SAND2013-4084C

Verification Testing for Sierra Mechanics

Brian Carnes
bcarnes@sandia.gov

Validation and Uncertainty Quantification Processes (1544)

SAND2013-XXX

ASME Verification and Validation Symposium
May 22-24, 2013

Sandia National Laboratories

Outline

- Verification overview
- Manufactured solutions using AD
- Verification tests as regression tests
- Auto-generated test documentation
- Feature coverage tool
- Examples from the Sierra verification test suite

Motivation for V&V

- We build codes for simulation of high-consequence events
- How do we know the codes give correct answers?
  1. **Validation**: correct for the overall prediction means code can predict physical events with accuracy
  2. **Solution Verification**: correct means we have quantified errors from approximate math models
  3. **Code Verification**: correct means that the code is free from coding errors and exhibits grid convergence.
- We provide evidence of correctness to our customers

Verification Activities

- At the code development level:
  - Code Verification (this talk)
  - Software Quality Engineering (SQE)
- At the application level:
  - Solution verification
  - A posteriori error estimation
  - Mesh adaptivity

Verification is Quantitative

- Verification tests **must** contain an exact, quantitative criterion based on the underlying math model:
  - Mass/force conservation
  - Symmetry
  - Patch tests
  - Analytic value of a derived quantity
  - Analytic solution for all of the solution variables
- This excludes things like code-to-code comparisons, beam theory, asymptotic solutions, etc.
- The last case is most desirable and enables:
  - Mesh refinement studies and order of accuracy estimation

Code Verification Tools

How can we lower the barriers to high quality code verification?

- Cost to build
- Cost to run
- Cost to maintain
- Difficulty assessing impact

Method of Manufactured Solutions

- Assume fairly simply solution form:

```
U(x,y)=(-pow(x, 2)*pow(y, 2)*pow(-x + 1, 2)*(-y + 1)
+ pow(x, 2)*y*pow(-x + 1, 2)*pow(-y + 1, 2) +
sin(pi_*x)*sin(pi_*y)/(5*lambda_))*sin(pi_*t/2)
```

- Old approach: Use Mathematica or SymPy to generate code for source terms (F_x):

```
F_x(x,y)= -pow(pi_, 2)*rho_*(-pow(x, 2)*pow(y, 2)*pow(-x + 1, 2)*(-y + 1) + pow(x, 2)*y*pow(-x + 1,
2)*pow(-y + 1, 2) + sin(pi_*x)*sin(pi_*y)/(5*lambda_))*sin(pi_*t/2)/4 - (60*lambda_*mu_*pow(x, 4)*y -
30*lambda_*mu_*pow(x, 4) - 120*lambda_*mu_*pow(x, 3)*y + 60*lambda_*mu_*pow(x, 3) +
120*lambda_*mu_*pow(x, 2)*pow(y, 3) - 180*lambda_*mu_*pow(x, 2)*pow(y, 2) +
120*lambda_*mu_*pow(x, 2)*y - 30*lambda_*mu_*pow(x, 2) - 120*lambda_*mu_*x*pow(y, 3) +
180*lambda_*mu_*x*pow(y, 2) - 60*lambda_*mu_*x*y + 20*lambda_*mu_*pow(y, 3) -
30*lambda_*mu_*pow(y, 2) + 10*lambda_*mu_*y - lambda_*pow(pi_, 2)*sin(pi_*x)*sin(pi_*y) +
lambda_*pow(pi_, 2)*cos(pi_*x)*cos(pi_*y) - 3*mu_*pow(pi_, 2)*sin(pi_*x)*sin(pi_*y) + mu_*pow(pi_,
2)*cos(pi_*x)*cos(pi_*y))*sin(pi_*t/2)/(5*lambda_)
```

Method of Manufactured Solutions (2)

- New approach: Use Automatic Differentiation (AD) to build the sources directly in the code:

```
trace_e = diff(u, x) + diff(v, y) +
diff(w, z)

p = (lambda_ + 2*mu_/3)*trace_e

exx = diff(u, x) - trace_e/3
eyy = diff(v, y) - trace_e/3
ezz = diff(w, z) - trace_e/3
exy = (diff(u, y) + diff(v, x))/2
eyz = (diff(v, z) + diff(w, y))/2
exz = (diff(u, z) + diff(w, x))/2
```

```
sxx = 2*mu_*exx + p
syy = 2*mu_*eyy + p
szz = 2*mu_*ezz + p
sxy = 2*mu_*exy
syz = 2*mu_*eyz
sxz = 2*mu_*exz

F_x = -( diff(sxx, x) +
diff(sxy, y) + diff(sxz, z) )
```

Sandia National Laboratories

Verification Testing is Not Regression Testing

- Regression tests for codes:
  - do we always get the same answer (or close)?
  - as
- Verif
  - do
- Main
- Runn
  - Of
  - Do

cost

Time	Num_Nodes	rf_bc_ipo_err	cf_bc_ifo_err	rf_bc_ifo_err
1	45	-1.766622983	-0.906772929	-0.883311491
1	225	-0.386660963	-0.193116706	-0.193330482
1	1377	-0.086498862	-0.042159427	-0.043249431
1	9537	-0.02028748	-0.009741115	-0.01014374
1	70785	-0.004930156	-0.002351035	-0.002465078
1	545025	-0.001219458	-0.000579983	-0.000609729

Time	Num_Nodes	rf_bc_ipo_err	cf_bc_ifo_err	rf_bc_ifo_err
1	45	-1.766622995	-0	
1	225	-0.386660963	-0.	
1	1377	-0.086498862	-0.042159427	-0.043249431
1	9537	-0.02028748	-0.009741115	0.01014374
1	70785	-0.004955616	-0.002352371	-0.002477808
1	545025	-0.001245279	-0.000581337	-0.000622639

Does the test still pass?

Sandia National Laboratories

Verification Testing is Not Regression Testing (2)

- Our solution: develop a test criteria based on rate of convergence (when possible)

Num_Nodes	l2_error	cf_bc_ip	rf_bc_ip	cf_bc_ifc	rf_bc_ifc	src_ipo_	eval_b1_	eval_b1t	eval_s2_err
225	2.52	2.88	2.83	2.88	2.83	4.62	2.21	1.94	2.56
1377	2.27	2.52	2.48	2.52	2.48	4.44	2.83	3.47	5.75
9537	2.14	2.27	2.25	2.27	2.25	4.26	1.71	1.98	-0.16
70785	2.07	2.13	2.11	2.13	2.11	4.14	2.15	2.04	2.37
545025	2.04	2.05	2.03	2.05	2.03	4.07	2.2	1.55	2.44

- Here we can easily diff against 2nd or 4th order with a tolerance using last row of data.

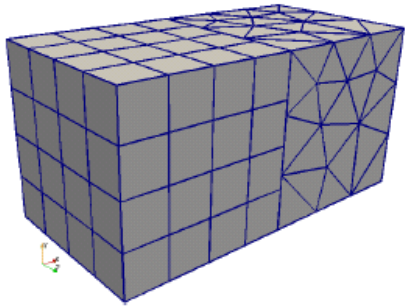
Can focus efforts on problematic variables, here point evaluations of the solution

Sandia National Laboratories

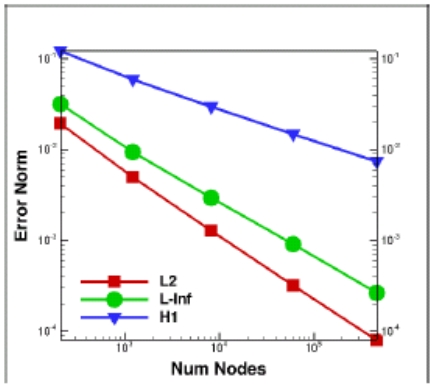
Documentation of VERTS

- Documenting the code is hard, but test suites?
- VERTS documentation:
  - What features tested
  - Error metrics used
  - Test quality (rate of convergence)
  - Latest results
- Most difficult is keeping results up to date
- We have developed auto-documentation tools to generate VERTS documentation on demand
  - Also includes latest input files

Auto-Documentation



Coarse Mesh

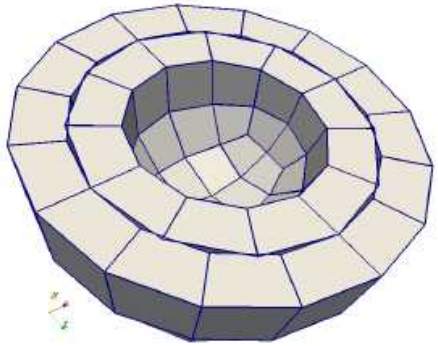


Error Norms

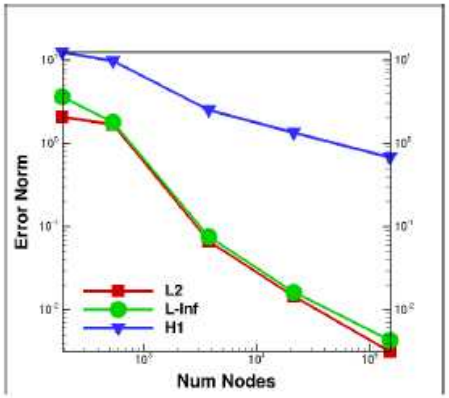
Figure 3.5: 1D Flat Contact: Hex8-Tet4 Tied

Table 3.5: 1D Flat Contact: Convergence Rates for Hex8-Tet4 Tied

Num Dofs	L^2	L^∞	H^1
1231	2.32	2.10	1.24
8284	2.16	1.82	1.09
60570	2.09	1.78	1.04
462800	2.04	1.82	1.02



Coarse Mesh



Error Norms

Figure 3.7: 3D Curved Contact: Hex8-Hex8 Case

Table 3.7: 3D Curved Contact: Convergence Rates for Hex8-Hex8

Num Dofs	L^2	L^∞	H^1
540	0.64	2.01	0.70
3752	5.00	4.91	2.11
21220	2.62	2.66	1.07
150700	2.35	2.05	1.06

Thermal contact verification (non-matching meshes)

- Plots of coarse meshes
- Errors vs. num nodes
- Table with convergence rates (in norms)

Sandia National Laboratories

Nightly Testing

- Old way: VERTS rarely tested, perhaps only before a big release
- Problems are only caught much later
- New approach: VERTS run nightly, but only on a large capacity cluster
  - Developers shielded from burden of running VERTS
  - Most use only 1-2 nodes so have little impact
  - Allows tests to run much longer (up to 2-3 hours)
- Similar approach used for performance testing

Feature Coverage Tool

- Big goal: verify that 100% of the code functions correctly
- Realistic goal: test the most critical code features

- How do we select critical code features?
  - Using input files / meshes from the user community

- Our tool is called Feature Coverage Tool (FCT)
  - Compares an input file against the verification tests
  - Identifies features not covered (gaps)
  - Identifies coverage of pairs of features

Feature Coverage Tool (2)

- The FCT converts input files to colorized HTML:

```
BEGIN ARIA MATERIAL Kryptonite 84+
   Density                = CONSTANT rho = 0.1 70+
   Thermal Conductivity = User_Function Name=Water_Thermal_Conductivity X=T
aria_rtest/verification/solution_verification/solution_verification.test|np8,
aria_rtest/verification/x11b11_nonlin/x11b11_nonlin.test|np1
   Specific Heat          = Encore_Function Name=Cp_Function 1+
   Viscosity              = Constant mu  = 2.0 10+
   heat conduction      = basic 63+
END ARIA MATERIAL Kryptonite
```

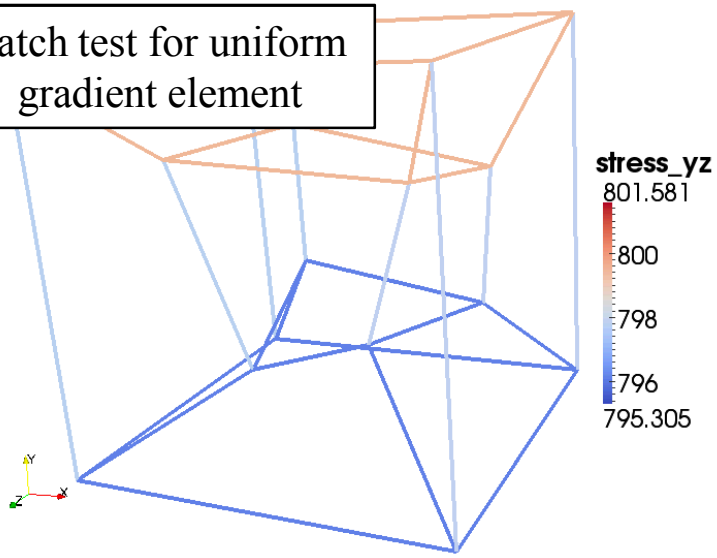
- Feature tested in VERTS (green)
- Tests for feature
- Number of tests for feature
- Untested feature (orange/red)

User can report X% features covered by VERTS or nightly testing

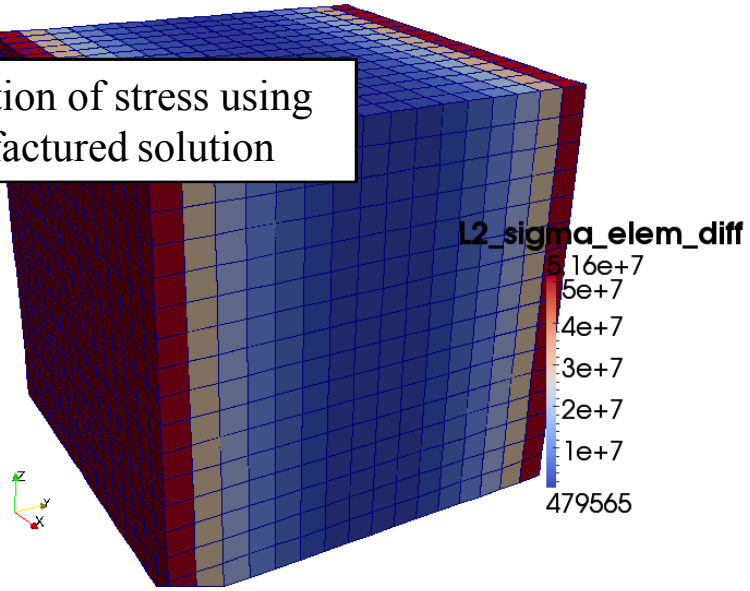
Sandia National Laboratories

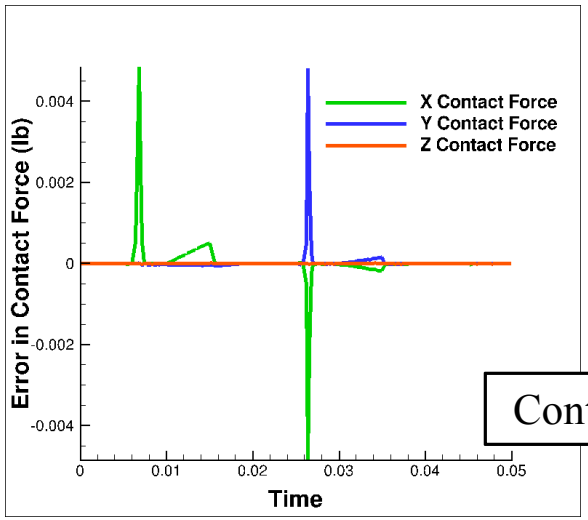
Example: Verification of SIERRA/SM

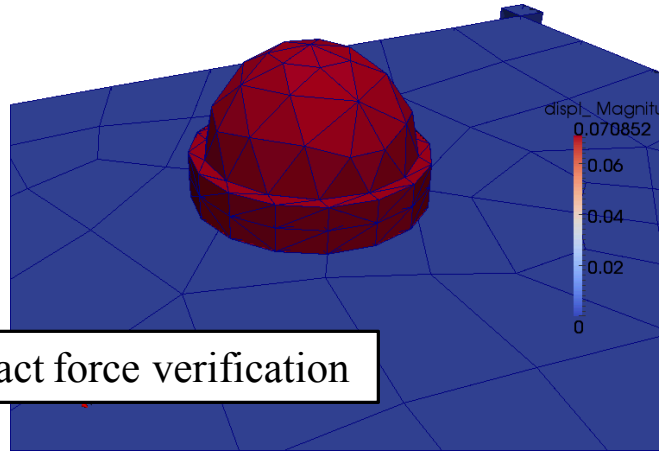
Patch test for uniform gradient element



Verification of stress using manufactured solution



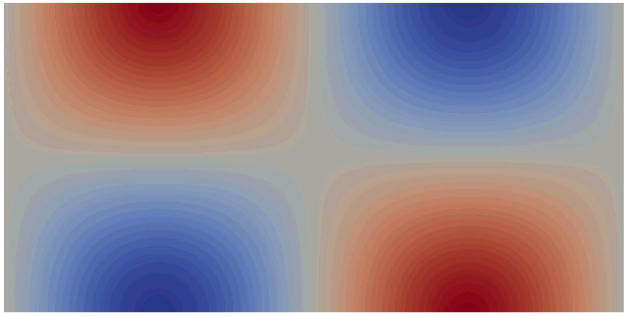


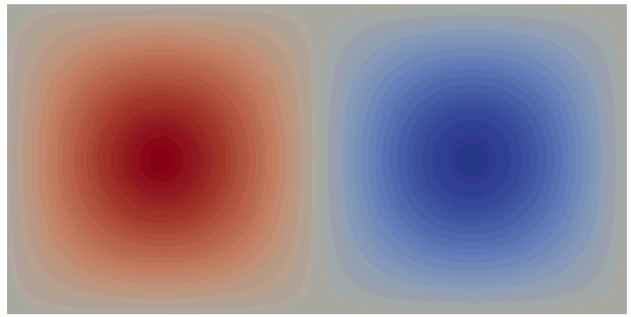


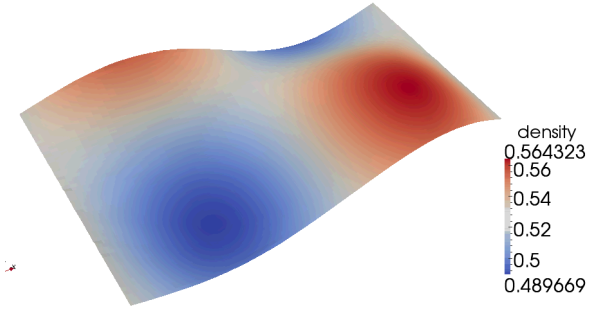
Contact force verification

Sandia National Laboratories

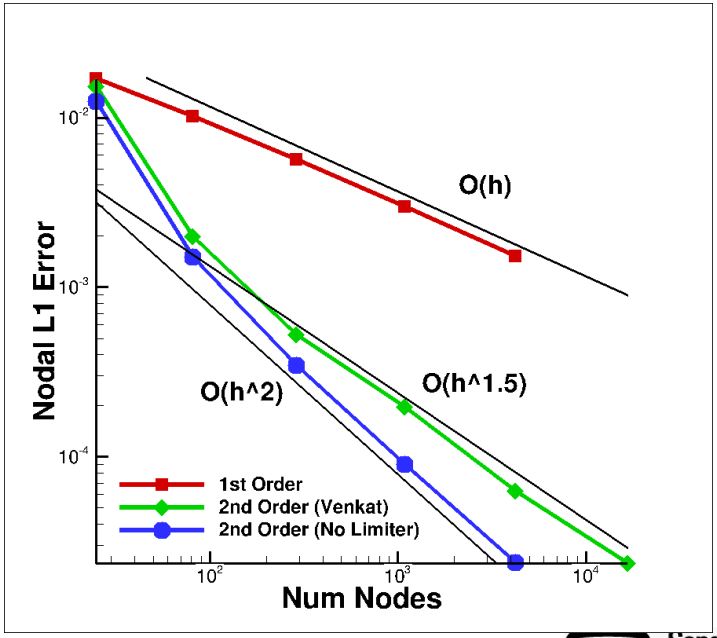
Example: Verification of SIERRA/Conchas







Convergence of Density

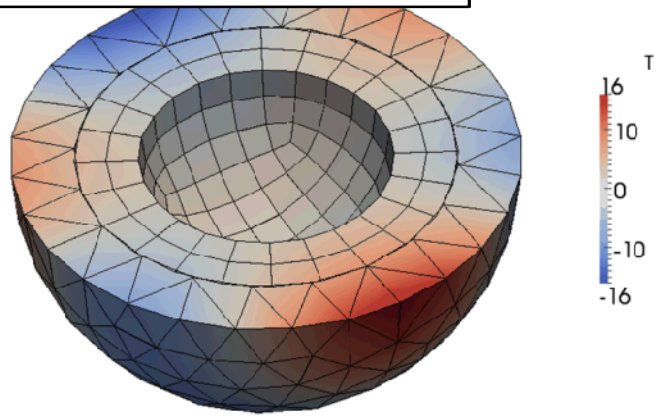


Euler flow: Perturbation of uniform supersonic channel flow (parallel to walls): Y-velocity is 5% of X-velocity

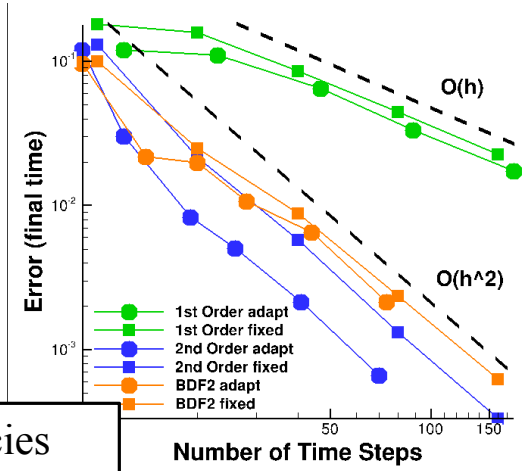
Sandia National Laboratories

Example: Verification of SIERRA/Aria

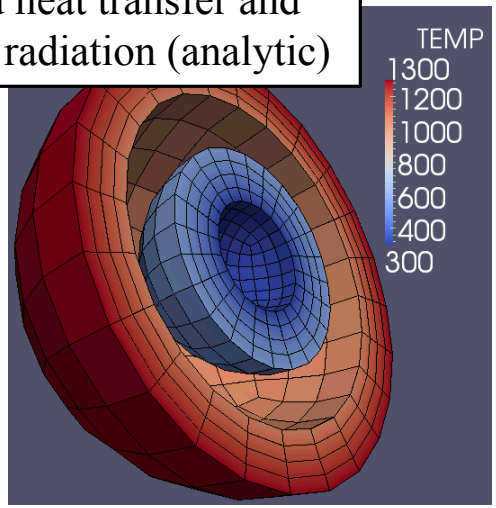
Thermal contact with non-matching grids (manuf. soln.)



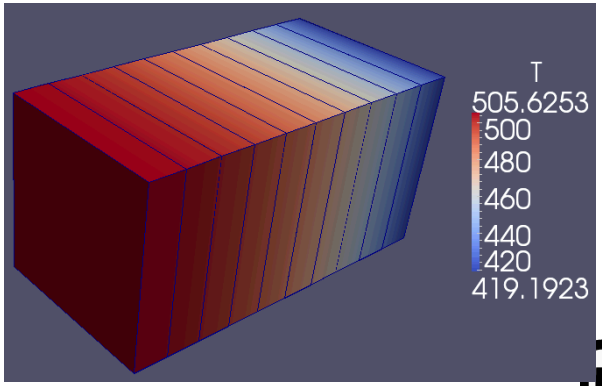
Adaptive time stepping algorithms (manuf. soln.)



Coupled heat transfer and enclosure radiation (analytic)



Chemistry – species evolution and heating



Sandia National Laboratories

Summary & Future Work

- Code verification
  - can be made more cost-effective
  - while still maintaining high quality testing
- Feature coverage analysis
  - enables assessment of impact of verification
  - provides continual feedback between users and code development team
- Future work
  - Lower the barriers to mesh refinement studies (Solution Verification)
  - Link auto-documentation of VERTS to FCT output