PECOS

Predictive Engineering and Computational Sciences

PDE Discretization and Analysis with libMesh

Roy H. Stogner[1]

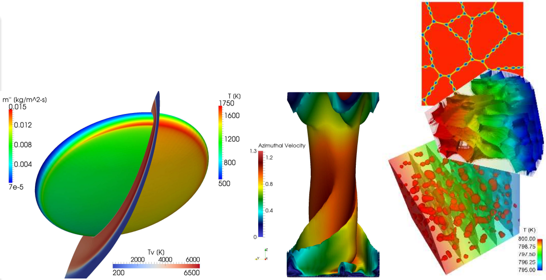
[1] The University of Texas at Austin

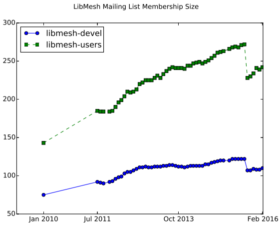
Feb 26, 2016

libMesh Finite Element Library

Scope

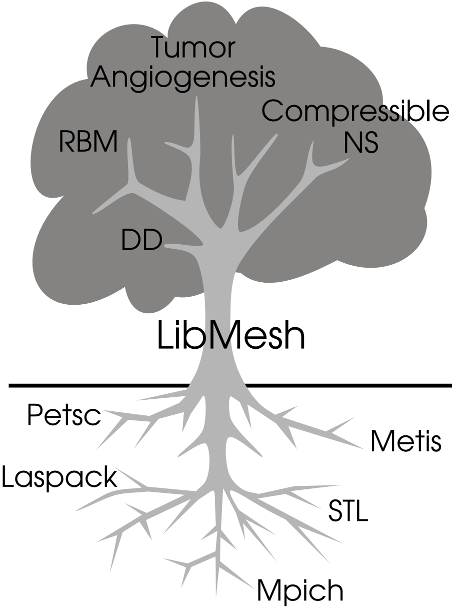
- Free, Open source
  - LGPL2 for core
- 35 Ph.D. theses, 393 papers (58 in 2015)
- ~ 10 current developers
- $160 - 240$ current users?





Challenges

- Radically different application types
- Widely dispersed core developers
  - INL, UT-Austin, U.Buffalo, JSC, MIT, Harvard, Argonne
- OSS, commercial, private applications



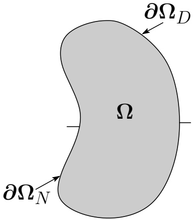
- Foundational (typically optional) library access via LibMesh's "roots".
- Application "branches" built off the library "trunk".
- Additional middleware layers (e.g. Akselos, GRINS, MOOSE) for more complex applications

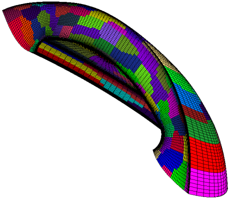
Typical Boundary Value Problem

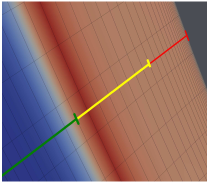
- Common BVP components:

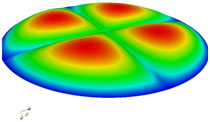
$$
\begin{aligned}
\boldsymbol{M}\frac{\partial \boldsymbol{u}}{\partial t} &= \boldsymbol{F}(\boldsymbol{u}) && \in \Omega \subset \mathbb{R}^n \\
\boldsymbol{G}(\boldsymbol{u}) &= 0 && \in \Omega \\
\boldsymbol{u} &= \boldsymbol{u}_D && \in \partial\Omega_D \\
\boldsymbol{N}(\boldsymbol{u}) &= 0 && \in \partial\Omega_N \\
\boldsymbol{u}(\boldsymbol{x}, 0) &= u_0(\boldsymbol{x})
\end{aligned}
$$

- Less common components:
  - Moving domain $\Omega(t)$, $\Omega(\boldsymbol{u}, t)$
  - Multi-dimensional manifolds
  - Self-overlapping, contact
  - Acceleration $\partial^2 u / \partial t^2$
  - Integro-differential equations

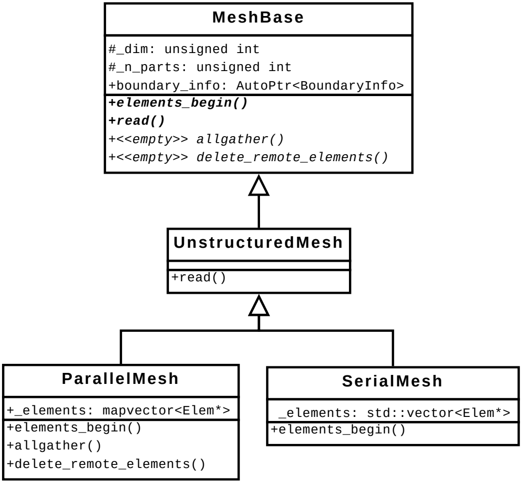




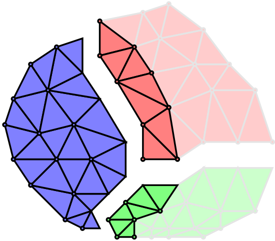




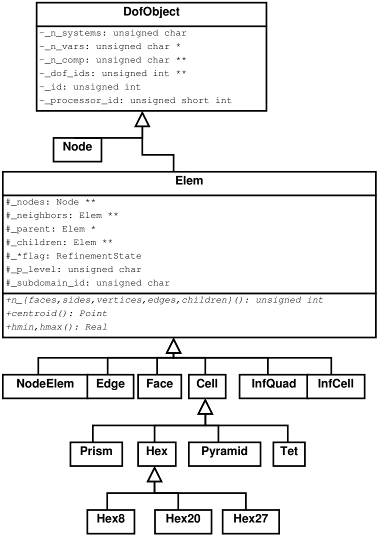
Mesh Classes



- `MeshBase` gives node or element iterators, all or active, global or local
- `SerialMesh` or `ParallelMesh` manages synchronized or distributed data



Geometric Element Classes



- Abstract interface gives mesh topology
- Concrete instantiations of mesh geometry
- Hides element type from most applications
- Base class data arrays allow more optimization, inlining

Similar trees:

- Shape functions, quadrature
- Periodic boundary maps
- Reference counts, comms
- Linear algebra, solvers
- Integrators, strategies

Poisson Example

- Consider the weak form arising from the Poisson equation,

$$\mathcal{R}(u,v) := \int_{\Omega^h} \left[\nabla u \cdot \nabla v - fv\right] dx = 0 \qquad \forall v \in \mathcal{V}$$

Poisson Example

- The LibMesh representation of the matrix and rhs assembly is similar to the mathematical statements.

```
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]);
  }
```

Poisson Example

- The LibMesh representation of the matrix and rhs assembly is similar to the mathematical statements.

```
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]);
  }
```

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

Poisson Example

- The LibMesh representation of the matrix and rhs assembly is similar to the mathematical statements.

```
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]);
  }
```

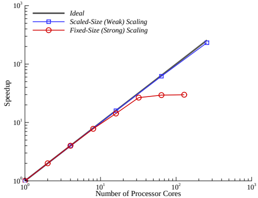
$$\boldsymbol{K}_{ij}^{e} = \sum_{q=1}^{N_q} \nabla\phi_j(\xi_q) \cdot \nabla\phi_i(\xi_q) |J(\xi_q)| w_q$$

Scalability: Hybrid MPI + Threads

Parallel:: interfaces wrap MPI

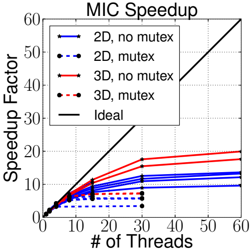
Library data is distributed, synchronized:

- Simpler, STL-compatible, inlined API
- Works with complex serializable classes



Threads:: interfaces TBB/pthreads/OpenMP

- Sparsity, AMR constraint calculation
- `FEMSystem` assembly routines
- Asynchronous I/O
- Mesh projections, utility functions



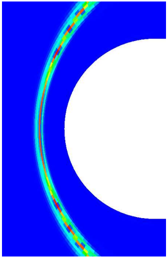
- Hybrid parallel apps with *no* threading or message passing code
- API-independence via `libMesh` wrappers

Adaptivity: Error Estimators, Error Indicators

Error decompositions

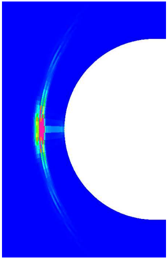
Subterms on each element K:

- $||u - u_h||^2_{\mathcal{H}} = \sum_K ||u - u_h||^2_{\mathcal{H}(\mathcal{K})} \leq \sum_K |\eta_K|^2$
- $Q(u) - Q(u_h) \approx \sum_K \eta_K$
- $|Q(u) - Q(u_h)| \leq \sum_K |\eta_K|$



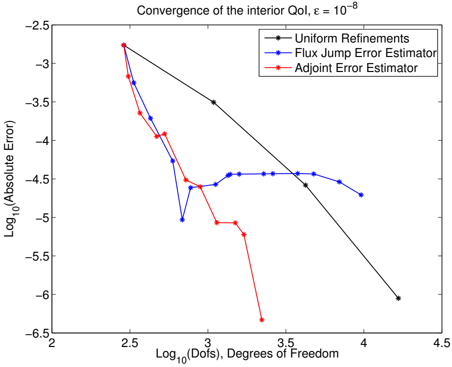
Refinement heuristics

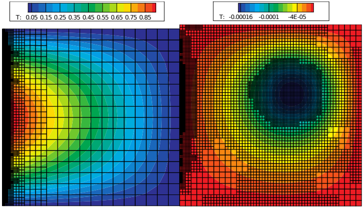
- Refinement/coarsening of elements with:
  - Worst/best fraction sorted by error
  - Error over/under fraction of tolerance
  - Error over/under target mesh size average
- h-vs-p refinement:
  - *a priori* singularity identification
  - Behavior vs. cost when h vs. p coarsening?



Goal-oriented Adaptivity

Refine to reduce solution error *when it influences QoI error*:





- Global error indicator targets layer alone; QoI temporarily plateaus
- Rapid convergence from adjoint-based refinement

Adjoint-based Error Indicators, Sensitivities

$$\begin{aligned} Q(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}) - Q(\tilde{\boldsymbol{u}}) &= \mathcal{R}(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}, \tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}}) + H.O.T. \\ q' &= Q_{\boldsymbol{\xi}}(\tilde{\boldsymbol{u}}; \boldsymbol{\xi}) - \mathcal{R}_{\boldsymbol{\xi}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{z}} - \Psi(\tilde{\boldsymbol{u}}; \boldsymbol{\xi}); \boldsymbol{\xi}) \end{aligned}$$

Adjoint Refinement

$$\mathcal{R}(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}, \tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}}) \approx \sum_E \mathcal{R}^E(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}, \tilde{\boldsymbol{z}}^{\boldsymbol{H}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}})$$

Adjoint Residual

$$\left|\mathcal{R}(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}, \tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}})\right| \leq \sum_E ||\mathcal{R}^E_{\boldsymbol{u}}||_{B(\mathbf{U}^E, \mathbf{V}^{E*})} \left|\left|\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{u}}^{\boldsymbol{h}}\right|\right|_{\mathbf{U}^E} \left|\left|\tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}}\right|\right|_{\mathbf{V}^E}$$

Generalized Adjoint Residual

$$\left|\mathcal{R}(\tilde{\boldsymbol{u}}^{\boldsymbol{h}}, \tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}}^{\boldsymbol{h}})\right| \leq \sum_E \left|\left|\tilde{\boldsymbol{z}}_i - \tilde{\boldsymbol{z}}^{\boldsymbol{h}}{}_i\right|\right|_i M_{ij} \left|\left|\tilde{\boldsymbol{u}}_j - \tilde{\boldsymbol{u}}^{\boldsymbol{h}}{}_j\right|\right|_j$$

Collaboration Strategies

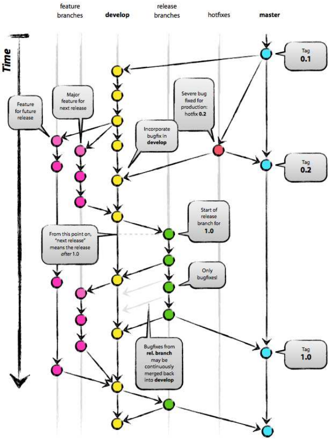
Communication

- Face to face, instant messaging, teleconference
- Email lists
  - `libmesh-users@sourceforge.net`, `libmesh-devel@sourceforge.net`
- Trac/Redmine/Sourceforge issue tracking
- GitHub issues

Code

- Email attachments
- Ticket attachments
- Repository forks!
- Pull requests!

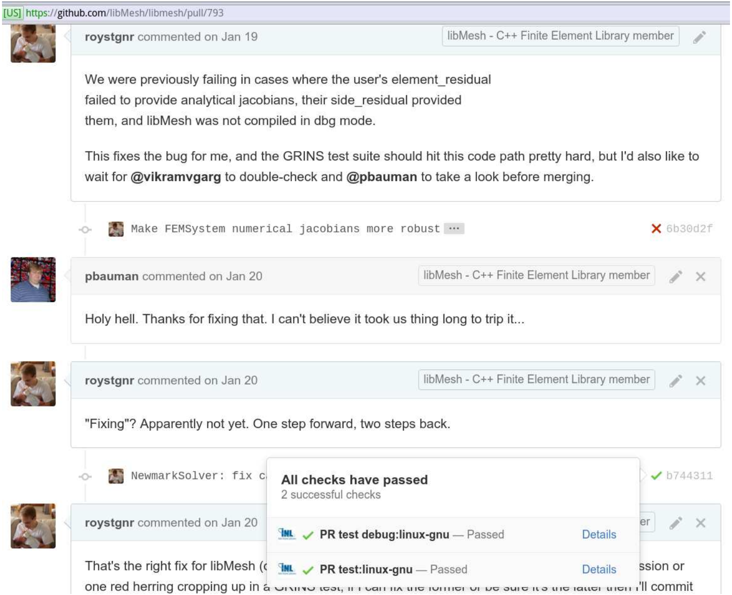
Git Guidelines



- Strive for "useful nonlinearity."
- Develop separate feature sets on separate branches; merge them back to master when complete.
- Minimize or eliminate periodic or unnecessary `merge` commits.
- `rebase` feature branches on top of master before merge + push
- Rebasing public branches is bad™. Complete the shared branch, branch from the shared branch locally, `rebase` from target, then `merge`

- `http://nvie.com/posts/a-successful-git-branching-model`

Issue, Pull Request Tracking



Tracking API Changes

API versions easily proliferate...

```
#if PETSC_VERSION_LESS_THAN(3,1,0)
  ierr = MatGetSubMatrix(matrix->mat(),
           _restrict_to_is, _restrict_to_is_complement,
           PETSC_DECIDE, MAT_INITIAL_MATRIX, &submat1);
#else
  ierr = MatGetSubMatrix(matrix->mat(),
           _restrict_to_is, _restrict_to_is_complement,
           MAT_INITIAL_MATRIX, &submat1);
#endif
```

- Maintain a wide range of external compatibility
  - Dropped PETSc 2.3.3 (2007) support for libMesh 1.0 (2016)
  - C++11 shims
- Limit `libMesh` API changes

Signaling API Changes

Development practices

- Old, new APIs *overlap*
- Easier with C++ function overloading, default arguments
  - Adding `f(a,b)` does not preclude keeping `f(a)`
  - Adding `f(a,b=default)` can replace `f(a)`

Runtime warnings

- `libmesh_experimental()` (in-flux APIs)
- `libmesh_deprecated()` (∼1 year, 1-2 releases)

Examples

- `OStringStream` workaround class
- `Parallel::` global functions

Regression Testing

Fetch and Branch	Toggle output	Time	0:00:04
Build libmesh	Toggle output	Time	0:01:20
Build framework	Toggle output	Time	0:04:37
Pull Bison	Toggle output	Time	0:00:05
Pull Thermochimica	Toggle output	Time	0:00:00
Build Bison	Toggle output	Time	0:04:09
Test Bison	Toggle output	Time	0:00:36
Pull Marmot	Toggle output	Time	0:00:06
Build Marmot	Toggle output	Time	0:03:39
Test Marmot	Toggle output	Time	0:00:35
Pull Yak	Toggle output	Time	0:00:11
Build Yak	Toggle output	Time	0:06:56
Test Yak	Toggle output	Time	0:00:23
Pull Grizzly	Toggle output	Time	0:00:03
Build Grizzly	Toggle output	Time	0:00:34
Test Grizzly	Toggle output	Time	0:00:24
Pull Bighorn	Toggle output	Time	0:00:03
Build Bighorn	Toggle output	Time	0:01:03
Test Bighorn	Toggle output	Time	0:00:01
Pull Rattlesnake	Toggle output	Time	0:00:25
Build Rattlesnake	Toggle output	Time	0:07:12
Test Rattlesnake	Toggle output	Time	0:00:09
Pull Pika	Toggle output	Time	0:00:01
Build Pika	Toggle output	Time	0:00:26
Test Pika	Toggle output	Time	0:00:36
Pull Redback	Toggle output	Time	0:00:01
Build Redback	Toggle output	Time	0:00:21
Test redback	Toggle output	Time	0:00:13
Pull Ferret	Toggle output	Time	0:00:01
Build Ferret	Toggle output	Time	0:01:48
Test Ferret	Toggle output	Time	0:02:11
Pull Hyrax	Toggle output	Time	0:00:00
Build Hyrax	Toggle output	Time	0:00:00
Test Hyrax	Toggle output	Time	0:00:00
Pull Relap-7	Toggle output	Time	0:00:00
Build Relap-7	Toggle output	Time	0:00:00
Test Relap-7	Toggle output	Time	0:00:00
Cleanup	Toggle output	Time	0:00:00

Fetch and Branch	Toggle output	Time	0:00:06	Exit status	0
Configure	Toggle output	Time	0:00:34	Exit status	0
Build	Toggle output	Time	0:01:50	Exit status	0
Install	Toggle output	Time	0:00:14	Exit status	0
Check	Toggle output	Time	0:03:31	Exit status	2

- ~ 7000 internal assertions in debug mode
- ~ 60 core example applications
- ~ 400 unit tests
- Configurations: optional features, index width, ParallelMesh, solver package, scalar precision, -np, –n_threads
- Middleware, application test suites

Assertions

```
libmesh_assert(c < _variables.size());
libmesh_assert(s < elem->n_sides());
libmesh_assert((ig >= Ug.first_local_index()) &&
               (ig < Ug.last_local_index()));
libmesh_assert(requested_ids[p].size() == ghost_objects_from_proc[p]);

libmesh_assert(obj_procid != DofObject::invalid_processor_id);
libmesh_assert(mesh.is_prepared());
MeshTools::libmesh_assert_valid_node_procids(mesh);

libmesh_assert(neigh->has_children());
libmesh_assert(error_estimator.error_norm.type(var) == H1_SEMINORM ||
               error_estimator.error_norm.type(var) == W1_INF_SEMINORM)
libmesh_assert(number_h_refinements > 0 || number_p_refinements > 0);
```

Assertions

```
libmesh_assert(c < _variables.size());
libmesh_assert(s < elem->n_sides());
libmesh_assert((ig >= Ug.first_local_index()) &&
               (ig < Ug.last_local_index()));
libmesh_assert(requested_ids[p].size() == ghost_objects_from_proc[p]);

libmesh_assert(obj_procid != DofObject::invalid_processor_id);
libmesh_assert(mesh.is_prepared());
MeshTools::libmesh_assert_valid_node_procids(mesh);

libmesh_assert(neigh->has_children());
libmesh_assert(error_estimator.error_norm.type(var) == H1_SEMINORM ||
               error_estimator.error_norm.type(var) == W1_INF_SEMINORM)
libmesh_assert(number_h_refinements > 0 || number_p_refinements > 0);
```

- Trust preconditions/postconditions, not any of your users

Assertions

```
libmesh_assert(c < _variables.size());
libmesh_assert(s < elem->n_sides());
libmesh_assert((ig >= Ug.first_local_index()) &&
               (ig < Ug.last_local_index()));
libmesh_assert(requested_ids[p].size() == ghost_objects_from_proc[p]);

libmesh_assert(obj_procid != DofObject::invalid_processor_id);
libmesh_assert(mesh.is_prepared());
MeshTools::libmesh_assert_valid_node_procids(mesh);

libmesh_assert(neigh->has_children());
libmesh_assert(error_estimator.error_norm.type(var) == H1_SEMINORM ||
               error_estimator.error_norm.type(var) == W1_INF_SEMINORM)
libmesh_assert(number_h_refinements > 0 || number_p_refinements > 0);
```

- Trust preconditions/postconditions, not any of your users
- Earlier errors are easier errors

Assertions

```
libmesh_assert(c < _variables.size());
libmesh_assert(s < elem->n_sides());
libmesh_assert((ig >= Ug.first_local_index()) &&
               (ig < Ug.last_local_index()));
libmesh_assert(requested_ids[p].size() == ghost_objects_from_proc[p]);

libmesh_assert(obj_procid != DofObject::invalid_processor_id);
libmesh_assert(mesh.is_prepared());
MeshTools::libmesh_assert_valid_node_procids(mesh);

libmesh_assert(neigh->has_children());
libmesh_assert(error_estimator.error_norm.type(var) == H1_SEMINORM ||
               error_estimator.error_norm.type(var) == W1_INF_SEMINORM)
libmesh_assert(number_h_refinements > 0 || number_p_refinements > 0);
```

- Trust preconditions/postconditions, not any of your users
- Earlier errors are easier errors
- Exceptions! Stack traces! Line numbers!

Manufactured Solution Testing

(Finding unknown unknowns)

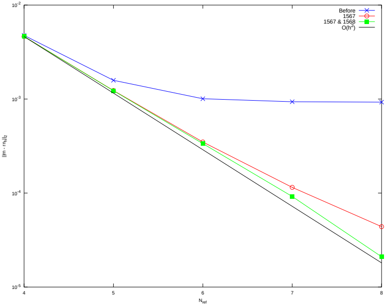
Verification of FIN-S hypersonics code

- FANS, Spalart-Allmaras
- Derivative:

$$\frac{d(sa)}{dx} = \frac{1}{\rho} * \left(\frac{d(\rho * sa)}{dx} - sa\frac{d\rho}{dx} \right)$$

- In code:

$$\frac{d(sa)}{dx} = \frac{1}{\rho} * \frac{d(\rho * sa)}{dx} - sa\frac{d\rho}{dx}$$



Manufactured Analytical Solution Abstraction library:
`https://manufactured-solutions.github.io/MASA/`

Autotools, Pros and Cons

Autoconf

- Manages feature selection
  - 50+ `--enable-foo` options
- Portability tests, workarounds
- POSIX shell dependence

Libtool

- DLL management in install
- Broader shared library support
- Easily used via automake
- Trickier in-build debugging

Automake

- dist, check, install targets
- Out-of-source builds
- *Standardized conventions*
- More difficult METHODS support
- “bootstrap” process
  - Do users have autotools?
  - Custom scripts for libMesh

Acknowledgements

Recent contributors:

- David Andrs
- Paul Bauman
- Vikram Garg
- Derek Gaston
- Dmitry Karpeev
- Benjamin Kirk
- David Knezevic
- Cody Permann
- John Peterson
- Sylvain Vallaghe

Useful resources:

- libMesh: `https://libmesh.github.io/`
- GRINS: `https://grinsfem.github.io/`
- MOOSE: `https://mooseframework.org/`
- MASA: `https://manufactured-solutions.github.io/MASA/`

Questions?