High-Performance-Scientific-Computing Community Analysis and Contribution Project

CSCI-5576

John Patterson

Computer Science
University of Colorado at Boulder

11.22.19

1 Introduction and Motivation

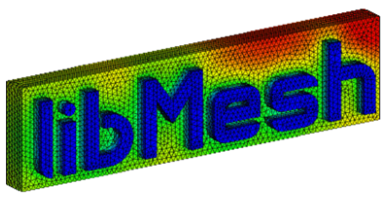
The intention of this community analysis and contribution project was to examine meshing libraries and common usage in the high performance scientific community. My research group has used the SCOREC libraries for prepossessing of the problem domains we examine. Since I am somewhat competent in the usages and format of the SCOREC code, I have made a contribution here. In addition, the libMesh group has a large contributor base convenient for the community analysis portion of this project. This allowed me to compare libMesh's software package to better my view of the commonplace functionalities of meshing software, as well as to broaden my view of scientific software contribution as a whole.

2 Community Analysis: LibMesh

LibMesh is a Finite element library intended to produce unstructured meshes in parallel. These meshes have been shown to provide adequate mesh data structures for a variety of physical problem domains of interest.

Of the 1077 website-listed publications that use libmesh a few applications include, reacting flows of a turbine machinery, discretion studies of the spallart-allmaras turbulence models, and tumor growth modeling.

The libMesh community began as a project to put University of Texas at Austin Professor Gram F. Carey's teachings into a public repository. The result was a large open-source finite-element meshing library. Some statistics of the project are shown below:



Repository URL	https://github.com/libMesh/libmesh
Documentation website	http://libmesh.github.io/index.html
Project Start Date	2002
Contributors in the last year	25
Contributors in the lifetime	62
Development Discussions	github issues, sourceforge comments, and email lists
Commits per week	≈ 21
Content per Commit	≈ 170 additions and 50 deletions
Listed Affiliations	The University of Texas at Austin CFDlab Technische Universitat Hamburg-harburg Institute of Modelling and Computation UT-Austin PECOS Center Idaho National Laboratory Computational Frameworks group NASA Lyndon B Johnson Space Center Akselos Inc. Patera at MIT

Since the code is open-sourced the contributions are also welcomed. In normal github fashion, the group outlines the guidelines for code revision/appending and accepts pull requests via github. To be included in developmental conversation, github issues as well as a mailing list at https://sourceforge.net/p/libmesh/mailman/ are open to the public. Contributed works must undergo unit testing before a merge to the master branch is completed.

libMesh's LGPL license reflects the affiliations that are using the tool. This particular license allows the software that accesses libMesh to be open or closed-source. Even though closed-sorce projects that use libMesh need not disclose their source code, they are required to disclose the modifications to the libMesh code they altered to utilize it. This is in contrast to a GPL licence in that the software that uses it must be open-source as well.

In personal trials of libMesh usage, the installation documentation is clear and strait forward, however, the procedure took the better part of a day. The installation required a minimal amount of external, but very pivotal high performance scientific computing libraries with the raw install. The developers created a framework that not only supports a large number of optional external libraries but also, further implementation of previously unsupported libraries is clear. Documentation on compiling personal code that uses the libMesh libraries are available.

libMesh also includes by default, common finite element operations. A few are listed :

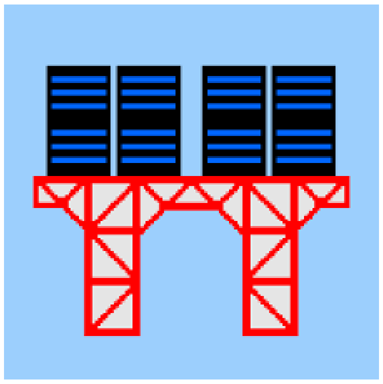
- 1,2,and 3D Finite Elements
- Partitioning Algorithms. (including Hilbert and Z-Curves)
- A Variety of Finite Element families
- A host of IO format translation utilities
- Mesh creation and modification utilities
- Tabulated Quadrature to 44th order

Sections of example codes demonstrating how to perform operations and solves are demonstrated in commented code at **http://users.oden.utexas.edu/~roystgnr/libmeshdoc/index.php**. Some example operations include:

- Introductory lessons:
  - Solving the Possion Problem in 1,2, and 3 dimensions
- Solving a System of equations
  - Solving the Stokes equations
- Transient Systems
- Mesh Adaptivity
- Eigenvalue problems
- Restriction to sub-domains
- Solution Transfer from one Domain to Another

3 My Community Contribution: SCOREC-core

SCOREC-core is a set of useful libraries largely maintained by Rensselaer Polytechnic Institute's **S**cientific **CO**mputation **RE**search **C**enter for generation and preparation of Parallelized Unstructured grids on high performance machines. In the workflow of CU's in-house PHASTA flow solver, SCOREC-core is used to generate and partition unstructured meshes.

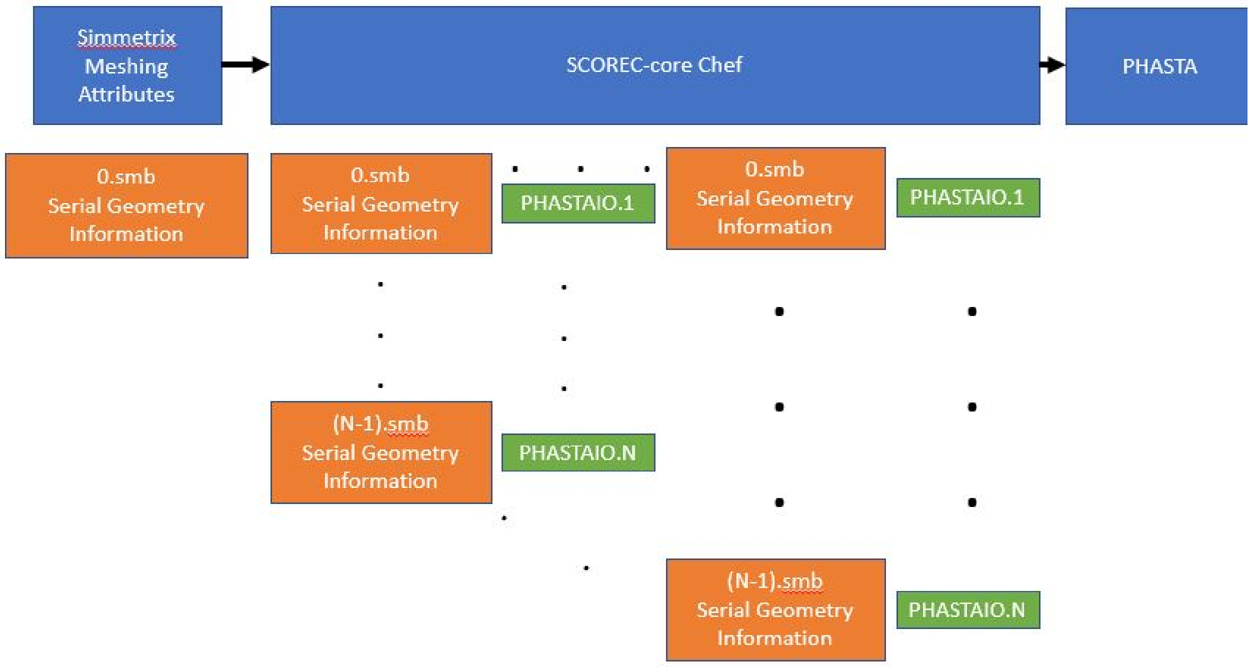


The PHASTA flow solver works to find solutions to Compressible fluid flow. In the research problems that the PHASTA group is concerned with, there are five equations that govern the dynamics of the motion of the fluid.

- Conservation of mass (1 equation)
- Conservation of Momentum (3 equations)
- Conservation of Energy (1 equation)

The methodology of the PHASTA flow solver in a Direct Numerical Simulation (DNS) is to make a system of 5 equations and 5 unknowns. A pivotal research interest of the PHASTA group is focused around finding effective turbulence models that cut down on computational time. **This often results in solving additional quantities for each element that are attempting to model (or cleverly approximate) some aspect of the physics that results in a quicker simulation.** The additional modeled quantities are field variables and for each mesh entity degree of freedom, an increase of a factor of 5.

The current PHASTA workflow generates a problem domain's mesh and partitions that mesh into (often) many thousand's of parts. This pre-proccessing results from the SCOREC-core tools "generate" and "chef".


Simmetrix Meshing Attributes
SCOREC-core Chef
PHASTA
0.smb Serial Geometry Information
0.smb Serial Geometry Information
PHASTAIO.1
0.smb Serial Geometry Information
PHASTAIO.1
(N-1).smb Serial Geometry Information
PHASTAIO.N
(N-1).smb Serial Geometry Information
PHASTAIO.N

In a huge mesh, the first partition will fail if,

$$nElements \times nDOFs > 2^{31} - 1 = 2^{\sum_i^4 2^i} - 1 = sizeof(4 \times int)$$

A recently generated 385 million element domain intended for a high Reynolds number Supersonic Compression ramp DNS, provided an effective test case to demonstrate that partitioning via chef was successful with 5 degrees of freedom per mesh entity and failure for anything larger:

$$\begin{aligned} log_2(385,314,490) + log_2(5) &= 28.51 + 2.3219 \\ &= 30.8424 < 31 \\ log_2(385,314,490) + log_2(6) &= 28.51 + 2.584 \\ &= 31.1 > 31 \end{aligned}$$

The intended improvement would increase the size of the array that is intended to make the original serial mesh:

$$\begin{aligned} log_2(vars) + log_2(nodes) &< log_2(sizeof(unsignedlonglong)) \\ &< 256 \end{aligned}$$

Modification to the code was made within phRestart.cc to the following routine.

```
void attachZeroSolution(Input& in, apf::Mesh* m)
```

```
int vars = in.ensa\_dof;

int nodes = m->count(0);

unsigned long long sizeM = ((unsigned long long) vars)*((unsigned long long) nodes);

double* data = new double[sizeM]();

attachField(m, "solution", data, vars);

delete [] data;
```

Although the mesh entity degrees of freedom are not a visualizable quantity until the solution is created from running the simulation, the mesh below was the trial mesh used for this code improvement. Further improvements to the chef code may include solution transfer from on domain to another.

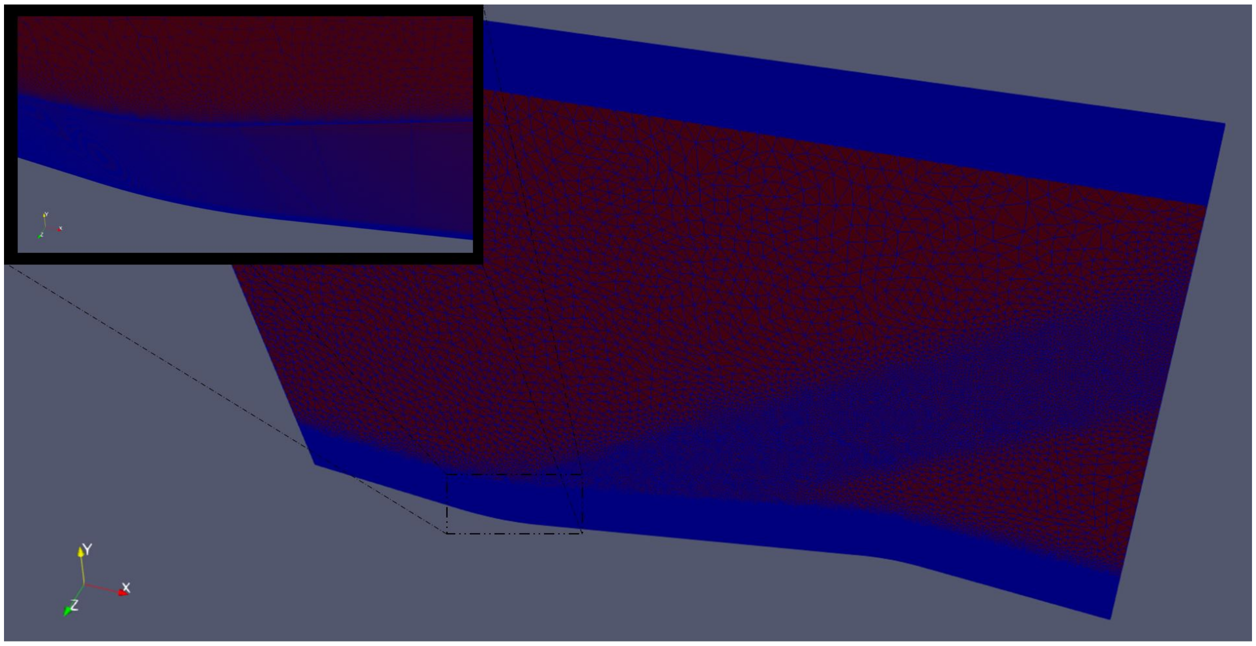


Figure 1: 385M final element mesh