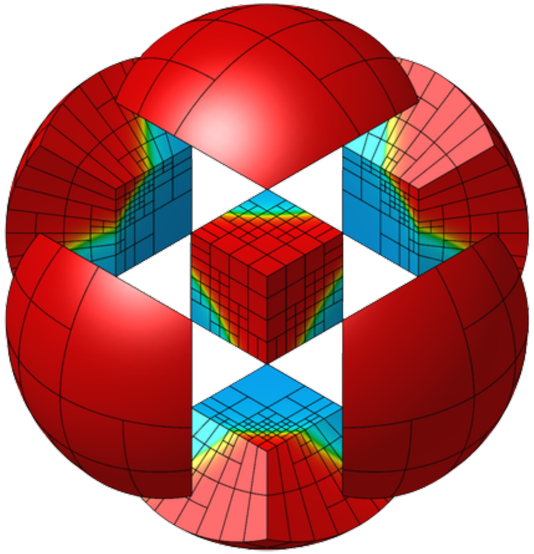
MFEM: Recent Developments



MFEM Workshop 2021

October 20, 2021, Virtual Meeting

Veselin Dobrev and the MFEM team

LLNL-PRES-828998

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

CASC

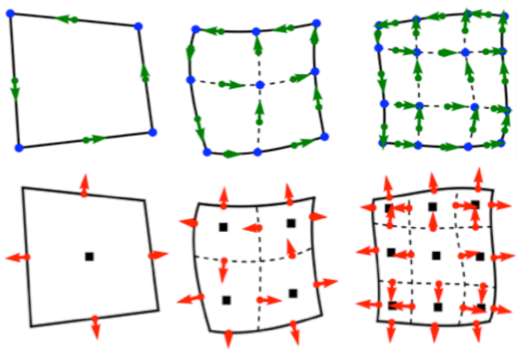
Lawrence Livermore National Laboratory

MFEM: Modular Finite Element Methods library

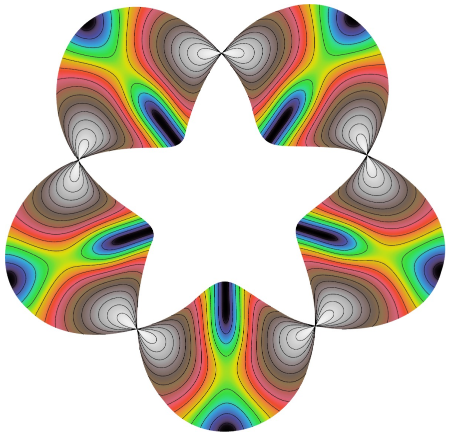
MFEM is open-source C++ library for scalable FE research and fast application prototyping

mfem.org

- Triangular, quadrilateral, tetrahedral, prismatic, and hexahedral; volume, surface and topologically periodic meshes
- Arbitrary order curvilinear mesh elements
- Arbitrary order H^1, H(curl), H(div) and L^2 elements
- Local conforming and non-conforming refinement
- NURBS geometries and discretizations
- Bilinear and linear forms for variety of methods (Galerkin, DG, DPG, IGA, …)
- Sparse matrices, smoothers, Krylov solvers, eigensolvers
- Scalable assembly and linear solvers through hypre
- Non-linear operators and non-linear solvers
- Explicit and implicit high-order time integration
- Example codes & Miniapps – simple proxies for high-order discretizations of various physics
- Integration with hypre, PETSc, SUNDIALS, STRUMPACK, and more

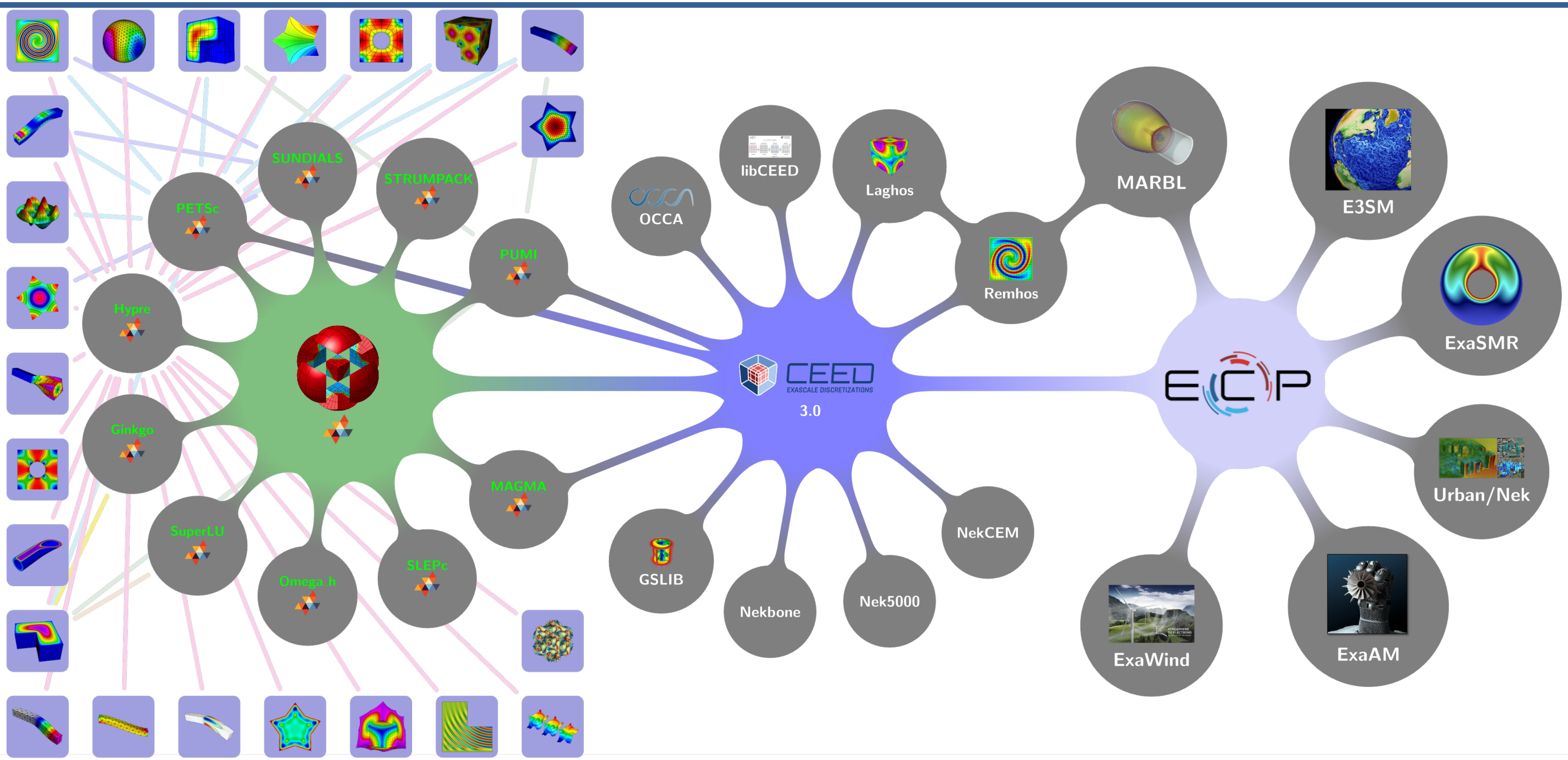


Linear, quadratic and cubic finite element spaces on curved meshes



Maxwell eigenmode on a Mobius strip, computed with LOBPCG+AMS

MFEM connections and integrations



MFEM connections to other projects

- Part of the Extreme-scale Scientific Software Development Kit, xSDK: xsdk.info
- Part of the FASTMath institute: fastmath-scidac.llnl.gov
- Engaged in SciDAC, e.g. RF-SciDAC and TDS-SciDAC

New developments in MFEM

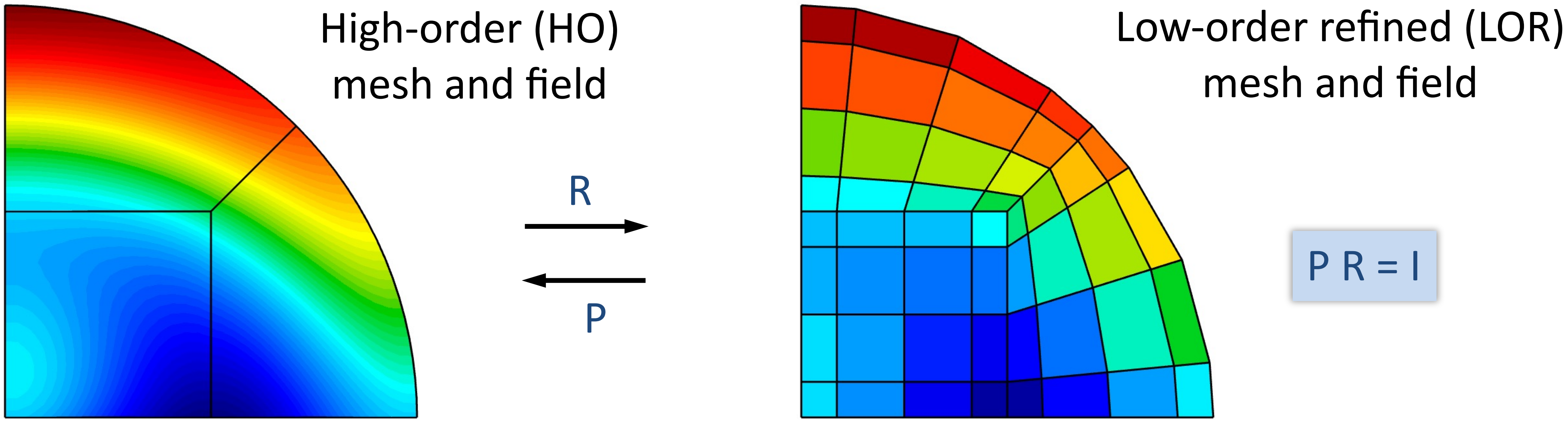
- Support for HYPRE preconditioners on GPUs
- Support for GPU-enabled PETSc
- Mesh optimization (TMOP) on GPU
- Memory manager improvements
- A64FX support (Fugaku CPUs)
- High-order ↔ Low-order-refined (HO/LOR) solution transfer
- Initial (serial) support for p- and hp-refinement
- Improved libCEED integration

Highlight: support for HYPRE on GPUs

MFEM Build option	MFEM Run option	HYPRE w/o CUDA	HYPRE with CUDA
`MFEM_USE_CUDA=NO`	`--device cpu`	Yes	No
`MFEM_USE_CUDA=YES`	`--device cpu`	Yes	Yes
`MFEM_USE_CUDA=YES`	`--device cuda`	Yes	Yes

- Almost all examples, miniapps, and tests work (a few exceptions)
- Data will be moved between host and device as needed; e.g., due to:
  - Algorithm not available on GPU yet
  - Build and/or run configuration
- Ideally: develop on CPU, then use on GPU
- Note: CPU and GPU use different solver options; e.g., iterations will differ
- Optimize: use visual profiler to uncover unnecessary host-device transfers

Highlight: HO ↔ LOR solution transfer



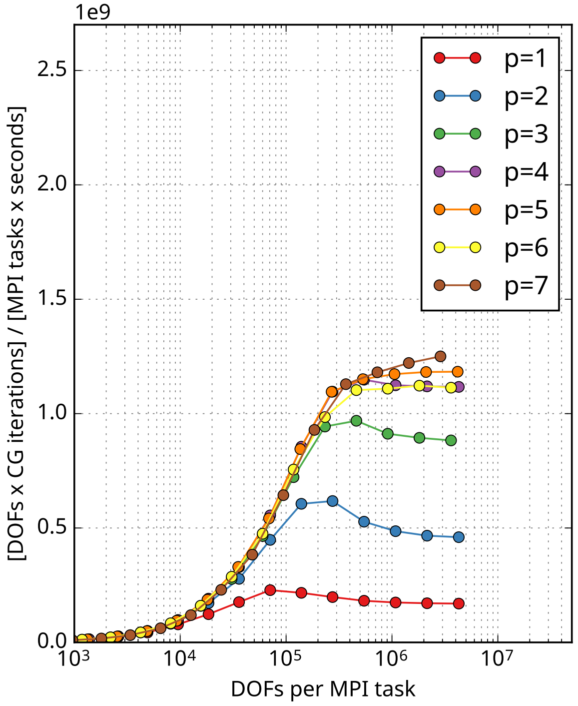
- Useful for connecting HO codes with (existing) LO codes
- R: interpolation or L^2 projection (conservative, conditionally)
- Demo: `lor-transfer` in `miniapps/tools`

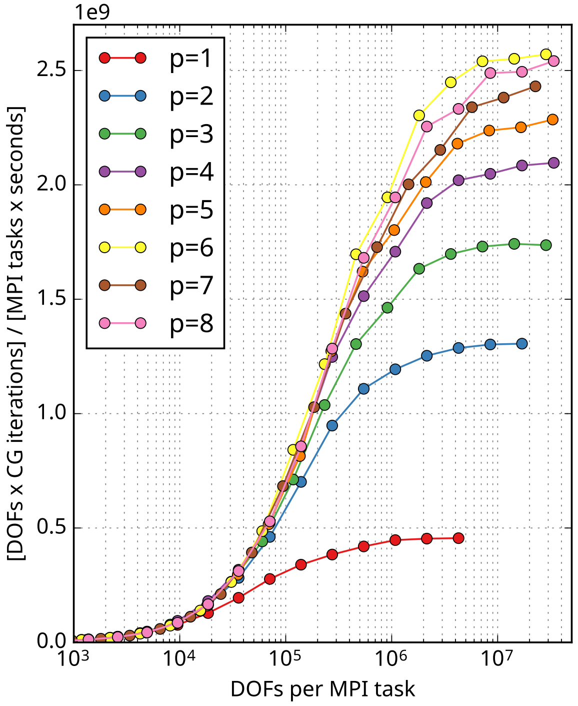
Highlight: libCEED integration

- libCEED: lightweight, portable, and performant operator evaluation github.com/CEED/libCEED
- Matrix-free algorithms with backends targeting CPUs and GPUs
- Integrated in MFEM for *diffusion*, *mass*, *advection*, for now

Backend:
cuda

Improvements
based on libCEED
coming soon



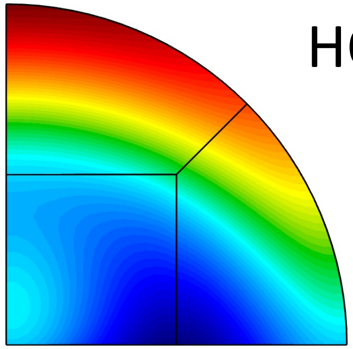


Backend:
ceed-cuda

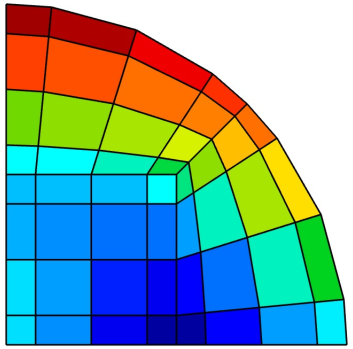
New developments in MFEM (cont.)

- Simplified construction for LOR solvers including H(div) and H(curl) spaces
- Added high-order matrix-free auxiliary Maxwell solver for H(curl) problems
- Support for NVIDIA's AmgX library
- Improved integration with the Ginkgo library
- Integration with the CEED-developed FMS library for high-order field and mesh I/O
- Added support for performance profiling with Caliper
- Improved testing: GitLab CI on LLNL's Quartz, Lassen, Corona machines

Highlight: LOR solvers (a.k.a. FEM-SEM preconditioner)



HO operator



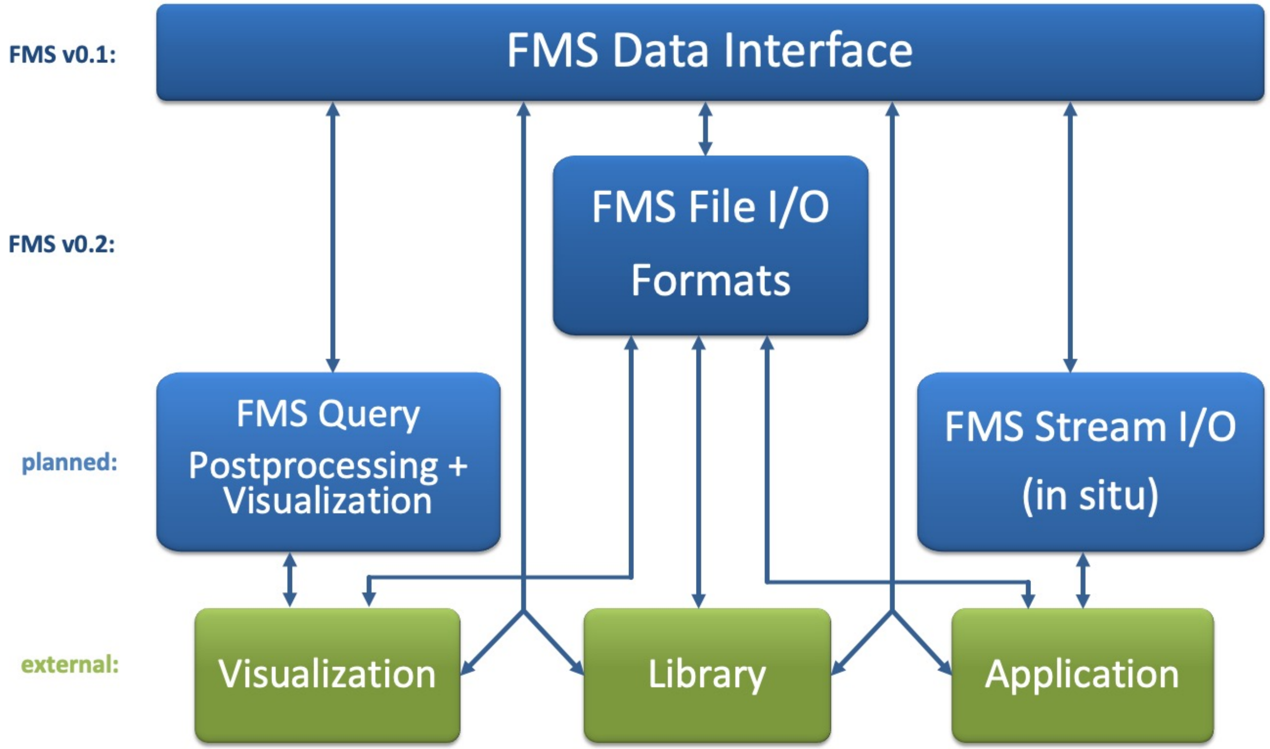
LOR operator

- Expensive to assemble
- Use matrix-free action
- Faster, memory efficient
- Higher accuracy
- Suitable for both CPU and GPU

- Same number of DOFs as HO operator
- Assemble LOR operator (e.g., ParCSR)
- Much sparser operator
- Build preconditioner for this operator
- Use as preconditioner for the HO operator

Highlight: FMS

- Released FMS v0.2
  - **Field and Mesh Specification** for high-order FE data, all FE space types
  - For data exchange, file I/O
  - Supported in MFEM
- Version 0.2 added file I/O support:
  - YAML-like ASCII format
  - Optional Conduit library support: json, yaml, and hdf5 binary formats
  - Supported in VisIt
- github.com/CEED/FMS

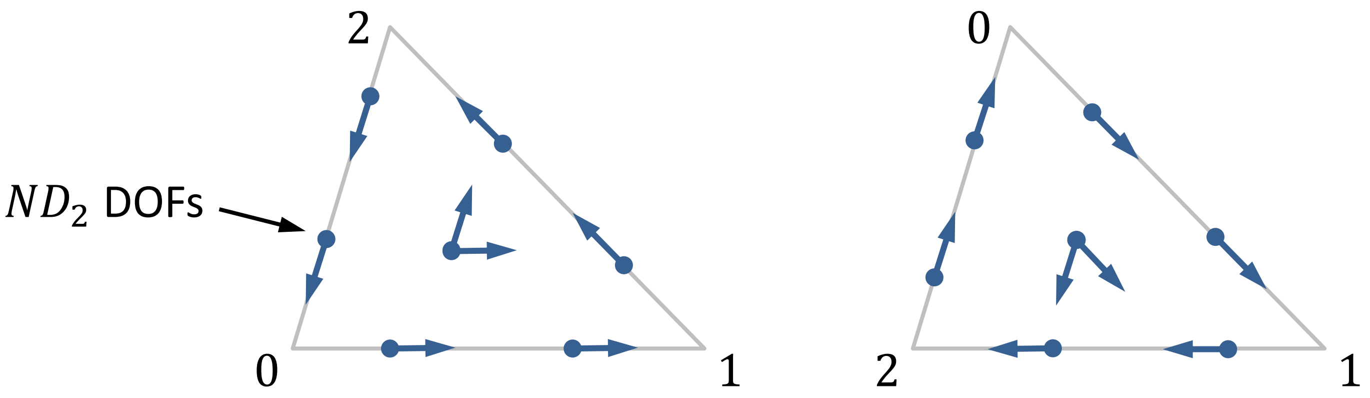


New developments in MFEM (cont.)

- Higher order (p >= 2) H(curl) elements on tetrahedral meshes with refinement support
- Support for pyramid elements, lowest order FE spaces only (for now)
- C++ MFEM Jupyter Notebooks with inline GLVis visualization
- Initial support for google-benchmarks in `tests/benchmarks`
- Support for hr-adaptivity using TMOP-based error estimator
- Many more …

Highlight: Higher-order H(curl) on tetrahedra

- Added support for *arbitrary global-to-element* DOF transformations
- For most FE types the DOF transformation is *permutation + optional sign flip*
- Allows for any tetrahedron orientation with HO (p >= 2) H(curl) FEs
- Also needed for HO (p >= 2) H(curl) FEs on prisms and pyramids



Face of back element

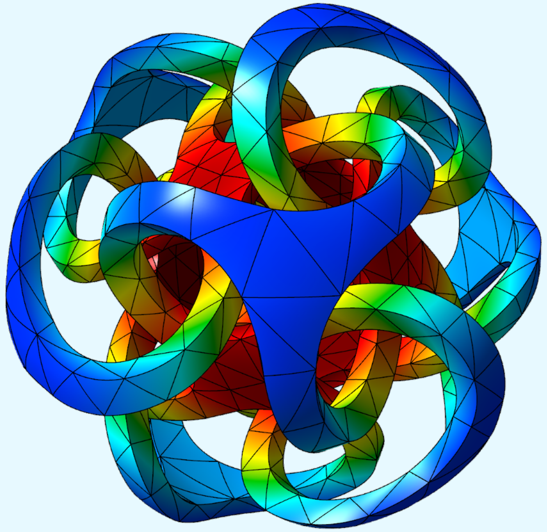
Face of front element

Mapping of interior DOFs requires a linear combination

Visualization

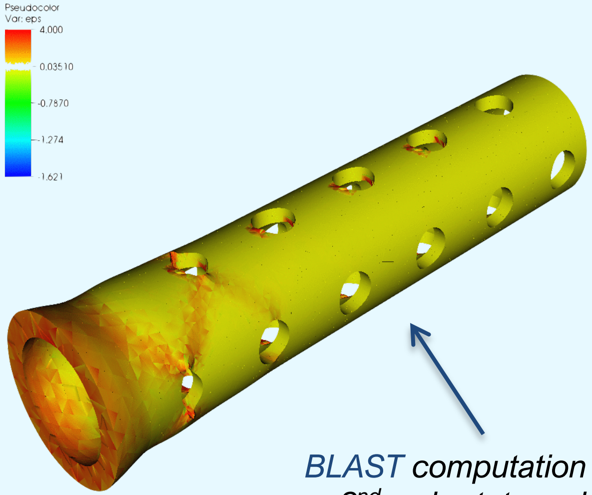
MFEM supports several options for accurate + flexible finite element visualization

GLVis
native lightweight visualization



glvis.org

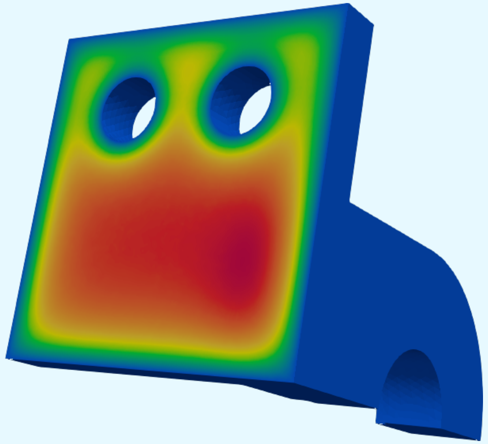
VisIt
general data analysis tool



BLAST computation
on 2nd order tet mesh

visit.llnl.gov

ParaView
general data analysis tool



paraview.org

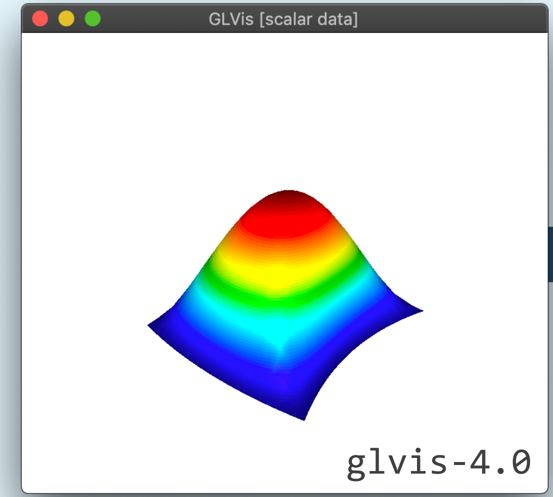
Additional I/O support: Conduit, ADIOS, VTK, FMS, …

Visualization

Web + Python support

Modern OpenGL

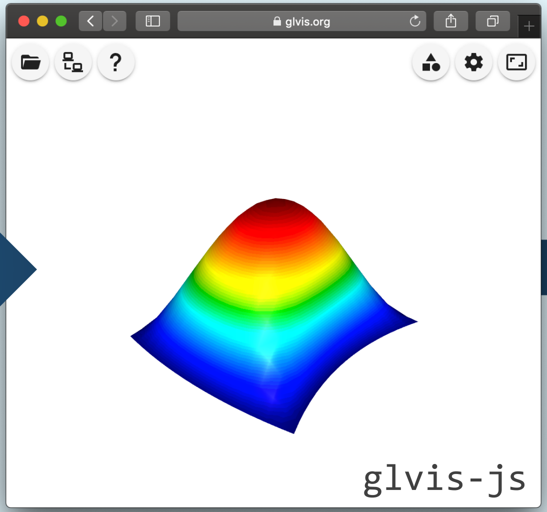
OpenGL3, SDL



glvis.org

Web Visualization

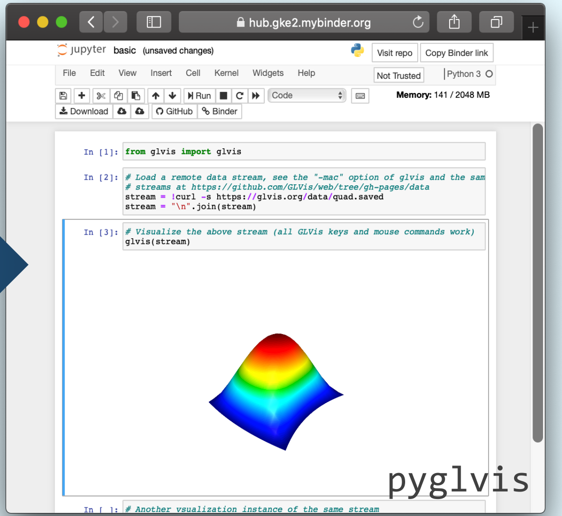
JS/WebAssembly, Emscripten



glvis.org/live

FEM in Python

PyMFEM, Jupyter, Binder



`pip install glvis`

Try `glvis-js` and `pyglvis` in your desktop or mobile browser

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory