PECOS

Predictive Engineering and Computational Sciences

Tools and Techniques for Code Verification using Manufactured Solutions

Nicholas Malaya, Roy Stogner, Robert Moser

Center for Predictive Engineering and Computational Sciences (PECOS)
Institute for Computational Engineering and Sciences (ICES)
The University of Texas at Austin

SIAM UQ Conference, Raleigh, North Carolina, April 3rd, 2011

Introduction to the Method of Manufactured Solutions

Verification of Scientific Software

- Verification ensures that the outputs of a computation accurately reflect the solution of the mathematical models.
- Are we solving our equations correctly?

Method of Exact Solutions

- Numerically solve the equations for a solution that can be determined analytically.

Method of Manufactured Solutions

- Often, analytical solutions either:
  - Do not exist
  - Do not fully exercise equations
- Alleviate this using MMS: "create" our own solutions

Generating MMS using Symbolic Packages

MMS Creation Process

- Start by “manufacturing” a suitable closed-form exact solution
- For example, the 10 parameter trigonometric solution of the form: (Roy, 2002)

$$\hat{u}(x,y,z,t) = \hat{u}_0 + \hat{u}_x\, f_s\left(\frac{a_{\hat{u}x}\pi x}{L}\right) + \hat{u}_y\, f_s\left(\frac{a_{\hat{u}y}\pi y}{L}\right) + \\ + \hat{u}_z\, f_s\left(\frac{a_{\hat{u}z}\pi z}{L}\right) + \hat{u}_t\, f_s\left(\frac{a_{\hat{u}_t}\pi t}{L}\right)$$

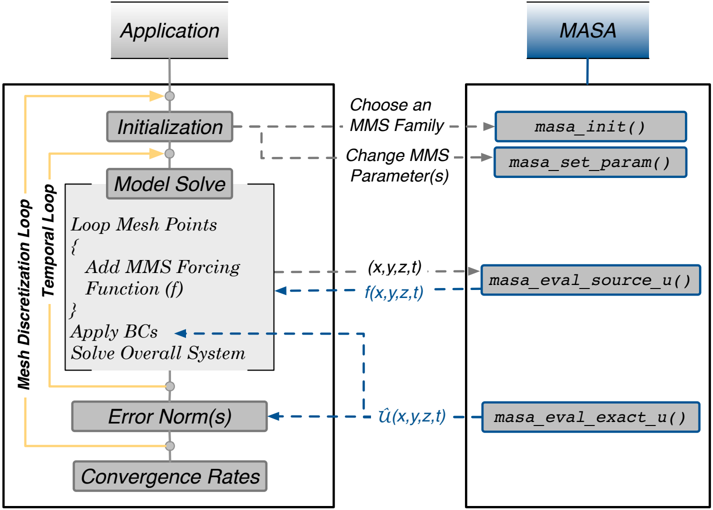
- Apply this solution to equations of interest, solve for source terms (residual)

Accomplished using packages such as: Maple, Mathematica, SymPy, Macsyma

C-code output

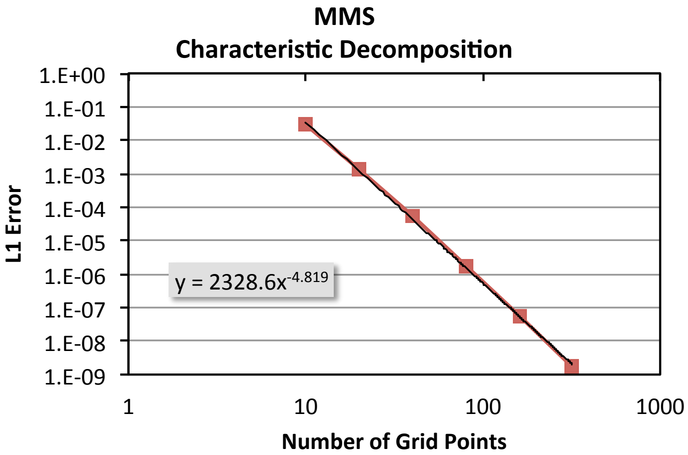
```c
 RHO = rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L) + rho_t * sin(a_rhot * PI * t / L);
  U = u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L) + u_t * cos(a_ut * PI * t / L);
  V = v_0 + v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_t * sin(a_vt * PI * t / L);
  W = w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L) + w_t * cos(a_wt * PI * t / L);
  P = p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L) + p_t * cos(a_pt * PI * t / L);
  M1 = A_mu * pow(P / R / RHO, 0.3e1 / 0.2e1);
  M2 = P / R / RHO + B_mu;
  MU = M1 / M2;
  Q_e = (0.3e1 * a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * U * U / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI
* x / L) + 0.3e1 * a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * V * V / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy *
PI * y / L) - 0.3e1 * a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * W * W / L / 0.2e1 + (0.4e1 * a_ux * a_ux * u_x * sin(a_ux * PI * x / L) + 0.3e1 * a_uy * a_uy * u_y * cos(a_uy * PI * y
/ L) + 0.3e1 * a_uz * a_uz * u_z * cos(a_uz * PI * z / L)) * MU * PI * PI * U * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_vx * a_vx * v_x * cos(a_vx * PI * x / L) + 0.4e1 * a_vy * a_vy * v_y *
sin(a_vy * PI * y / L) + 0.3e1 * a_vz * a_vz * v_z * sin(a_vz * PI * z / L)) * MU * PI * PI * V * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_wx * a_wx * w_x * sin(a_wx * PI * x / L) + 0.3e1 * a_wy *
a_wy * w_y * sin(a_wy * PI * y / L) + 0.4e1 * a_wz * a_wz * w_z * cos(a_wz * PI * z / L)) * MU * PI * PI * W * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y *
cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / (Gamma - 0.1e1) / L - (0.2e1 * a_rhox * a_px * rho_x * p_x * cos(a_rhox * PI * x / L) * sin(a_px * PI * x / L) + 0.2e1
* a_rhoy * a_py * rho_y * p_y * sin(a_rhoy * PI * y / L) * cos(a_py * PI * y / L) + 0.2e1 * a_rhoz * a_pz * rho_z * p_z * cos(a_rhoz * PI * z / L) * sin(a_pz * PI * z / L)) * PI * PI * k *
pow(L, -0.2e1) / R * pow(RHO, -0.2e1) + (U * U + V * V + W * W) * a_rhot * PI * rho_t * cos(a_rhot * PI * t / L) / L / 0.2e1 - a_pt * PI * p_t * sin(a_pt * PI * t / L) / (Gamma - 0.1e1) / L -
(a_uy * u_y * sin(a_uy * PI * y / L) + a_vx * v_x * sin(a_vx * PI * x / L)) * PI * RHO * U * V / L - (a_uz * u_z * sin(a_uz * PI * z / L) - a_wx * w_x * cos(a_wx * PI * x / L)) * PI * RHO * U
* W / L + (a_vz * v_z * cos(a_vz * PI * z / L) + a_wy * w_y * cos(a_wy * PI * y / L)) * PI * RHO * V * W / L + (a_px * a_px * p_x * cos(a_px * PI * x / L) + a_py * a_py * p_y * sin(a_py * PI
* y / L) + a_pz * a_pz * p_z * cos(a_pz * PI * z / L)) * PI * PI * k * pow(L, -0.2e1) / R / RHO - (0.4e1 * a_ux * a_ux * u_x * u_x * pow(cos(a_ux * PI * x / L), 0.2e1) - 0.4e1 * a_ux * a_vy *
u_x * v_y * cos(a_ux * PI * x / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_ux * a_wz * u_x * w_z * cos(a_ux * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_uy * a_uy * u_y * u_y *
pow(sin(a_uy * PI * y / L), 0.2e1) + 0.6e1 * a_uy * a_vx * u_y * v_x * sin(a_uy * PI * y / L) * sin(a_vx * PI * x / L) + 0.3e1 * a_uz * a_uz * u_z * u_z * pow(sin(a_uz * PI * z / L), 0.2e1) -
0.6e1 * a_uz * a_wx * u_z * w_x * sin(a_uz * PI * z / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_vx * a_vx * v_x * v_x * pow(sin(a_vx * PI * x / L), 0.2e1) + 0.4e1 * a_vy * a_vy * v_y * v_y *
pow(cos(a_vy * PI * y / L), 0.2e1) + 0.4e1 * a_vy * a_wz * v_y * w_z * cos(a_vy * PI * y / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_vz * a_vz * v_z * v_z * pow(cos(a_vz * PI * z / L), 0.2e1) +
0.6e1 * a_vz * a_wy * v_z * w_y * cos(a_vz * PI * z / L) * cos(a_wy * PI * y / L) + 0.3e1 * a_wx * a_wx * w_x * w_x * pow(cos(a_wx * PI * x / L), 0.2e1) + 0.3e1 * a_wy * a_wy * w_y * w_y *
pow(cos(a_wy * PI * y / L), 0.2e1) + 0.4e1 * a_wz * a_wz * w_z * w_z * pow(sin(a_wz * PI * z / L), 0.2e1)) * MU * PI * PI * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) +
a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / L - a_ut * PI * u_t * RHO * U * sin(a_ut * PI * t / L) / L + a_vt * PI * v_t * RHO * V * cos(a_vt * PI *
t / L) / L - a_wt * PI * w_t * RHO * W * sin(a_wt * PI * t / L) / L + (U * U + V * V + W * W) * a_rhox * PI * rho_x * U * cos(a_rhox * PI * x / L) / L / 0.2e1 - (U * U + V * V + W * W) *
a_rhoy * PI * rho_y * V * sin(a_rhoy * PI * y / L) / L / 0.2e1 + (U * U + V * V + W * W) * a_rhoz * PI * rho_z * W * cos(a_rhoz * PI * z / L) / L / 0.2e1 - (a_rhox * a_rhox * rho_x *
sin(a_rhox * PI * x / L) + a_rhoy * a_rhoy * rho_y * cos(a_rhoy * PI * y / L) + a_rhoz * a_rhoz * rho_z * sin(a_rhoz * PI * z / L)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.2e1) -
(0.2e1 * a_rhox * a_rhox * rho_x * rho_x * pow(cos(a_rhox * PI * x / L), 0.2e1) + 0.2e1 * a_rhoy * a_rhoy * rho_y * rho_y * pow(sin(a_rhoy * PI * y / L), 0.2e1) + 0.2e1 * a_rhoz * a_rhoz *
rho_z * rho_z * pow(cos(a_rhoz * PI * z / L), 0.2e1)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.3e1) - (0.3e1 * a_rhox * a_uz * rho_x * u_z * cos(a_rhox * PI * x / L) * sin(a_uz *
PI * z / L) - 0.3e1 * a_rhox * a_wx * rho_x * w_x * cos(a_rhox * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_rhoy * a_vz * rho_y * v_z * sin(a_rhoy * PI * y / L) * cos(a_vz * PI * z / L)
+ 0.3e1 * a_rhoy * a_wy * rho_y * w_y * sin(a_rhoy * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_rhoz * a_ux * rho_z * u_x * cos(a_rhoz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 *
a_rhoz * a_vy * rho_z * v_y * cos(a_rhoz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_rhoz * a_wz * rho_z * w_z * cos(a_rhoz * PI * z / L) * sin(a_wz * PI * z / L)) * MU * (0.3e1 * B_mu
* R * RHO + P) * PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_pz * PI * p_z * Gamma * W * sin(a_pz * PI * z / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_px * a_uy * p_x *
u_y * sin(a_px * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_px * a_vx * p_x * v_x * sin(a_px * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_py * a_ux * p_y * u_x * cos(a_py * PI * y
/ L) * cos(a_ux * PI * x / L) + 0.4e1 * a_py * a_vy * p_y * v_y * cos(a_py * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_py * a_wz * p_y * w_z * cos(a_py * PI * y / L) * sin(a_wz * PI *
z / L) - 0.3e1 * a_pz * a_vz * p_z * v_z * sin(a_pz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_pz * a_wy * p_z * w_y * sin(a_pz * PI * z / L) * cos(a_wy * PI * y / L)) * (0.3e1 * B_mu
* R * RHO + P) * MU * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 + a_py * PI * p_y * Gamma * V * cos(a_py * PI * y / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_rhox * a_uy *
rho_x * u_y * cos(a_rhox * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhox * a_vx * rho_x * v_x * cos(a_rhox * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_rhoy * a_ux * rho_y * u_x
* sin(a_rhoy * PI * y / L) * cos(a_ux * PI * x / L) + 0.4e1 * a_rhoy * a_vy * rho_y * v_y * sin(a_rhoy * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhoy * a_wz * rho_y * w_z *
sin(a_rhoy * PI * y / L) * sin(a_wz * PI * z / L) - 0.3e1 * a_rhoz * a_vz * rho_z * v_z * cos(a_rhoz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_rhoz * a_wy * rho_z * w_y * cos(a_rhoz
* PI * z / L) * cos(a_wy * PI * y / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 + (0.4e1 * a_rhox * a_ux * rho_x * u_x *
cos(a_rhox * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_rhox * a_vy * rho_x * v_y * cos(a_rhox * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhox * a_wz * rho_x * w_z * cos(a_rhox
* PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_rhoy * a_uy * rho_y * u_y * sin(a_rhoy * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhoy * a_vx * rho_y * v_x * sin(a_rhoy * PI * y /
L) * sin(a_vx * PI * x / L) - 0.3e1 * a_rhoz * a_uz * rho_z * u_z * cos(a_rhoz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_rhoz * a_wx * rho_z * w_x * cos(a_rhoz * PI * z / L) *
cos(a_wx * PI * x / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_px * PI * p_x * Gamma * U * sin(a_px * PI * x / L) / (Gamma
- 0.1e1) / L + (0.4e1 * a_px * a_ux * p_x * u_x * sin(a_px * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_px * a_vy * p_x * v_y * sin(a_px * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 *
a_px * a_wz * p_x * w_z * sin(a_px * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_py * a_uy * p_y * u_y * cos(a_py * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_py * a_vx * p_y * v_x
* cos(a_py * PI * y / L) * sin(a_vx * PI * x / L) - 0.3e1 * a_pz * a_uz * p_z * u_z * sin(a_pz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_pz * a_wx * p_z * w_x * sin(a_pz * PI * z /
L) * cos(a_wx * PI * x / L)) * (0.3e1 * B_mu * R * RHO + P) * MU * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 - (0.3e1 * a_px * a_uz * p_x * u_z * sin(a_px * PI * x / L)
* sin(a_uz * PI * z / L) - 0.3e1 * a_px * a_wx * p_x * w_x * sin(a_px * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_py * a_vz * p_y * v_z * cos(a_py * PI * y / L) * cos(a_vz * PI * z /
L) + 0.3e1 * a_py * a_wy * p_y * w_y * cos(a_py * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_pz * a_ux * p_z * u_x * sin(a_pz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 * a_pz *
a_vy * p_z * v_y * sin(a_pz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_pz * a_wz * p_z * w_z * sin(a_pz * PI * z / L) * sin(a_wz * PI * z / L)) * (0.3e1 * B_mu * R * RHO + P) * MU *
PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1;
```

Generic Verification Approach Using MMS

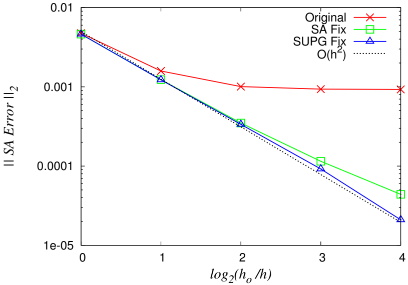


Example: Euler Equations

5th Order Weno Scheme



Detecting Bugs



Verification of FIN-S

- FANS, Spalart-Allmaras
- Model Derivative:

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

- In code:

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

Why is this not more commonly done!?!

Manufactured Analytical Solutions Abstractions Library

Goal: Provide a repository and standardized interface for MMS usage

High Priority:

- Extreme fidelity to generated MMS
- Portability
- Traceability
- Extensible

Low Priority:

- Speed/Performance

Verifying the "Verifier"

Precision is not negotiable: users must trust output!

MASA Testing

- Error target $<$ 1e-15
  - Absolute error on local machines
  - Relative error (other)
  - On all supported compiler sets
- -O0 not sufficient
  - -fp-model precise (Intel)
  - -fno-unsafe-math-optimizations (GNU)
  - -Kieee -Mnofpapprox (PGI)
- "make check"
  - Run by Buildbot every two hours

```
[nick@magus trunk]$ make check
-----------------------------------------
Initializing MASA Tests
-----------------------------------------
PASS: init.sh
PASS: misc
PASS: fail_cond
PASS: catch_exception
PASS: register
PASS: poly
PASS: uninit
PASS: vec
PASS: purge
PASS: heat_const_steady
PASS: euler1d

.           .
.           .
.           .

-----------------------------------------
Finalizing MASA Tests
-----------------------------------------
===================
All 62 tests passed
===================
```

Portability

Software Environment

- Built with: Autotools, C++
- Supports Intel, GNU, Portland Group compilers
- C/C++ interfaces
- Fortran interfaces
  - iso_c_bindings
  - Fortran 2003 Standard

Testing

- SVN: version control
- Buildbot: automated testing
  - Multiple Platforms
- GCOV: line coverage
  - 15,826 lines of code
  - 13,195 lines of testing
  - 98%+ line coverage

LCOV - code coverage report

				Hit	Total	Coverage
Current view:	top level - build/src					
Test:	MASA		Lines:	3077	3132	98.2 %
Date:	2011-04-27		Functions:	946	1277	74.1 %

Filename	Line Coverage		Functions	
ablation.cpp	100.0 %	167 / 167	94.4 %	34 / 36
axi_cns.cpp	100.0 %	73 / 73	58.3 %	14 / 24
axi_euler.cpp	100.0 %	71 / 71	58.3 %	14 / 24
cmasa.cpp	100.0 %	98 / 98	100.0 %	71 / 71
cns.cpp	100.0 %	328 / 328	54.4 %	37 / 68
cp_normal.cpp	100.0 %	68 / 68	87.5 %	21 / 24
euler.cpp	100.0 %	426 / 426	54.3 %	50 / 92
euler_chem.cpp	100.0 %	118 / 118	57.7 %	15 / 26
euler_transient.cpp	100.0 %	72 / 72	60.0 %	12 / 20
fans_sa.cpp	100.0 %	402 / 402	54.3 %	38 / 70
heat.cpp	100.0 %	318 / 318	69.2 %	72 / 104
masa.f90	83.3 %	10 / 12	80.0 %	4 / 5
masa_class.cpp	94.4 %	184 / 195	70.0 %	35 / 50
masa_core.cpp	98.3 %	340 / 346	91.4 %	222 / 243
masa_internal.h	86.7 %	111 / 128	69.1 %	221 / 320
masa_map.cpp	100.0 %	25 / 25	100.0 %	6 / 6
radiation.cpp	88.8 %	71 / 80	62.5 %	10 / 16
rans_sa.cpp	100.0 %	93 / 93	96.3 %	52 / 54
smasa.h	100.0 %	1 / 1	50.0 %	2 / 4
sod.cpp	91.0 %	101 / 111	80.0 %	16 / 20

Generated by: LCOV version 1.9

Traceability

Doxygen provides code and model documentation

where $\phi = \rho, u, v, w$ or p, and $f_s(\cdot)$ functions denote either sine or cosine function. Note that in this case, ϕ_x, ϕ_y and ϕ_z are constants and the subscripts do not denote differentiation.

Although **?** provide the constants used in the manufactured solutions for the 2D supersonic and subsonic cases for Euler and Navier-Stokes equations, only the source term for the 2D mass conservation equation (3.20) is presented.

Source terms for mass conservation (Q_ρ), momentum (Q_u, Q_v and Q_w) and total energy (Q_{e_t}) equations are obtained by symbolic manipulations of compressible steady Euler equations above using Maple 13 (**?**) and are presented in the following sections for the one, two and three-dimensional cases.

3.2.2.1 1D Steady Euler

The manufactured analytical solutions (3.52) for each one of the variables in one-dimensional case of Euler equations are:

$$\begin{aligned} \rho(x) &= \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) \\ u(x) &= u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) \\ p(x) &= p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) \end{aligned} \quad (3.26)$$

The MMS applied to Euler equations consists in modifying the 1D Euler equations (3.20) – (3.22) by adding a source term to the right-hand side of each equation:

$$\begin{aligned} \frac{\partial(\rho u)}{\partial x} &= Q_\rho \\ \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(p)}{\partial x} &= Q_u \\ \frac{\partial(\rho u e_t)}{\partial x} + \frac{\partial(pu)}{\partial x} &= Q_{e_t} \end{aligned} \quad (3.27)$$

so the modified set of equations (3.27) conveniently has the analytical solution given in Equation (3.53).

Source terms Q_ρ, Q_u and Q_{e_t} are obtained by symbolic manipulations of equations above using Maple and are presented in the following sections. The following auxiliary variables have been included in order to improve readability and computational efficiency:

$$\begin{aligned} \mathtt{Rho}_1 &= \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) \\ U_1 &= u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) \\ \mathtt{P}_1 &= p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) \end{aligned}$$

where the subscripts refer to the 1D case.

The mass conservation equation written as an operator is:

$$\mathcal{L} = \frac{\partial(\rho u)}{\partial x}$$

k	u_0	u_x	u_y	u_z	v_0	L
v_0	v_x	v_y	v_z	w_0	w_x	w_y
rho_0	rho_x	rho_y	rho_z	p_0	p_y	p_z
a_px	a_py	a_pz	a_rhox	a_rhoy	a_rhoz	a_ux
a_uy	a_uz	a_vx	a_vy	a_vz	a_wx	a_wy
a_wz	mu	Gamma				

Table 3.6: Parameters used by the 3D Steady Euler

- masa_eval_2d_exact_u()
- masa_eval_2d_exact_v()
- masa_eval_2d_exact_p()
- masa_eval_2d_exact_rho()
- masa_eval_2d_grad_u()
- masa_eval_2d_grad_v()
- masa_eval_2d_grad_p()
- masa_eval_2d_grad_rho()

3.2.3.3 3D Steady Euler

Initialization:

- euler_3d

Functions:

- masa_init()
- masa_eval_3d_source_rho_u()
- masa_eval_3d_source_rho_v()
- masa_eval_3d_source_rho_w()
- masa_eval_3d_source_rho_e()
- masa_eval_3d_source_rho()
- masa_eval_3d_exact_u()
- masa_eval_3d_exact_v()
- masa_eval_3d_exact_w()
- masa_eval_3d_exact_p()

Available Solutions in MASA 0.40

Equations	Dimensions	Time
Euler	1,2,3, axi	Transient, Steady
Non linear heat conduction	1,2,3	Transient, Steady
Navier-Stokes	1,2,3, axi	Transient, Steady
N-S + Sutherland	3	Transient, Steady
N-S + ablation	1	Transient, Steady
Burgers	2	Transient, Steady
Sod Shock Tube	1	Transient
Euler + chemistry	1	Steady
RANS: Spalart-Allmaras	1	Steady
FANS: SA	2	Steady
FANS: SA + wall	2	Steady
Radiation	1	Steady
SMASA: Gaussian	1	Steady

Future Solution Development

Single Physics

- Additional RANS models (k-ω, k-ϵ, etc.)
- Shocks

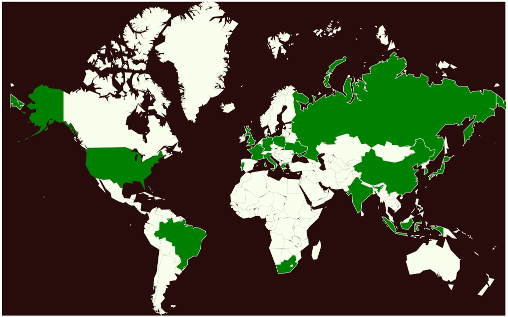
Multiphysics

- Turbulence with chemistry
- Flow with improved transport

Importing New Solutions in MASA 0.40

- Model document detailing analytical solution and source terms
- Latex documents can be loaded directly into MASA documentation
- Source and analytical terms in C/C++/Fortran90
- Willingness to share

Public Library



Current Release

- MASA 0.40.2
- `https://red.ices.utexas.edu/projects/software`
- Open source, LGPL V2.1, free

Maple MMS: 3D Navier-Stokes Energy Term

$$
\begin{aligned}
Qe = &- \frac{a_{px}\pi p_x}{L}\frac{\gamma}{\gamma-1}\sin\left(\frac{a_{px}\pi x}{L}\right)\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right] + \\
&+ \frac{a_{py}\pi p_y}{L}\frac{\gamma}{\gamma-1}\cos\left(\frac{a_{py}\pi y}{L}\right)\left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right] + \\
&- \frac{a_{pz}\pi p_z}{L}\frac{\gamma}{\gamma-1}\sin\left(\frac{a_{pz}\pi z}{L}\right)\left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right] + \\
&+ \frac{a_{\rho x}\pi \rho_x}{2L}\cos\left(\frac{a_{\rho x}\pi x}{L}\right)\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\left(\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2 + \right. \\
&\quad \left. + \left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2 + \left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right) + \\
&- \frac{a_{\rho y}\pi \rho_y}{2L}\sin\left(\frac{a_{\rho y}\pi y}{L}\right)\left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\left(\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2 + \right. \\
&\quad \left. + \left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2 + \left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right) + \\
&+ \frac{a_{\rho z}\pi \rho_z}{2L}\cos\left(\frac{a_{\rho z}\pi z}{L}\right)\left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\left(\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2 + \right. \\
&\quad \left. + \left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2 + \left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right) + \\
&+ \frac{a_{ux}\pi u_x}{2L}\cos\left(\frac{a_{ux}\pi x}{L}\right)\left\{\left(\left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2 + \left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2 + \right.\right. \\
&\quad \left. + 3\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2\right)\left[\rho_0 + \rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right] + \\
&\quad \left. + \left[p_0 + p_x\cos\left(\frac{a_{px}\pi x}{L}\right) + p_y\sin\left(\frac{a_{py}\pi y}{L}\right) + p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]\frac{2\gamma}{(\gamma-1)}\right\} + \\
&- \frac{a_{uy}\pi u_y}{L}\sin\left(\frac{a_{uy}\pi y}{L}\right)\left[v_0 + v_x\cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y\sin\left(\frac{a_{vy}\pi y}{L}\right) + v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\left[\rho_0 + \rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot \\
&\quad \cdot\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right] + \\
&- \frac{a_{uz}\pi u_z}{L}\sin\left(\frac{a_{uz}\pi z}{L}\right)\left[w_0 + w_x\sin\left(\frac{a_{wx}\pi x}{L}\right) + w_y\sin\left(\frac{a_{wy}\pi y}{L}\right) + w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\left[\rho_0 + \rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot \\
&\quad \cdot\left[u_0 + u_x\sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y\cos\left(\frac{a_{uy}\pi y}{L}\right) + u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right] +
\end{aligned}
$$

But wait, there's more!

$$
-\frac{a_{\rho x}^2\pi^2 k\rho_x\left[2\cos\left(\frac{a_{\rho x}\pi x}{L}\right)^2\rho_x+\sin\left(\frac{a_{\rho x}\pi x}{L}\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\right]\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]}{RL^2\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^3}+
$$

$$
-\frac{a_{\rho y}^2\pi^2 k\rho_y\left[2\sin\left(\frac{a_{\rho y}\pi y}{L}\right)^2\rho_y+\cos\left(\frac{a_{\rho y}\pi y}{L}\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\right]\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]}{RL^2\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^3}+
$$

$$
-\frac{a_{\rho z}^2\pi^2 k\rho_z\left[2\cos\left(\frac{a_{\rho z}\pi z}{L}\right)^2\rho_z+\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\right]\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]}{RL^2\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^3}+
$$

$$
-\frac{4}{3}\frac{a_{ux}^2\pi^2\mu u_x}{L^2}\left[\cos\left(\frac{a_{ux}\pi x}{L}\right)^2 u_x-\sin\left(\frac{a_{ux}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{uy}^2\pi^2\mu u_y}{L^2}\left[\sin\left(\frac{a_{uy}\pi y}{L}\right)^2 u_y-\cos\left(\frac{a_{uy}\pi y}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{uz}^2\pi^2\mu u_z}{L^2}\left[\sin\left(\frac{a_{uz}\pi z}{L}\right)^2 u_z-\cos\left(\frac{a_{uz}\pi z}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{vx}^2\pi^2\mu v_x}{L^2}\left[\sin\left(\frac{a_{vx}\pi x}{L}\right)^2 v_x-\cos\left(\frac{a_{vx}\pi x}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{4}{3}\frac{a_{vy}^2\pi^2\mu v_y}{L^2}\left[\cos\left(\frac{a_{vy}\pi y}{L}\right)^2 v_y-\sin\left(\frac{a_{vy}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{vz}^2\pi^2\mu v_z}{L^2}\left[\cos\left(\frac{a_{vz}\pi z}{L}\right)^2 v_z-\sin\left(\frac{a_{vz}\pi z}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{wx}^2\pi^2\mu w_x}{L^2}\left[\cos\left(\frac{a_{wx}\pi x}{L}\right)^2 w_x-\sin\left(\frac{a_{wx}\pi x}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{wy}^2\pi^2\mu w_y}{L^2}\left[\cos\left(\frac{a_{wy}\pi y}{L}\right)^2 w_y-\sin\left(\frac{a_{wy}\pi y}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{4}{3}\frac{a_{wz}^2\pi^2\mu w_z}{L^2}\left[\sin\left(\frac{a_{wz}\pi z}{L}\right)^2 w_z-\cos\left(\frac{a_{wz}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\right]+
$$

$$
-\frac{a_{vx}\pi v_x}{L}\sin\left(\frac{a_{vx}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot
$$
$$
\cdot\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]+
$$

$$
+\frac{a_{vy}\pi v_y}{2L}\cos\left(\frac{a_{vy}\pi y}{L}\right)\left\{\left(\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2+\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\right.\right.
$$
$$
\left.+3\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]+
$$
$$
\left.+\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]\frac{2\gamma}{\gamma-1}\right\}+
$$

$$
+\frac{a_{vz}\pi v_z}{L}\cos\left(\frac{a_{vz}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot
$$
$$
\cdot\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]+
$$

$$
+\frac{a_{wx}\pi w_x}{L}\cos\left(\frac{a_{wx}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot
$$
$$
\cdot\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]+
$$

$$
+\frac{a_{wy}\pi w_y}{L}\cos\left(\frac{a_{wy}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot
$$
$$
\cdot\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]+
$$

$$
-\frac{a_{wz}\pi w_z}{2L}\sin\left(\frac{a_{wz}\pi z}{L}\right)\left\{\left(3\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2+\right.\right.
$$
$$
\left.+\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]+
$$
$$
\left.+\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]\frac{2\gamma}{(\gamma-1)}\right\}+
$$

$$
+\frac{a_{px}^2\pi^2 kp_x\cos\left(\frac{a_{px}\pi x}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]}+
$$

$$
+\frac{a_{py}^2\pi^2 kp_y\sin\left(\frac{a_{py}\pi y}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]}+
$$

$$
+\frac{a_{pz}^2\pi^2 kp_z\cos\left(\frac{a_{pz}\pi z}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]}+
$$

$$
-\frac{2a_{px}a_{\rho x}\pi^2 kp_x\rho_x\cos\left(\frac{a_{\rho x}\pi x}{L}\right)\sin\left(\frac{a_{px}\pi x}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^2}+
$$

$$
-\frac{2a_{py}a_{\rho y}\pi^2 kp_y\rho_y\sin\left(\frac{a_{\rho y}\pi y}{L}\right)\cos\left(\frac{a_{py}\pi y}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^2}+
$$

$$
-\frac{2a_{pz}a_{\rho z}\pi^2 kp_z\rho_z\cos\left(\frac{a_{\rho z}\pi z}{L}\right)\sin\left(\frac{a_{pz}\pi z}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]^2}+
$$

$$
+\frac{4}{3}\frac{a_{ux}a_{vy}\pi^2\mu u_x v_y}{L^2}\cos\left(\frac{a_{ux}\pi x}{L}\right)\cos\left(\frac{a_{vy}\pi y}{L}\right)+
$$

$$
-\frac{4}{3}\frac{a_{ux}a_{wz}\pi^2\mu u_x w_z}{L^2}\cos\left(\frac{a_{ux}\pi x}{L}\right)\sin\left(\frac{a_{wz}\pi z}{L}\right)+
$$

$$
-\frac{2a_{uy}a_{vx}\pi^2\mu u_y v_x}{L^2}\sin\left(\frac{a_{uy}\pi y}{L}\right)\sin\left(\frac{a_{vx}\pi x}{L}\right)+
$$

$$
+\frac{2a_{uz}a_{wx}\pi^2\mu u_z w_x}{L^2}\cos\left(\frac{a_{wx}\pi x}{L}\right)\sin\left(\frac{a_{uz}\pi z}{L}\right)+
$$

$$
-\frac{4}{3}\frac{a_{vy}a_{wz}\pi^2\mu v_y w_z}{L^2}\cos\left(\frac{a_{vy}\pi y}{L}\right)\sin\left(\frac{a_{wz}\pi z}{L}\right)+
$$

$$
-\frac{2a_{vz}a_{wy}\pi^2\mu v_z w_y}{L^2}\cos\left(\frac{a_{vz}\pi z}{L}\right)\cos\left(\frac{a_{wy}\pi y}{L}\right).
$$

Combinatorial Explosion

Explosion in source term size

- Complexity increases with more sophisticated mathematical models
  - Sutherland viscosity model has over 1,612,000 characters
- Large memory requirements (128 GB not sufficient for Sutherland)
- Computational intensity
- segfaults

Hierarchic MMS

- Decompose each equation into sub-terms
- Each term is operated on individually by the manufactured solution
- Resulting expressions are re-combined to regain source term

The Hierarchic MMS

Consider the full 3D Navier-Stokes energy equation:

$$\mathcal{L} = \frac{\partial(\rho e_t)}{\partial t} + \nabla \cdot (\rho \mathbf{u} e_t) + \nabla \cdot \mathbf{q} + \nabla \cdot (p\mathbf{u}) - \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u})$$

Decompose:

$$\begin{aligned}
\mathcal{L}_1 &= \frac{\partial(\rho e_t)}{\partial t} \\
\mathcal{L}_2 &= \nabla \cdot (\rho \mathbf{u} e_t) \\
\mathcal{L}_3 &= \nabla \cdot \mathbf{q} \\
\mathcal{L}_4 &= \nabla \cdot (p\mathbf{u}) \\
\mathcal{L}_5 &= -\nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u})
\end{aligned}$$

Hierarchic MMS extensions:

- Expand each component of divergence
- Transient and steady cases ($\mathcal{L}_1$ = 0)
- Sutherland model only requires altering $\mathcal{L}_5$
- Navier-Stokes $\rightarrow$ Euler and additional terms
- Can assist in debugging

C-code output

```
Q_e = -0.2e1 * cos(a_rhox * PI * x / L) * rho_x * sin(a_px * PI * x / L) * k * p_x * a_px * a_rhox * PI * PI * pow(L, -0.2e1) / R * pow(rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y *
cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.2e1) - 0.2e1 * sin(a_rhoy * PI * y / L) * rho_y * cos(a_py * PI * y / L) * k * p_y * a_py * a_rhoy * PI * PI * pow(L, -0.2e1) /
R * pow(rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.2e1) - 0.2e1 * cos(a_rhoz * PI * z / L) * rho_z * sin(a_pz * PI * z
/ L) * k * p_z * a_pz * a_rhoz * PI * PI * pow(L, -0.2e1) / R * pow(rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.2e1) -
(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z *
sin(a_rhoz * PI * z / L)) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L)) * u_y * sin(a_uy * PI * y / L) * a_uy * PI / L + (w_0 + w_x *
sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI
* z / L)) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L)) * w_x * cos(a_wx * PI * x / L) * a_wx * PI / L - (w_0 + w_x * sin(a_wx * PI * x
/ L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) * (u_0 +
u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L)) * u_z * sin(a_uz * PI * z / L) * a_uz * PI / L + sin(a_py * PI * y / L) * k * p_y * a_py * a_py *
PI * PI * pow(L, -0.2e1) / R / (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) + cos(a_px * PI * x / L) * k * p_x * a_px *
a_px * PI * PI * pow(L, -0.2e1) / R / (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) + (w_0 + w_x * sin(a_wx * PI * x / L) +
w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) * (v_x *
cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0) * v_z * cos(a_vz * PI * z / L) * a_vz * PI / L + cos(a_pz * PI * z / L) * k * p_z * a_pz * a_pz *
PI * PI * pow(L, -0.2e1) / R / (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) - (0.2e1 * pow(cos(a_rhox * PI * x / L), 0.2e1)
* rho_x + sin(a_rhox * PI * x / L) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L))) * (p_0 + p_x * cos(a_px * PI * x / L) +
p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L)) * k * rho_x * a_rhox * a_rhox * PI * PI * pow(L, -0.2e1) / R * pow(rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy *
PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.3e1) - (0.2e1 * pow(sin(a_rhoy * PI * y / L), 0.2e1) * rho_y + cos(a_rhoy * PI * y / L) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y
* cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L))) * (p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L)) * k * rho_y * a_rhoy *
a_rhoy * PI * PI * pow(L, -0.2e1) / R * pow(rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.3e1) - (0.2e1 * pow(cos(a_rhoz *
PI * z / L), 0.2e1) * rho_z + sin(a_rhoz * PI * z / L) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L))) * (p_0 + p_x *
cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L)) * k * rho_z * a_rhoz * a_rhoz * PI * PI * pow(L, -0.2e1) / R * pow(rho_0 + rho_x * sin(a_rhox * PI * x /
L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L), -0.3e1) + 0.4e1 / 0.3e1 * mu * u_x * v_y * cos(a_ux * PI * x / L) * cos(a_vy * PI * y / L) * a_ux * a_vy * PI * PI *
pow(L, -0.2e1) - 0.4e1 / 0.3e1 * mu * u_x * w_z * cos(a_ux * PI * x / L) * sin(a_wz * PI * z / L) * a_ux * a_wz * PI * PI * pow(L, -0.2e1) - 0.2e1 * mu * u_y * v_x * sin(a_uy * PI * y / L) *
sin(a_vx * PI * x / L) * a_uy * a_vx * PI * PI * pow(L, -0.2e1) + 0.2e1 * mu * u_z * w_x * cos(a_wx * PI * x / L) * sin(a_uz * PI * z / L) * a_uz * a_wx * PI * PI * pow(L, -0.2e1) - 0.4e1 /
0.3e1 * mu * v_y * w_z * cos(a_vy * PI * y / L) * sin(a_wz * PI * z / L) * a_vy * a_wz * PI * PI * pow(L, -0.2e1) - 0.2e1 * mu * v_z * w_y * cos(a_vz * PI * z / L) * cos(a_wy * PI * y / L) *
a_vz * a_wy * PI * PI * pow(L, -0.2e1) + (w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y
* cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) * (v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0) * w_y * cos(a_wy * PI * y /
L) * a_wy * PI * (-(0.3e1 * pow(w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L), 0.2e1) + pow(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy *
PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0, 0.2e1) + pow(u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L), 0.2e1)) * (rho_0 + rho_x *
sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) / L / 0.2e1 - Gamma * (p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z
* cos(a_pz * PI * z / L)) / L / (Gamma - 0.1e1)) * w_z * sin(a_wz * PI * z / L) * a_wz * PI / L + (-0.1e1) * (v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI
* z / L) + v_0) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI
* y / L) + u_z * cos(a_uz * PI * z / L)) * v_x * sin(a_vx * PI * x / L) * a_vx * PI * ((pow(u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L),
0.2e1) + pow(w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L), 0.2e1) + 0.3e1 * pow(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L)
+ v_z * sin(a_vz * PI * z / L) + v_0, 0.2e1)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) / L / 0.2e1 + Gamma * (p_0 +
p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L)) / L / (Gamma - 0.1e1)) * v_y * cos(a_vy * PI * y / L) * a_vy * PI / L - Gamma * (w_0 + w_x *
sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * sin(a_pz * PI * z / L) * p_z * a_pz * PI / L / (Gamma - 0.1e1) + 0.1e1 / 0.2e1 * cos(a_rhoz * PI * z /
L) * (w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L)) * (pow(u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z *
cos(a_uz * PI * z / L), 0.2e1) + pow(w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L), 0.2e1) + pow(v_x * cos(a_vx * PI * x / L) + v_y *
sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0, 0.2e1)) * rho_z * a_rhoz * PI * ((pow(w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z
/ L), 0.2e1) + pow(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0, 0.2e1) + 0.3e1 * pow(u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy *
PI * y / L) + u_z * cos(a_uz * PI * z / L), 0.2e1)) * (rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L)) / L / 0.2e1 + Gamma *
(p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L)) / L / (Gamma - 0.1e1)) * u_x * cos(a_ux * PI * x / L) * a_ux * PI / L - Gamma * (u_0 + u_x *
sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L)) * sin(a_px * PI * x / L) * p_x * a_px * PI / L / (Gamma - 0.1e1) + Gamma * (v_x * cos(a_vx * PI * x / L)
+ v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0) * cos(a_py * PI * y / L) * p_y * a_py * PI / L / (Gamma - 0.1e1) + 0.4e1 / 0.3e1 * (-pow(cos(a_ux * PI * x / L), 0.2e1) *
u_x + sin(a_ux * PI * x / L) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L))) * mu * u_x * a_ux * a_ux * PI * PI * pow(L, -0.2e1) +
(-pow(sin(a_uy * PI * y / L), 0.2e1) * u_y + cos(a_uy * PI * y / L) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L))) * mu * u_y * a_uy *
a_uy * PI * PI * pow(L, -0.2e1) + (-pow(sin(a_uz * PI * z / L), 0.2e1) * u_z + cos(a_uz * PI * z / L) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz *
PI * z / L))) * mu * u_z * a_uz * a_uz * PI * PI * pow(L, -0.2e1) - (pow(sin(a_vx * PI * x / L), 0.2e1) * v_x - cos(a_vx * PI * x / L) * (v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI *
y / L) + v_z * sin(a_vz * PI * z / L) + v_0)) * mu * v_x * a_vx * a_vx * PI * PI * pow(L, -0.2e1) - 0.4e1 / 0.3e1 * (pow(cos(a_vy * PI * y / L), 0.2e1) * v_y - sin(a_vy * PI * y / L) * (v_x *
cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0)) * mu * v_y * a_vy * a_vy * PI * PI * pow(L, -0.2e1) - (pow(cos(a_vz * PI * z / L), 0.2e1) * v_z -
sin(a_vz * PI * z / L) * (v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0)) * mu * v_z * a_vz * a_vz * PI * PI * pow(L, -0.2e1) +
(-pow(cos(a_wx * PI * x / L), 0.2e1) * w_x + sin(a_wx * PI * x / L) * (w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L))) * mu * w_x * a_wx *
a_wx * PI * PI * pow(L, -0.2e1) + (-pow(cos(a_wy * PI * y / L), 0.2e1) * w_y + sin(a_wy * PI * y / L) * (w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz *
PI * z / L))) * mu * w_y * a_wy * a_wy * PI * PI * pow(L, -0.2e1) + 0.4e1 / 0.3e1 * (-pow(sin(a_wz * PI * z / L), 0.2e1) * w_z + cos(a_wz * PI * z / L) * (w_0 + w_x * sin(a_wx * PI * x / L) +
w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L))) * mu * w_z * a_wz * a_wz * PI * PI * pow(L, -0.2e1) + cos(a_rhox * PI * x / L) * (u_0 + u_x * sin(a_ux * PI * x / L) + u_y *
cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L)) * (pow(u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L), 0.2e1) + pow(w_0 + w_x *
sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L), 0.2e1) + pow(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) +
v_0, 0.2e1)) * rho_x * a_rhox * PI / L / 0.2e1 - sin(a_rhoy * PI * y / L) * (v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0) * (pow(u_0 + u_x
* sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L), 0.2e1) + pow(w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI *
z / L), 0.2e1) + pow(v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_0, 0.2e1)) * rho_y * a_rhoy * PI / L / 0.2e1;
```

Reduced source term C-output

```
 RHO = rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L) + rho_t * sin(a_rhot * PI * t / L);
  U = u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L) + u_t * cos(a_ut * PI * t / L);
  V = v_0 + v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_t * sin(a_vt * PI * t / L);
  W = w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L) + w_t * cos(a_wt * PI * t / L);
  P = p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L) + p_t * cos(a_pt * PI * t / L);
  M1 = A_mu * pow(P / R / RHO, 0.3e1 / 0.2e1);
  M2 = P / R / RHO + B_mu;
  MU = M1 / M2;
  Q_e = (0.3e1 * a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * U * U / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI
* x / L) + 0.3e1 * a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * V * V / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy *
PI * y / L) - 0.3e1 * a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * W * W / L / 0.2e1 + (0.4e1 * a_ux * a_ux * u_x * sin(a_ux * PI * x / L) + 0.3e1 * a_uy * a_uy * u_y * cos(a_uy * PI * y
/ L) + 0.3e1 * a_uz * a_uz * u_z * cos(a_uz * PI * z / L)) * MU * PI * PI * U * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_vx * a_vx * v_x * cos(a_vx * PI * x / L) + 0.4e1 * a_vy * a_vy * v_y *
sin(a_vy * PI * y / L) + 0.3e1 * a_vz * a_vz * v_z * sin(a_vz * PI * z / L)) * MU * PI * PI * V * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_wx * a_wx * w_x * sin(a_wx * PI * x / L) + 0.3e1 * a_wy *
a_wy * w_y * sin(a_wy * PI * y / L) + 0.4e1 * a_wz * a_wz * w_z * cos(a_wz * PI * z / L)) * MU * PI * PI * W * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y *
cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / (Gamma - 0.1e1) / L - (0.2e1 * a_rhox * a_px * rho_x * p_x * cos(a_rhox * PI * x / L) * sin(a_px * PI * x / L) + 0.2e1
* a_rhoy * a_py * rho_y * p_y * sin(a_rhoy * PI * y / L) * cos(a_py * PI * y / L) + 0.2e1 * a_rhoz * a_pz * rho_z * p_z * cos(a_rhoz * PI * z / L) * sin(a_pz * PI * z / L)) * PI * PI * k *
pow(L, -0.2e1) / R * pow(RHO, -0.2e1) + (U * U + V * V + W * W) * a_rhot * PI * rho_t * cos(a_rhot * PI * t / L) / L / 0.2e1 - a_pt * PI * p_t * sin(a_pt * PI * t / L) / (Gamma - 0.1e1) / L -
(a_uy * u_y * sin(a_uy * PI * y / L) + a_vx * v_x * sin(a_vx * PI * x / L)) * PI * RHO * U * V / L - (a_uz * u_z * sin(a_uz * PI * z / L) - a_wx * w_x * cos(a_wx * PI * x / L)) * PI * RHO * U
* W / L + (a_vz * v_z * cos(a_vz * PI * z / L) + a_wy * w_y * cos(a_wy * PI * y / L)) * PI * RHO * V * W / L + (a_px * a_px * p_x * cos(a_px * PI * x / L) + a_py * a_py * p_y * sin(a_py * PI
* y / L) + a_pz * a_pz * p_z * cos(a_pz * PI * z / L)) * PI * PI * k * pow(L, -0.2e1) / R / RHO - (0.4e1 * a_ux * a_ux * u_x * u_x * pow(cos(a_ux * PI * x / L), 0.2e1) - 0.4e1 * a_ux * a_vy *
u_x * v_y * cos(a_ux * PI * x / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_ux * a_wz * u_x * w_z * cos(a_ux * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_uy * a_uy * u_y * u_y *
pow(sin(a_uy * PI * y / L), 0.2e1) + 0.6e1 * a_uy * a_vx * u_y * v_x * sin(a_uy * PI * y / L) * sin(a_vx * PI * x / L) + 0.3e1 * a_uz * a_uz * u_z * u_z * pow(sin(a_uz * PI * z / L), 0.2e1) -
0.6e1 * a_uz * a_wx * u_z * w_x * sin(a_uz * PI * z / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_vx * a_vx * v_x * v_x * pow(sin(a_vx * PI * x / L), 0.2e1) + 0.4e1 * a_vy * a_vy * v_y * v_y *
pow(cos(a_vy * PI * y / L), 0.2e1) + 0.4e1 * a_vy * a_wz * v_y * w_z * cos(a_vy * PI * y / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_vz * a_vz * v_z * v_z * pow(cos(a_vz * PI * z / L), 0.2e1) +
0.6e1 * a_vz * a_wy * v_z * w_y * cos(a_vz * PI * z / L) * cos(a_wy * PI * y / L) + 0.3e1 * a_wx * a_wx * w_x * w_x * pow(cos(a_wx * PI * x / L), 0.2e1) + 0.3e1 * a_wy * a_wy * w_y * w_y *
pow(cos(a_wy * PI * y / L), 0.2e1) + 0.4e1 * a_wz * a_wz * w_z * w_z * pow(sin(a_wz * PI * z / L), 0.2e1)) * MU * PI * PI * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) +
a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / L - a_ut * PI * u_t * RHO * U * sin(a_ut * PI * t / L) / L + a_vt * PI * v_t * RHO * V * cos(a_vt * PI *
t / L) / L - a_wt * PI * w_t * RHO * W * sin(a_wt * PI * t / L) / L + (U * U + V * V + W * W) * a_rhox * PI * rho_x * U * cos(a_rhox * PI * x / L) / L / 0.2e1 - (U * U + V * V + W * W) *
a_rhoy * PI * rho_y * V * sin(a_rhoy * PI * y / L) / L / 0.2e1 + (U * U + V * V + W * W) * a_rhoz * PI * rho_z * W * cos(a_rhoz * PI * z / L) / L / 0.2e1 - (a_rhox * a_rhox * rho_x *
sin(a_rhox * PI * x / L) + a_rhoy * a_rhoy * rho_y * cos(a_rhoy * PI * y / L) + a_rhoz * a_rhoz * rho_z * sin(a_rhoz * PI * z / L)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.2e1) -
(0.2e1 * a_rhox * a_rhox * rho_x * rho_x * pow(cos(a_rhox * PI * x / L), 0.2e1) + 0.2e1 * a_rhoy * a_rhoy * rho_y * rho_y * pow(sin(a_rhoy * PI * y / L), 0.2e1) + 0.2e1 * a_rhoz * a_rhoz *
rho_z * rho_z * pow(cos(a_rhoz * PI * z / L), 0.2e1)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.3e1) - (0.3e1 * a_rhox * a_uz * rho_x * u_z * cos(a_rhox * PI * x / L) * sin(a_uz *
PI * z / L) - 0.3e1 * a_rhox * a_wx * rho_x * w_x * cos(a_rhox * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_rhoy * a_vz * rho_y * v_z * sin(a_rhoy * PI * y / L) * cos(a_vz * PI * z / L)
+ 0.3e1 * a_rhoy * a_wy * rho_y * w_y * sin(a_rhoy * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_rhoz * a_ux * rho_z * u_x * cos(a_rhoz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 *
a_rhoz * a_vy * rho_z * v_y * cos(a_rhoz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_rhoz * a_wz * rho_z * w_z * cos(a_rhoz * PI * z / L) * sin(a_wz * PI * z / L)) * MU * (0.3e1 * B_mu
* R * RHO + P) * PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_pz * PI * p_z * Gamma * W * sin(a_pz * PI * z / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_px * a_uy * p_x *
u_y * sin(a_px * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_px * a_vx * p_x * v_x * sin(a_px * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_py * a_ux * p_y * u_x * cos(a_py * PI * y
/ L) * cos(a_ux * PI * x / L) + 0.4e1 * a_py * a_vy * p_y * v_y * cos(a_py * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_py * a_wz * p_y * w_z * cos(a_py * PI * y / L) * sin(a_wz * PI *
z / L) - 0.3e1 * a_pz * a_vz * p_z * v_z * sin(a_pz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_pz * a_wy * p_z * w_y * sin(a_pz * PI * z / L) * cos(a_wy * PI * y / L)) * (0.3e1 * B_mu
* R * RHO + P) * MU * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 + a_py * PI * p_y * Gamma * V * cos(a_py * PI * y / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_rhox * a_uy *
rho_x * u_y * cos(a_rhox * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhox * a_vx * rho_x * v_x * cos(a_rhox * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_rhoy * a_ux * rho_y * u_x
* sin(a_rhoy * PI * y / L) * cos(a_ux * PI * x / L) + 0.4e1 * a_rhoy * a_vy * rho_y * v_y * sin(a_rhoy * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhoy * a_wz * rho_y * w_z *
sin(a_rhoy * PI * y / L) * sin(a_wz * PI * z / L) - 0.3e1 * a_rhoz * a_vz * rho_z * v_z * cos(a_rhoz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_rhoz * a_wy * rho_z * w_y * cos(a_rhoz
* PI * z / L) * cos(a_wy * PI * y / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 + (0.4e1 * a_rhox * a_ux * rho_x * u_x *
cos(a_rhox * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_rhox * a_vy * rho_x * v_y * cos(a_rhox * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhox * a_wz * rho_x * w_z * cos(a_rhox
* PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_rhoy * a_uy * rho_y * u_y * sin(a_rhoy * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhoy * a_vx * rho_y * v_x * sin(a_rhoy * PI * y /
L) * sin(a_vx * PI * x / L) - 0.3e1 * a_rhoz * a_uz * rho_z * u_z * cos(a_rhoz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_rhoz * a_wx * rho_z * w_x * cos(a_rhoz * PI * z / L) *
cos(a_wx * PI * x / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_px * PI * p_x * Gamma * U * sin(a_px * PI * x / L) / (Gamma
- 0.1e1) / L + (0.4e1 * a_px * a_ux * p_x * u_x * sin(a_px * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_px * a_vy * p_x * v_y * sin(a_px * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 *
a_px * a_wz * p_x * w_z * sin(a_px * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_py * a_uy * p_y * u_y * cos(a_py * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_py * a_vx * p_y * v_x
* cos(a_py * PI * y / L) * sin(a_vx * PI * x / L) - 0.3e1 * a_pz * a_uz * p_z * u_z * sin(a_pz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_pz * a_wx * p_z * w_x * sin(a_pz * PI * z /
L) * cos(a_wx * PI * x / L)) * (0.3e1 * B_mu * R * RHO + P) * MU * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 - (0.3e1 * a_px * a_uz * p_x * u_z * sin(a_px * PI * x / L)
* sin(a_uz * PI * z / L) - 0.3e1 * a_px * a_wx * p_x * w_x * sin(a_px * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_py * a_vz * p_y * v_z * cos(a_py * PI * y / L) * cos(a_vz * PI * z /
L) + 0.3e1 * a_py * a_wy * p_y * w_y * cos(a_py * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_pz * a_ux * p_z * u_x * sin(a_pz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 * a_pz *
a_vy * p_z * v_y * sin(a_pz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_pz * a_wz * p_z * w_z * sin(a_pz * PI * z / L) * sin(a_wz * PI * z / L)) * (0.3e1 * B_mu * R * RHO + P) * MU *
PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1;
```

Shortcomings

Symbolic Shortcomings

- Hierarchic decomposition is an improvement, but is it enough?
  - Even with factorization, source terms still massive
  - Generating manufactured solutions was a full time job at PECOS
- Everything we have discussed has been generated outside of MASA
  - Not thrilled with Maple, Mathematica

Enter Automatic Differentiation

- AD numerically evaluates the derivative of a function
  - applies chain rule repeatedly
- Superior error characteristics (round-off)
- Slow (but we barely care)
- Several libraries: NAG, Sacado, etc.

"Dual Numbers" - [Clifford 1873],[Study 1891]

- A new element ϵ
- Closed under addition and multiplication:

$$\left\{ \sum_{i=0}^{m} a_i \epsilon^i : a_i \in \mathbb{R}, m < \infty \right\}$$

- Take the quotient with $\epsilon^2 \equiv 0$

$$\mathbb{D}[\mathbb{R}] \equiv \{a + b\epsilon : a, b \in \mathbb{R}\}$$

- Used with quaternions to represent rotations and translations
- Arithmetic:

$$\begin{aligned}
(a + b\epsilon) + (c + d\epsilon) &= ((a + c) + (b + d)\epsilon) \\
(a + b\epsilon) - (c + d\epsilon) &= ((a + c) - (b + d)\epsilon) \\
(a + b\epsilon) \times (c + d\epsilon) &= (ac) + ad\epsilon + bc\epsilon + bd\epsilon^2 \\
&= ((ac) + (ad + bc)\epsilon)
\end{aligned}$$

“Hyper-dual Numbers” - [Fike 2009]

- Add two new elements ϵ_1, ϵ_2 to $\mathbb{R}$
- Take the quotient with $\epsilon_1^2 \equiv \epsilon_2^2 \equiv 0$

$$\mathbb{H}[\mathbb{R}] \equiv \{a + b\epsilon_1 + c\epsilon_2 + d\epsilon_1\epsilon_2 : a, b, c, d \in \mathbb{R}\}$$

- Arithmetic:

$$\begin{aligned}
(a + b\epsilon_1 + c\epsilon_2 + d\epsilon_1\epsilon_2) + (e + f\epsilon_1 + g\epsilon_2 + h\epsilon_1\epsilon_2) &= \\
((a+e) + (b+f)\epsilon_1 + (c+g)\epsilon_2 + (d+h)\epsilon_1\epsilon_2) & \\
(a + b\epsilon_1 + c\epsilon_2 + d\epsilon_1\epsilon_2) - (e + f\epsilon_1 + g\epsilon_2 + h\epsilon_1\epsilon_2) &= \\
((a-e) + (b-f)\epsilon_1 + (c-g)\epsilon_2 + (d-h)\epsilon_1\epsilon_2) & \\
(a + b\epsilon_1 + c\epsilon_2 + d\epsilon_1\epsilon_2) \times (e + f\epsilon_1 + g\epsilon_2 + h\epsilon_1\epsilon_2) &= \\
(ae) + (af + be)\epsilon_1 + (ag + ce)\epsilon_2 + (ah + de + bg + cf)\epsilon_1\epsilon_2 &
\end{aligned}$$

"Hyper-Dual Numbers"

- With $\epsilon_1^2 \equiv \epsilon_2^2 \equiv 0 \equiv (\epsilon_1 \epsilon_2)^2 = 0$
- Where $X \equiv x + \epsilon_1 + \epsilon_2$, we find:

$$f(X) = f(x) + f'(x)\epsilon_1 + f'(x)\epsilon_2 + f''(x)\epsilon_1\epsilon_2$$

- The Taylor series truncates exactly at the second-derivative term
- Using hyper-dual numbers results in first- and second-derivative calculations that are exact, regardless of step size
- methods for computing exact higher derivatives can be created by using more non-real parts (ϵ_3, for instance)

Data Type Independence: Generic Programming

- Template arguments can be data types
- Class data members can depend on template argument
- Methods and functions can as well

Example

```
template <typename T>
class NumericVector {
T* data;
};
```

This is how MASA provides arbitrary precision, e.g.
masa_eval_source$<$ double $>$()

Constructing higher rank objects

Templated classes can be instantiated with plain data types

- or with other instantiated templated types
- or with other instantiations of themselves

Example

```
NumericVector<float> floatvec;
NumericVector<complex<double> > complexdoublevec;
NumericVector<NumericVector<float> > floatmatrix;
```

MASA PDE Examples

Manufactured Solution

```
// Arbitrary manufactured solutions
U.template get<0>() = u_0 + u_x * sin(a_ux * PI * x / L) +
                            u_y * cos(a_uy * PI * y / L);

U.template get<1>() = v_0 + v_x * cos(a_vx * PI * x / L) +
                            v_y * sin(a_vy * PI * y / L);

ADScalar RHO = rho_0 + rho_x * sin(a_rhox * PI * x / L) +
                       rho_y * cos(a_rhoy * PI * y / L);

ADScalar P = p_0 + p_x * cos(a_px * PI * x / L) +
                   p_y * sin(a_py * PI * y / L);
```

MASA PDE Examples

Source Terms: Euler

```
// Gas state
ADScalar T = P / RHO / R;
ADScalar E = 1. / (Gamma-1.) * P / RHO;
ADScalar ET = E + .5 * U.dot(U);

// Mass, momentum and energy
Scalar Q_rho = raw_value(divergence(RHO*U));
RawArray Q_rho_u = raw_value(divergence(RHO*U.outerproduct(U)) +
                   P.derivatives());
 Scalar Q_rho_e = raw_value(divergence((RHO*ET+P)*U));
```

MASA PDE Examples

Source Terms: Navier-Stokes

```
// Gas state
ADScalar T = P / RHO / R;
ADScalar E = 1. / (Gamma-1.) * P / RHO;
ADScalar ET = E + .5 * U.dot(U);

// Constitutive laws
Tensor GradU = gradient(U);
Tensor Tau = mu * (GradU + transpose(GradU) -
  2./3. * divergence(U) * RawArray::identity());
FullArray q = -k * T.derivatives();

Scalar Q_rho = raw_value(divergence(RHO*U));
RawArray Q_rho_u = raw_value(divergence(RHO*U.outerproduct(U) - Tau) +
                   P.derivatives());
 Scalar Q_rho_e = raw_value(divergence((RHO*ET+P)*U + q - Tau.dot(U)));
```

Operator-Overloaded Multi-precision

Example

```
template <typename T, typename S>
struct ShadowNumber {
T _val;
S _shadow;
};
```

- Simultaneous calculation with multiple floating point representations, to estimate rounding error.

Example

```
ShadowNumber<float, double>        shadowed_float;
ShadowNumber<double, long double> shadowed_double;
```

Future AD work

Future Work

- Automatic Latex Generation
- Latex Parser – MMS generator
- Complex multiphysics
- Inverse Problems

Stochastic MASA and QUESO Verification

Initial Effort

- Conjugate Prior(s)
  - Posterior is in the same family as the prior
- Initial problem: Gaussian
- QUESO: multilevel Monte Carlo
  - {5k, 50k, 500k} samples
- Tricky: sampling posterior
  - Convergence rates can be bounded

```
5k Posterior Mean         = 2.19617
50k Posterior Mean        = 2.18629
500k Posterior Mean       = 2.1789
SMASA Posterior Mean      = 2.17749

5k Posterior Std.Dev.     = 0.31234
50k Posterior Std.Dev.    = 0.313286
500k Posterior Std.Dev.   = 0.303793
SMASA Posterior Std.Dev.  = 0.301511
```

Future work: expand AD to inverse problems.

Conclusions

Summary

- MMS is not a difficult concept, but can be tricky and time consuming
- Must have a high degree of confidence in your verification suite
- MASA is an open source library designed to:
  - Increase use of existing MMS in the community
  - Provide a standardized interface and toolset to the community
  - Serve as an example of high quality verification software
  - Available at: https://red.ices.utexas.edu/projects/software

Conclusions

Thank you!

Have a well verified day.

nick@ices.utexas.edu