Core Features

- Support for mixed finite elements on hybrid unstructured grids in one through three dimensions.
- Assorted finite element classes including $DG/PUH, C^0, C^1$ bases
- Adaptive refinement with hanging nodes for arbitrary elements, $p$ refinement for hierarchical bases
- Parallel system assembly and solution
- Parallel spatial adaptivity, projection, interpolation for general element types
- Nonlinear FEM framework with adaptive time integration
- Parallel distributed mesh data structures and algorithms
- Mesh and solution export and import using common data and visualization formats
- Integration with third party software packages such as:
  - PETSc, Trilinos, or LASPack sparse linear algebra
  - METIS, ParMETIS mesh partitioning
  - Triangle, Tela mesh generation
- Inline API documentation with Doxygen

Object Oriented Programming

- Applications are written to use Abstract Base Classes, e.g.:
  - Quadrature rule $QBase$
  - Geometric element $Elem$
- Finite element class $FEBase$
- Then code is dispatched at runtime to Concrete Derived Classes, e.g.:
  - Gaussian quadrature $QSimpon$
  - Hierarchic polynomials $FE<HIERARCHIC>$$^{\text{CloughTocher}}$
- Optional higher level $FEMSystem$, $TimeSolver$ class hierarchies encapsulate physics, time integration.

Verification

Regression Testing: Automatic checking of example applications, unit tests via BuildBot
- Assertions:
  - GNU ldd++ options verify container validly, index bounds-checking
  - ~ 8000 high-level assertions added by developers
- $\text{libmesh}++;$ function预言ions $\rightarrow$ satisfied
- $\text{libmesh}++;$ function预言ions $\rightarrow$ satisfied

Manufactured Solutions:

Convergence Rates:

\[
\begin{align*}
\frac{R_n}{R_{n+1}} & \overset{\text{O}(p^{-1})}{\rightarrow} \\
\frac{\eta_n}{\eta_{n+1}} & \overset{\text{O}(p^{-1})}{\rightarrow} \\
\frac{\tau_n}{\tau_{n+1}} & \overset{\text{O}(p^{-1})}{\rightarrow}
\end{align*}
\]

Testing convergence of physics code with residual $R_n$ to arbitrary analytic solutions $\phi_{n+1}$ via added forcing function $\eta_n$, $\tau_n$.

Multiphysics

\textbf{IbMesh}: applications results have been published for compressible and incompressible flow and transport, phase change, anisog-eous, radiation transport, many other mathematical models.

Development

Open Source: $\text{libMesh}$ is freely available from $\text{http://libmesh.sf.net}$, and is redistributable under the GNU Lesser General Public License

Research: $\text{libMesh}$ itself is the subject of one peer-reviewed paper [KPSC06], and has been used in the production of data for dozens of applications papers published by $\text{libMesh}$ developers and users.

Adaptivity

Error Estimation: a posteriori error estimators based on patch recovery, solution refinement, or jump residuals are available and applicable to a wide range of problems.

Adaptive Mesh Smoothing: $\text{libMesh}$ integrates variational mesh optimization code, which can be applied using error estimation results to produce solution-adapted mesh redistribution.

Adaptive Mesh Refinement/Coarsening (AMRC): natively supported, with non-conforming hierarchic meshes generated and hanging node continuity constraints automatically applied.

Adaptive $p$ refinement with hierarchical bases.

Goal-Oriented Adaptivity: global error norms are less important to PECOS than errors in local scalar quantity of interest (QoI) functionals. PECOS and collaborators are developing and verifying adjoint-based adaptivity capabilities, designed to specifically reduce error in a specified QoI.

Parallelism

Parallel Execution: Each distributed-memory Message Passing Interface (MPI) process performs calculations on its own element set. On hybrid parallel architectures, elements can be further subpartitioned for individual shared-memory threads using Threading Building Blocks (TBB).

Domain (Re-)Partitioning: Meshes are automatically partitioned for the current number of processors, reconstructed when necessary for partitioning-independent I/O, and repartitioned for load balancing after AMRC.

Data Synchronization: Ghost cells are automatically located. Neighboring processors communicate solution coefficients, constraint equations, etc. as required.

Adjoint-based Sensitivities

\begin{align*}
\text{Application-Perturbation: Automatic adjoint calculations can be run on any} \\
\text{LinearImplicitSystem, any } FEDSystem, \text{ or any} \\
\text{NonlinearImplicitSystem. Application code} \\
\text{must assemble:}
\end{align*}

\begin{align*}
\text{QoI scalar } & \frac{\partial Q}{\partial u} \\
\text{QoI derivative vector } & \frac{\partial Q}{\partial \phi} \\
\text{Jacobian matrix } & \frac{J}{\partial \phi}
\end{align*}

Efficient: Requires one linear adjoint solve per QoI, one dot product per parameter.

Fully finite differenced sensitivities require one nonlinear forward solve per parameter.

References


Thanks to Benjamin Kirk, John Peterson, Vikram Garg, Varis Carey, Derek Gaston, Larisa Branets, Graham Carey, Paul Bauman, and many other contributors and collaborators.