## Software Components

<table>
<thead>
<tr>
<th>Code</th>
<th>From</th>
<th>Lang</th>
<th>LOC</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMMPS</td>
<td>Sandia</td>
<td>C++</td>
<td>118,000</td>
<td>MD simulator</td>
</tr>
<tr>
<td>FVM</td>
<td>Purdue</td>
<td>C++</td>
<td>10,000</td>
<td>Core for FVM solver</td>
</tr>
<tr>
<td>MPM</td>
<td>UNM</td>
<td>F90/F77/C</td>
<td>25,000</td>
<td>Core for MPM solver</td>
</tr>
<tr>
<td>Rar Gas Dyn</td>
<td>Purdue</td>
<td>F90</td>
<td>6,000</td>
<td>Boltzmann-ESBGK</td>
</tr>
<tr>
<td>Reactive MD</td>
<td>Purdue</td>
<td>C</td>
<td>27,500</td>
<td>Reactive force fields</td>
</tr>
<tr>
<td>Dislocations</td>
<td>Purdue</td>
<td>C/F77</td>
<td>2,000</td>
<td>Micromechanical models</td>
</tr>
<tr>
<td>Rappture</td>
<td>Purdue</td>
<td>C/C++/XML</td>
<td>60,000</td>
<td>Input/output handling</td>
</tr>
<tr>
<td>MEMOSA</td>
<td>Purdue</td>
<td>C/C++/Python</td>
<td>10,000</td>
<td>Simulation framework</td>
</tr>
<tr>
<td>Im Boundary</td>
<td>Purdue</td>
<td>C</td>
<td>5,000</td>
<td>FVM / MPM connection</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Purdue</td>
<td>C++</td>
<td>10,000</td>
<td>Sensitivity analysis</td>
</tr>
<tr>
<td>UQ drivers</td>
<td>Purdue</td>
<td>C/Python</td>
<td>5,000</td>
<td>Uncertainty quantification</td>
</tr>
<tr>
<td>Models</td>
<td>Purdue</td>
<td>C/F77</td>
<td>5,000</td>
<td>Various physical models</td>
</tr>
</tbody>
</table>
Project Infrastructure

- Subversion repository
  - FVM, MPM, LAMMPS code under build/test
  - 2.7 million lines of code in total under Subversion control
- Automated build system
  - make + 1,500 lines of Python
  - Configuration files for various platforms
- Wiki for project notes
MEMOSA Architecture

Python Scripts
Solver Suite

Python Bindings
- Finite Volume Method (FVM)
- Material Point Method (MPM)

C/C++ Data Objects

Setup
- ANSYS
- ProE
- CUBIT

Visualization
- ParaView

Compact Models
- LAMMPS

Mesoscale micromechanical models
Object-Oriented Infrastructure

- C++/F90 core for solvers
- C++ templates for arithmetic types
- Python objects for high-level scripting
Python Bindings

.i files for 60 classes (so far)

```cpp
template<class T>
class FlowModel : public Model {
  public:
    FlowModel(const GeomFields& geo);
    virtual ~FlowModel();
    virtual void init();
    void advance(const int iterations);
};
```

fmodel = models.FlowModelA(geomFields);

fmodel.init()
fmodel.advance(numIterations)
...
Regression Tests

☑ Nightly regression tests: 351 tests
  ➔ 252 FVM core
  ➔ 14 MPM core
  ➔ 80 Python/UQ (puq)
  ➔ 5 LAMMPS

☑ Framework for adding new tests
☑ Web-based system for browsing results
from puq import *

def run():
    x = Parameter('x', 'x', min=-2, max=2)
    y = Parameter('y', 'y', mean=0, dev=1)

    #host = InteractiveHost()
    host = PBSHost(env='/scratch/env.sh', qname='standby', walltime='2:00')

    #uq = MonteCarlo([x,y], num=50)
    uq = Smolyak([x,y], level=2)

    prog = TestProgram('./rosen_prog.py')
    return Sweep(uq, host, prog)
UQ for Any Program

local host > sweep start example1.py
Start example1.py
Sweep id is 256586451
Executing ./rosen_prog.py --x=0.0 --y=0.0
Executing ./rosen_prog.py --x=2.0 --y=0.0
Executing ./rosen_prog.py --x=2.0 --y=0.0
... --x=1.7 --y=0.2

Processing <HDF5 dataset "z": shape (13,), type "<f8">>
Calculated response surface:
   RMSE = 7.40e+02 (2.05e+01 %)
   R-Squared = 0.62
   Surface = -345.7 - 2.0*x - 266.7*y + 300.0*y**2 + 301.0*x**2
The response surface has a mean=455.666666664 and stddev=593.021840833.
Live on memsHUB.org

- Run UQ in a Linux workspace
- Course: ME597/AAE590 - Introduction to Uncertainty Quantification

Demo >>
Computational Power

Steele
848 x 8 core Dell 1950
60 teraFLOPS

Coates
1,000 x 8 core HP DL165
All 10 GigE
90 teraFLOPS

LLNL Hera, LANL Lobo
Plans for This Year

- GUI for UQ framework
- DAKOTA integration
- Create a database of PDFs and response surfaces
- Improve connection to ParaView run in parallel
- More regression tests