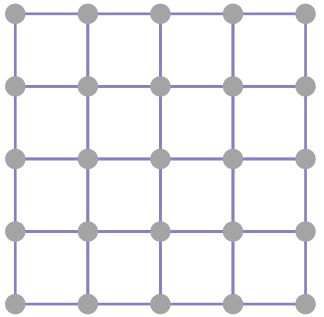
Refined Poisson problem

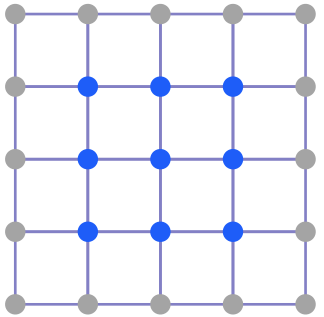


1. Convert cell-centred grid to nodal
2. Solve the problem on the current level
3. Generate the new level (cell -> nodal)
4. Inactivate ‘covering’ nodes and nodes on level boundaries
5. Solve the problem at the refined level
6. Interpolate the solution from the lower level to the higher level
7. Add the interpolated solution to the refined solution

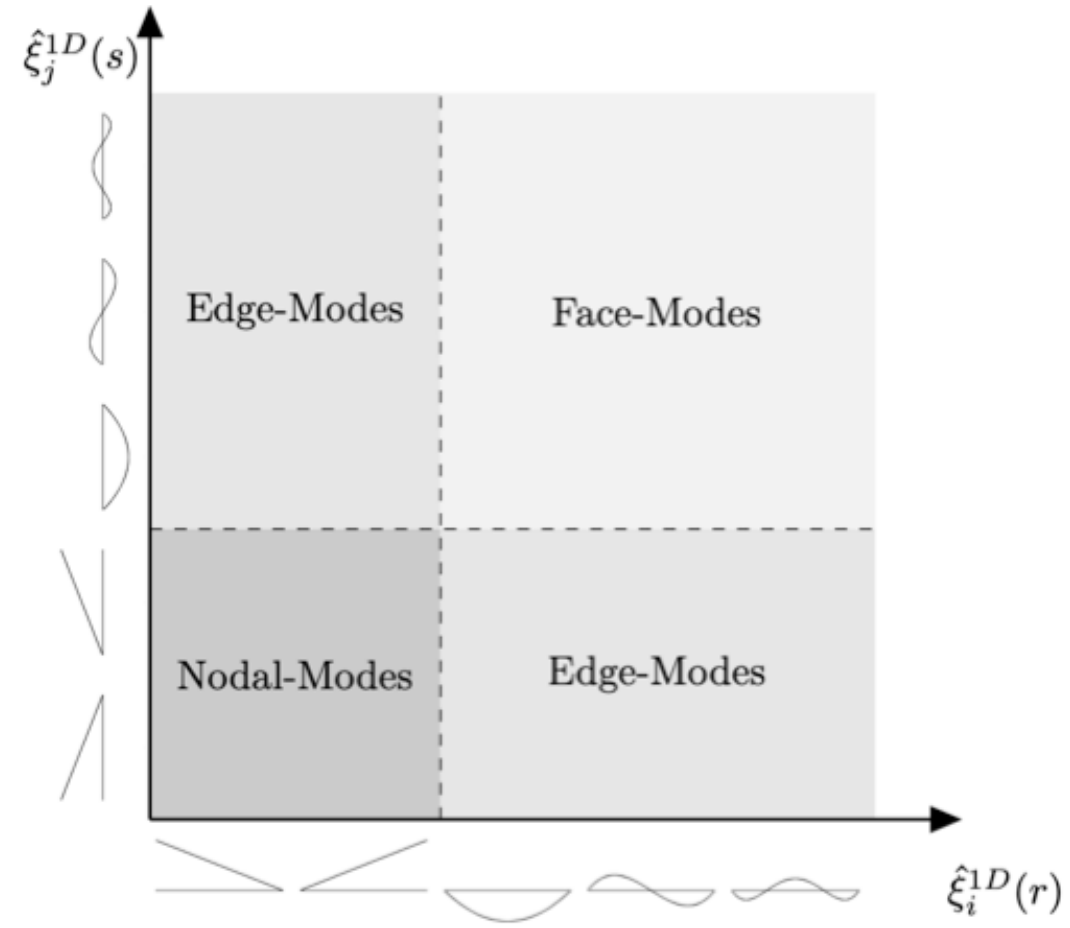
Implementation – Matrix Assembly

- Use AMReX indexing (covering entire domain at all levels) as basis for the global node ID
- Do initially assign some memory for all points, but start by assuming everything is inactive – i.e. single entry of 0
- Go through all active elements on a level and re-allocate memory just for the active elements, then assemble using MATMPIAIJ
- Use ‘redistribute’ preconditioner to remove null elements in the solve





Integrated Legendre polynomial



Legendre polynomials (Spectral elements)

$$L_0(r) = 1$$
$$L_1(r) = r$$
$$L_i(r) = \frac{1}{n}\left[(2i-1)rL_{i-1}(r) - (i-1)L_{i-2}(r)\right] \quad i = 2, 3, \ldots, p,$$

$$\int_{-1}^{1} L_i L_j \, \mathrm{d}r = \begin{cases} \frac{2}{2i+1}, & \text{if } i = j \\ 0, & \text{else} \end{cases}$$

Integrated Legendre polynomials (Hierarchical elements)

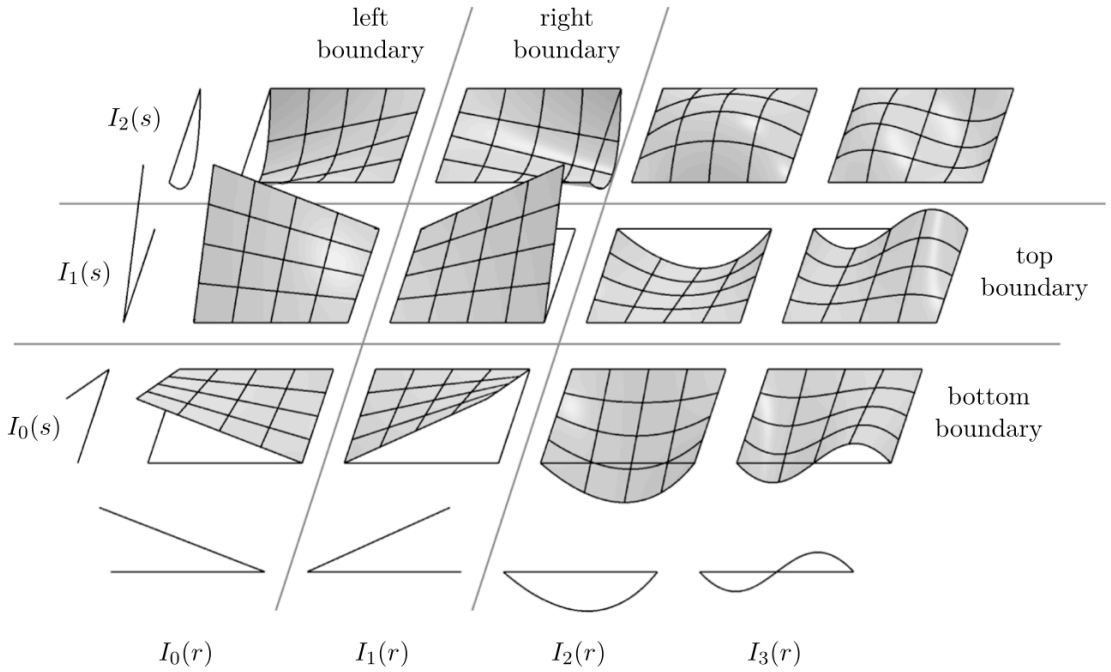
$$P_i(r) = \sqrt{\frac{2i-1}{2}} \int_{-1}^{r} L_{i-1}(t)dt = \frac{1}{\sqrt{4i-2}} \left(L_i(r) - L_{i-2}(r)\right) \quad i = 2, 3, \ldots$$

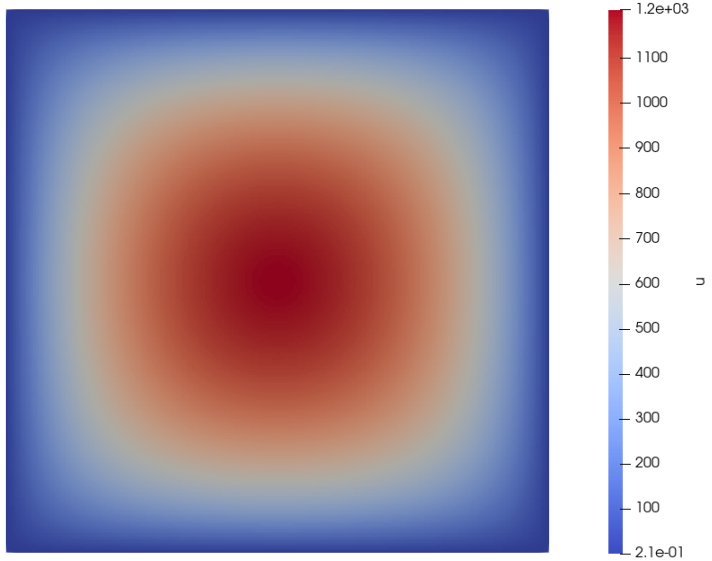
$$\int_{-1}^{1} P_i'(x) P_j'(x) dx = 0, \quad \text{if} \quad i \neq j$$

Image from Nils Dietrisch Zander, Ph.D. dissertation, Technischen Universität München (2016)

Higher order implementation (on a uniform mesh) using Integrated Legendre polynomials

$$u(\boldsymbol{x}) = \sum_i N_i(\boldsymbol{x})\hat{u}_i + \sum_i N_i(\boldsymbol{x})\psi(\boldsymbol{x})\hat{a}_i.$$





Output of second order integrated Legendre polynomial implementation
on a 256 x 256 mesh

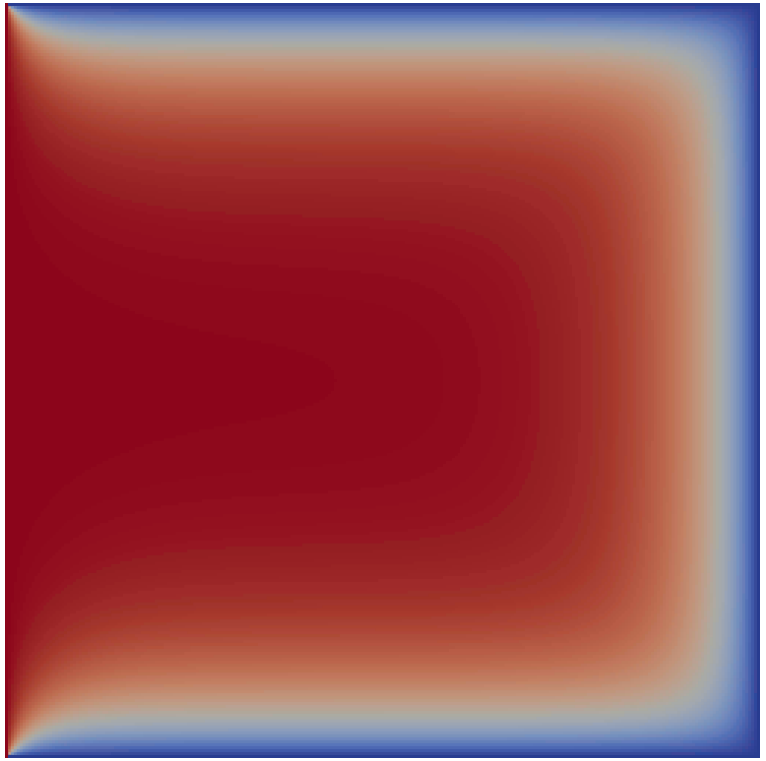
The number of nodes stays constant as the polynomial order increases, only additional edge and face modes

Time-dependent problems

- Heat equation fully implemented

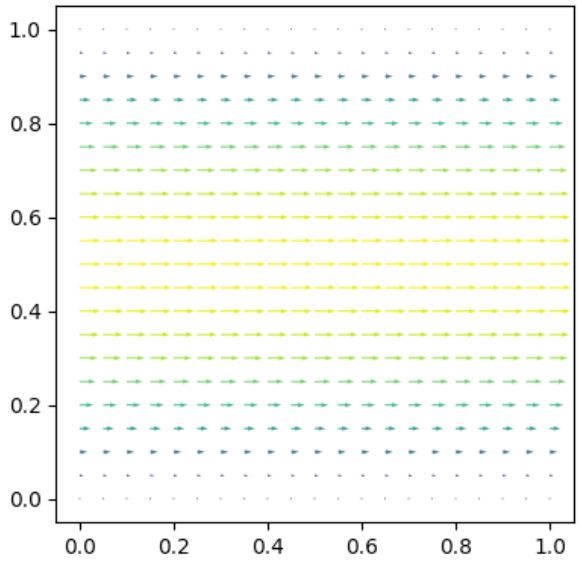
$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

- Time discretization via Backward Euler
- First order only, no refinement yet.
- Solution matches with existing established unstructured FEM package (MFEM)



Time-dependent problems

- Navier-Stokes implementation underway
  - Channel flow test case
  - Chorin's scheme
  - Matching step by step with MOOSE



– 1: Compute the tentative velocity $u_h^\star$ by solving

$$\langle v, D_t^n u_h^\star \rangle + \left\langle v, \nabla u_h^{n-1} \cdot u_h^{n-1} \right\rangle + \langle \nu \nabla v, \nabla u_h^\star \rangle = \langle v, f^n \rangle \quad \forall v \in V_h,$$

including any boundary conditions for the velocity.

– 2. Compute the corrected pressure p_h^n by solving

$$\langle \nabla q, \nabla p_h^n \rangle = -\langle q, \nabla \cdot u_h^\star \rangle / t_n \quad \forall q \in Q_h,$$

including any boundary conditions for the pressure.

– 3. Compute the corrected velocity u_h^n by solving

$$\langle v, u_h^n \rangle = \langle v, u_h^\star \rangle - t_n \langle v, \nabla p_h^n \rangle \quad \forall v \in V_h$$

Challenges

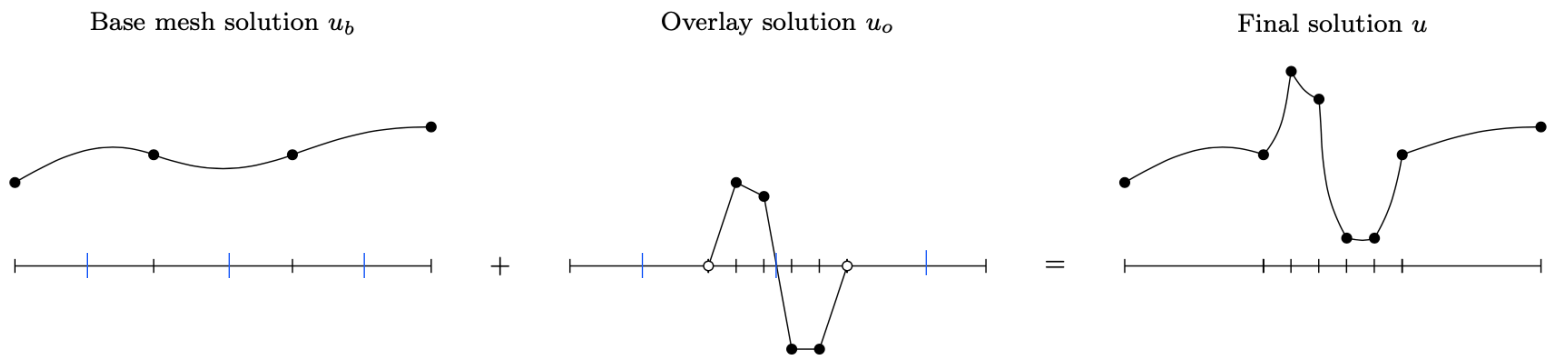
Need to get PETSc to match AMReX's data partitioning

- PETSc results needed in AMReX for file output, assembly at next timestep, AMReX mesh refinement functions (e.g. interpolation)
- Current workaround is to scatter results to all processes – not ideal
- IS index scheme, DM methods, other options?
- Halo/shared nodes – how to get non-local data (VecCreateGhost)?
- PETSc local assembly without using stashing

Challenges

Higher order AND multi-level refinement

- Higher order nodes/edges would conflict with higher level nodes
- Once area for refinement is determined, need to re-do existing level with refined areas linearized
- Reduces performance advantage of structured grids



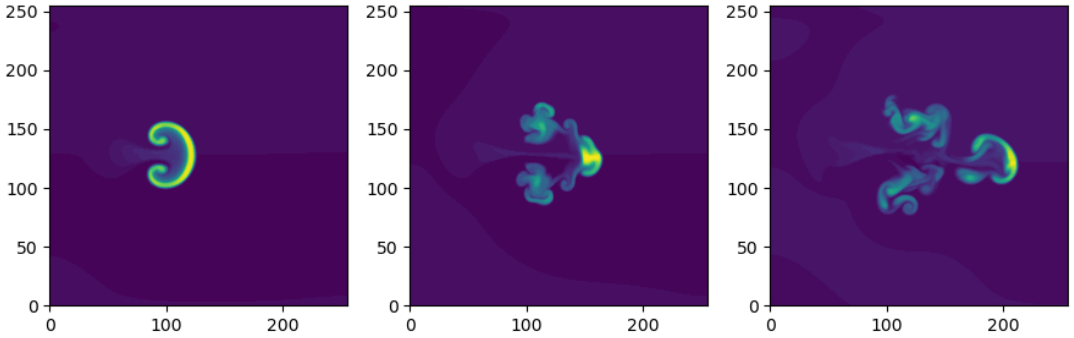
Challenges

GPU development

- Want to be easy-to-use and portable
- Using AMReX, PETSc, Hypre and Eigen – all need to be built together with support for GPU
- Performance portability – memory management

Next steps

- Code redevelopment – make more general and flexible. Currently a lot is hard-coded
- Implementation of ‘Blob2D’ fusion test case (https://hermes3.readthedocs.io/en/latest/examples.html#d-drift-plane)



Solve for the electron density, pressure, vorticity, connection length

$$\frac{\partial n_e}{\partial t} = -\nabla\cdot(n_e\mathbf{v}_{E\times B}) + \nabla\cdot\frac{1}{e}\mathbf{j}_{sh}$$

$$p_e = e n_e T_e$$

$$\frac{\partial \omega}{\partial t} = -\nabla\cdot(\omega\mathbf{v}_{E\times B}) + \nabla\left(p_e\nabla\times\frac{\mathbf{b}}{B}\right) + \nabla\cdot\mathbf{j}_{sh}$$

$$\nabla\cdot\left(\frac{1}{B^2}\nabla_\perp\phi\right) = \omega$$

$$\nabla\cdot\mathbf{j}_{sh} = \frac{n_e\phi}{L_{||}}$$

Summary

- Proof of concept for the first working C++ FEM implementation in AMReX
- Working towards a real use-case (fusion applications)
- Dynamic multi-level hp-refinement is a capability uniquely suited to AMReX
- Will be open-source (but needs code restructuring first)

Science and Technology Facilities Council

Hartree Centre

Thank you

hartree.stfc.ac.uk @HartreeCentre STFC Hartree Centre hartree@stfc.ac.uk