libMesh

A New User's Experience

Shayan Hoshyari

April, 2016

Outline

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations
4. Solve the discrete system

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations
4. Solve the discrete system
5. Write out results

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations
4. Solve the discrete system
5. Write out results
6. Optionally estimate error, refine the mesh, and repeat

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations
4. Solve the discrete system
5. Write out results
6. Optionally estimate error, refine the mesh, and repeat

libMesh is a C++ library that offers functionality to handle the tasks above, with the exception of step 3.

Motivation

Major components of a mesh based numerical solution technique:

1. Read the mesh from file
2. Initialize data structures
3. Construct a discrete representation of the governing equations
4. Solve the discrete system
5. Write out results
6. Optionally estimate error, refine the mesh, and repeat

libMesh is a C++ library that offers functionality to handle the tasks above, with the exception of step 3.

Resources

- **libMesh website:**
  https://libmesh.github.io
- **Stable Releases:**
  https://github.com/libMesh/libmesh/releases
- **Development tree:**
  $ git clone git://github.com/libMesh/libmesh.git

Environment Variables

```
# libMesh important directories
export LIBMESH_SRC="/path/to/libmesh/source/dir"
export LIBMESH_DIR="/path/to/libmesh/install/dir"

# Flavour to be used in runtime:
export METHOD="opt"

# If you wish to compile with PETSc
# MPI path is taken from PETSc configuration
export PETSC_ARCH="petsc-architecture"
export PETSC_DIR="/path/to/petsc"
```

Configure and Make

```
# Read through configure help first
./configure --help

# Configure libMesh
./configure METHODS="dbg opt gprof" \
--disable-fortran \
--with-metis=PETSc \
--disable-strict-lgpl \
--prefix=/home/hooshi/code/libmesh/mpich-petsc \
--enable-laspack \
--enable-unique-ptr

# Make and install
make -j 8
make install
```

Linking With libMesh

For small applications use the ‘libmesh-config’ executable found in the install directory.

```
`libmesh-config --cxx` \
-o foo foo.C \
`libmesh-config --cxxflags --include --ldflags`
```

For larger applications include Make.common in your Makefile. See Makefiles in the example folders.

Poisson Problem

- The equation:
  $\Delta u = f \qquad u \in \Omega$

- Penalty method for Dirichlet boundary condition:
  $\frac{1}{\epsilon} u(x) + u_n(x) = \frac{1}{\epsilon} u_0(x) \qquad x \in \partial\Omega$

- A mesh with appropriate boundary tags should be available in one of the many supported formats, e.g., gmsh, Triangle and Tetgen.

Supported Mesh Elements

Effect

- The mapping from the reference element to the computational element.
- Limits on the available finite element basis functions.

- 2, 3, and 4 noded edges (Edge2, Edge3, Edge4)
- 3 and 6 noded triangles (Tri3, Tri6)
- 4, 8, and 9 noded quadrilaterals (Quad4, Quad8, Quad9)
- 4 and 10 noded tetrahedrals (Tet4, Tet10)
- 8, 20, and 27 noded hexahedrals (Hex8, Hex20, Hex27)
- 6, 15, and 18 noded prisms (Prism6, Prism15, Prism18)
- 5, 13, and 14 noded pyramids (Pyramid5, Pyramid13, Pyramid14)

The main function

```
//Assembe Function to be introduced later
void assemble_poisson(EquationSystems& es, const string&);

int main (int argc, char** argv)
{
  // Initialize libraries.
  LibMeshInit init (argc, argv);

  // Create a mesh
  Mesh mesh(init.comm());

  // Read the mesh
  GmshIO gmsh(mesh);
  gmsh.read("../mesh/simple_box_5.msh");
  mesh.all_second_order(true);
  mesh.prepare_for_use();

 // Create an equation systems object.
  EquationSystems equation_systems (mesh);
```

The main function

```
// Add a system to be solved
equation_systems.
add_system<LinearImplicitSystem> ("Poisson");

// Add a variable
equation_systems.get_system("Poisson")
.add_variable("u", SECOND, HIERARCHIC);

// Attach the assembler function
equation_systems.get_system("Poisson")
.attach_assemble_function (assemble_poisson);
```

The main function

```
  // Initialize the data structures.
  equation_systems.init();

  // Solve the system
  equation_systems.get_system("Poisson").solve();

  // write the results
  VTKIO (mesh).write_equation_systems
  ("out.pvtu", equation_systems);

  return 0;
}
```

Systems

Some useful ones

- ExplicitSystem
- LinearImplicitSystem
- TransientLinearImplicitSystem
- DifferentiableSystem
- FEMSystem

Finite Element Families

- Lagrange (Up to quadratic)
- High Order C0
  - Hierarchic
  - Bernstein
  - Szabo-Babuska
- C1 elements
  - Hermite
  - Clough-Tocher
- Discontinuous elements
  - Monomials
  - L2-Lagrange
  - L2-Hierarchic
- Vector-valued elements
  - Lagrange-Vec
  - Nedelec first type
  - No Raviart-Thomas as of now

Assemble Function

Nomenclature for the *i*th degree of freedom and *q*th quadrature point (volume or surface).

Code	Math	Description
`JxW[q]`	$\|J(\xi_q)\|w_q$	Jacobian times weight
`phi[i][q]`	$\phi_i(\xi_q)$	value of i^{th} shape fn.
`dphi[i][q]`	$\nabla\phi_i(\xi_q)$	value of i^{th} shape fn. gradient
`d2phi[i][q]`	$\nabla\nabla\phi_i(\xi_q)$	value of i^{th} shape fn. Hessian
`xyz[q]`	$x(\xi_q)$	location of ξ_q in physical space
`normals[q]`	$\vec{n}(x(\xi_q))$	normal vector at x on a side

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)|w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)|w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Assembling Local LHS and RHS

```
for(q=0; q<Nq; ++q)
  for (i=0; i<Ns; ++i)
  {
   Fe(i) += JxW[q] * f(xyz[q])*phi[i][q];

   for (j=0; j<Ns; ++j)
    Ke(i,j) += JxW[q]*(dphi[j][q]*dphi[i][q]);
  }
```

$$F_i^e = \sum_{q=1}^{N_q} f(x(\xi_q))\ \phi_i(\xi_q)\ |J(\xi_q)| w_q$$

$$K_{ij}^e = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q)\ |J(\xi_q)| w_q$$

Further Functionality

- Runtime selection of solver package: PETSc, Trillinos, Laspack
- Code can be parallelized with minor changes through threading or MPI
- Various error estimation algorithms
- Mesh adaptation and repartitioning in parallel
- Discontinuous Galerkin Methods

References

Roy Stogner, Derek Gaston
libMesh Finite Element Library.
http://users.ices.utexas.edu/~roystgnr/libmeshpdfs/roystgnr/sandia_libmesh.pdf

Benjamin S. Kirk, John W. Peterson, Roy H. Stogner
The libMesh Finite Element Library.
http://www.training.prace-ri.eu/uploads/tx_pracetmo/libmesh.pdf