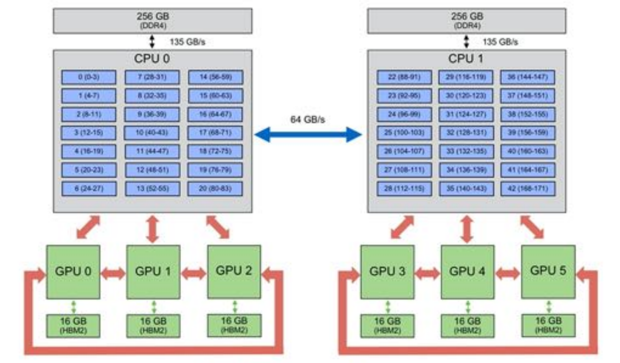
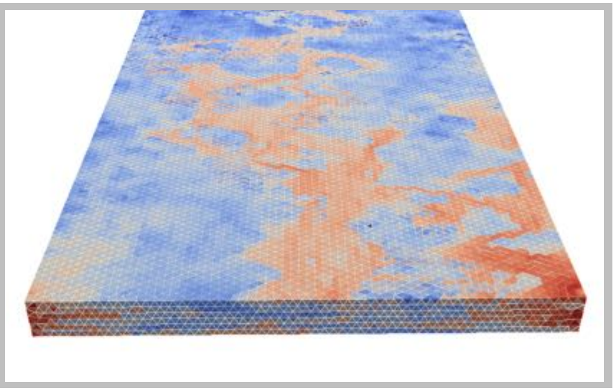
Introduction to Trilinos and MrHyDE

MrHyDE = {M}ulti-{r}esolution {Hy}bridized {D}ifferential {E}quations





Tim Wildey

Optimization and Uncertainty Quantification Department

Center for Computing Research

U.S. DEPARTMENT OF ENERGY | Office of Science

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Any questions from yesterday?

Tutorial Outline

Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization

Tutorial Outline

Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

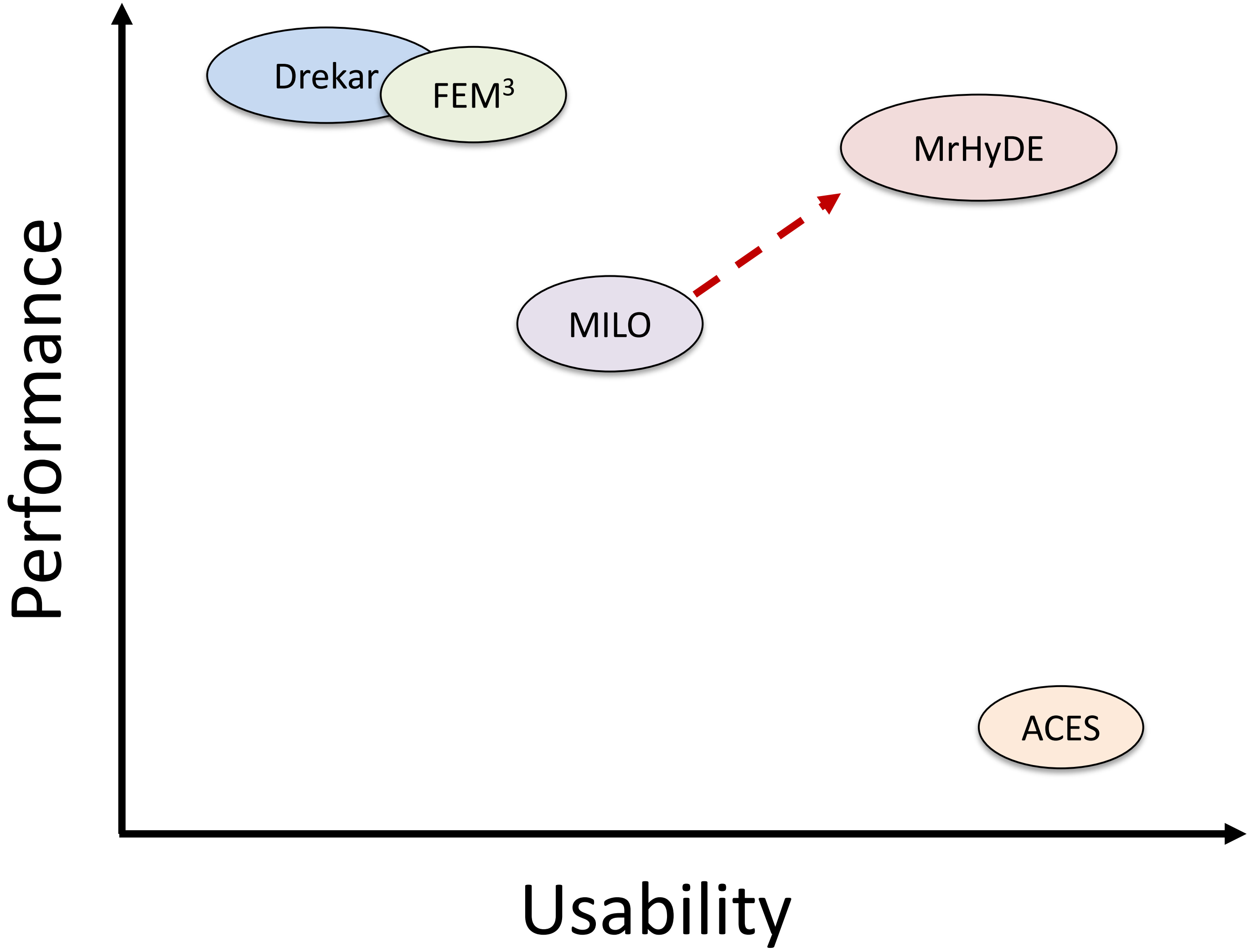
Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization

Usability and Performance



Disclaimer: This chart is based on the subjective assessment from one user/developer.
Not included: Albany, MFEM, deal.II, FEniCS, …

Performance Gains

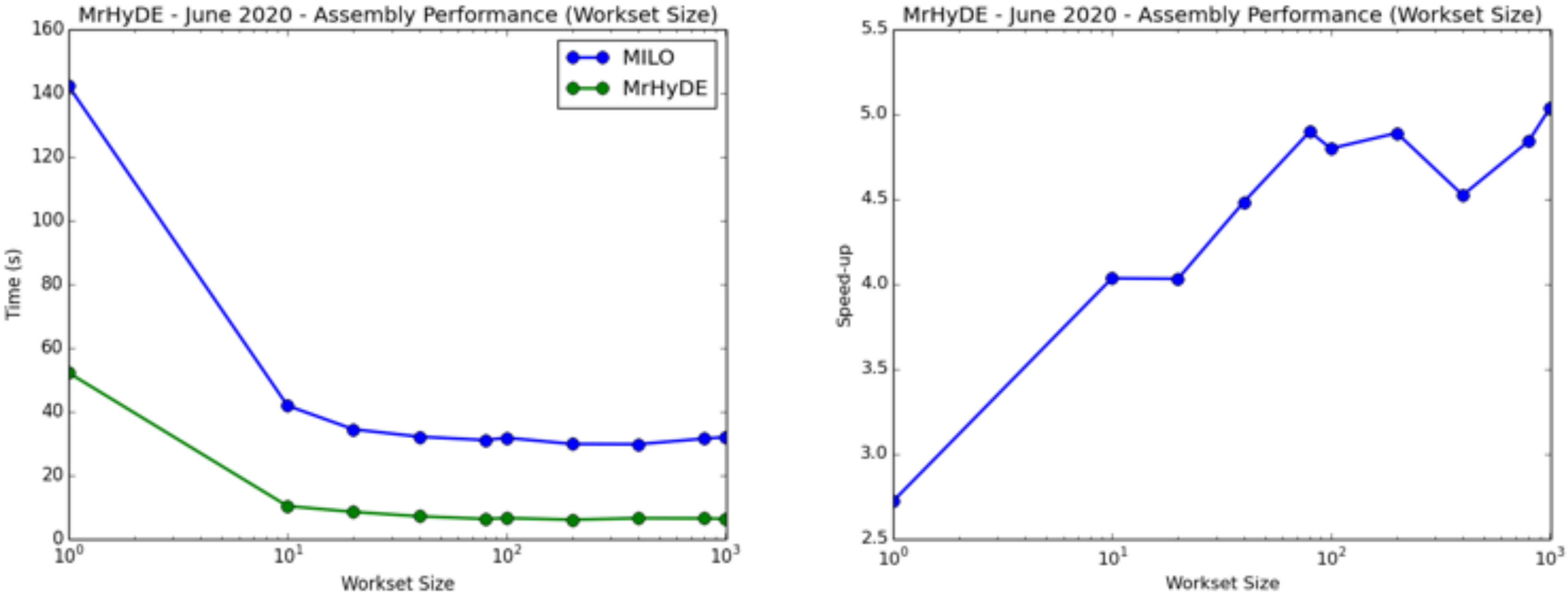
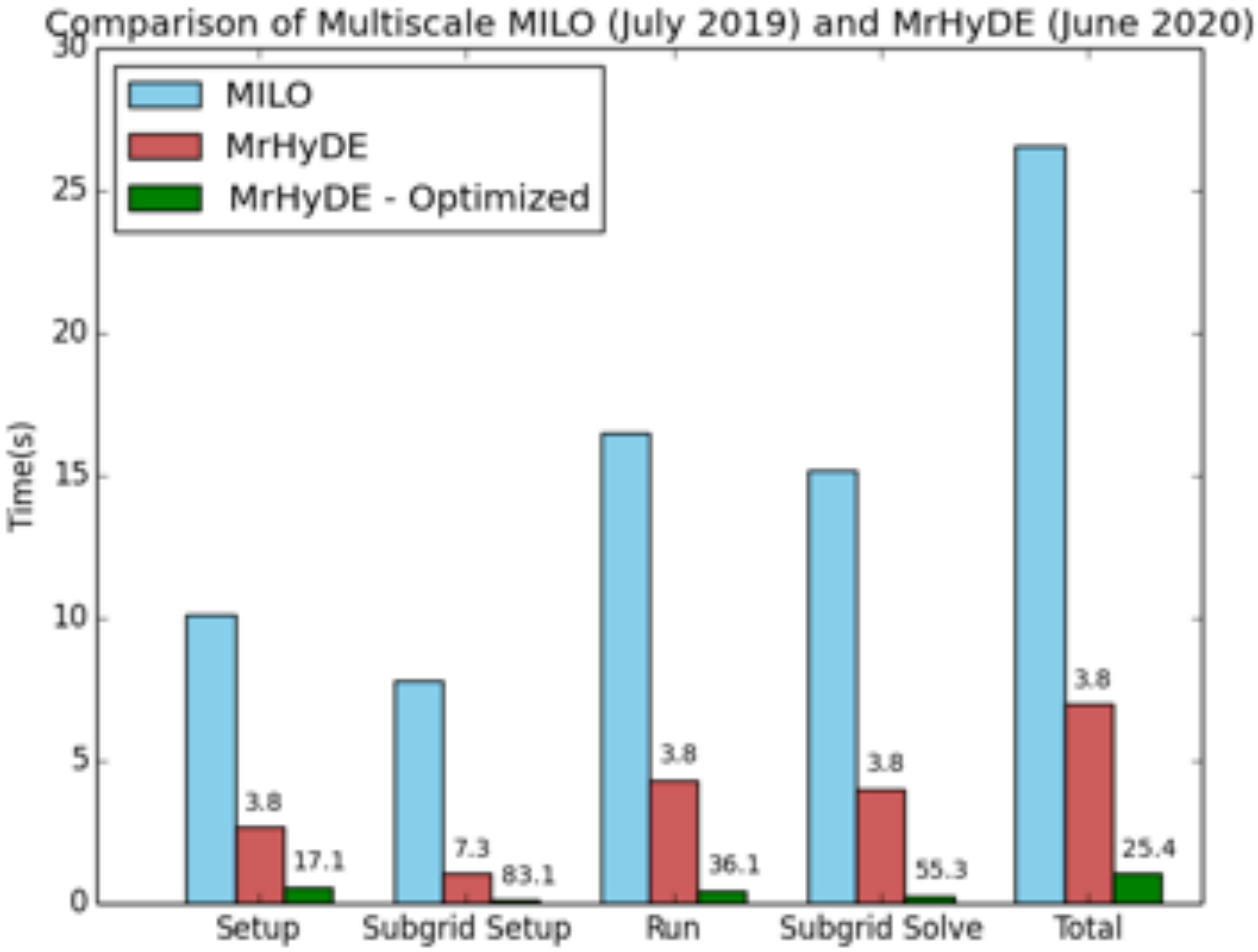


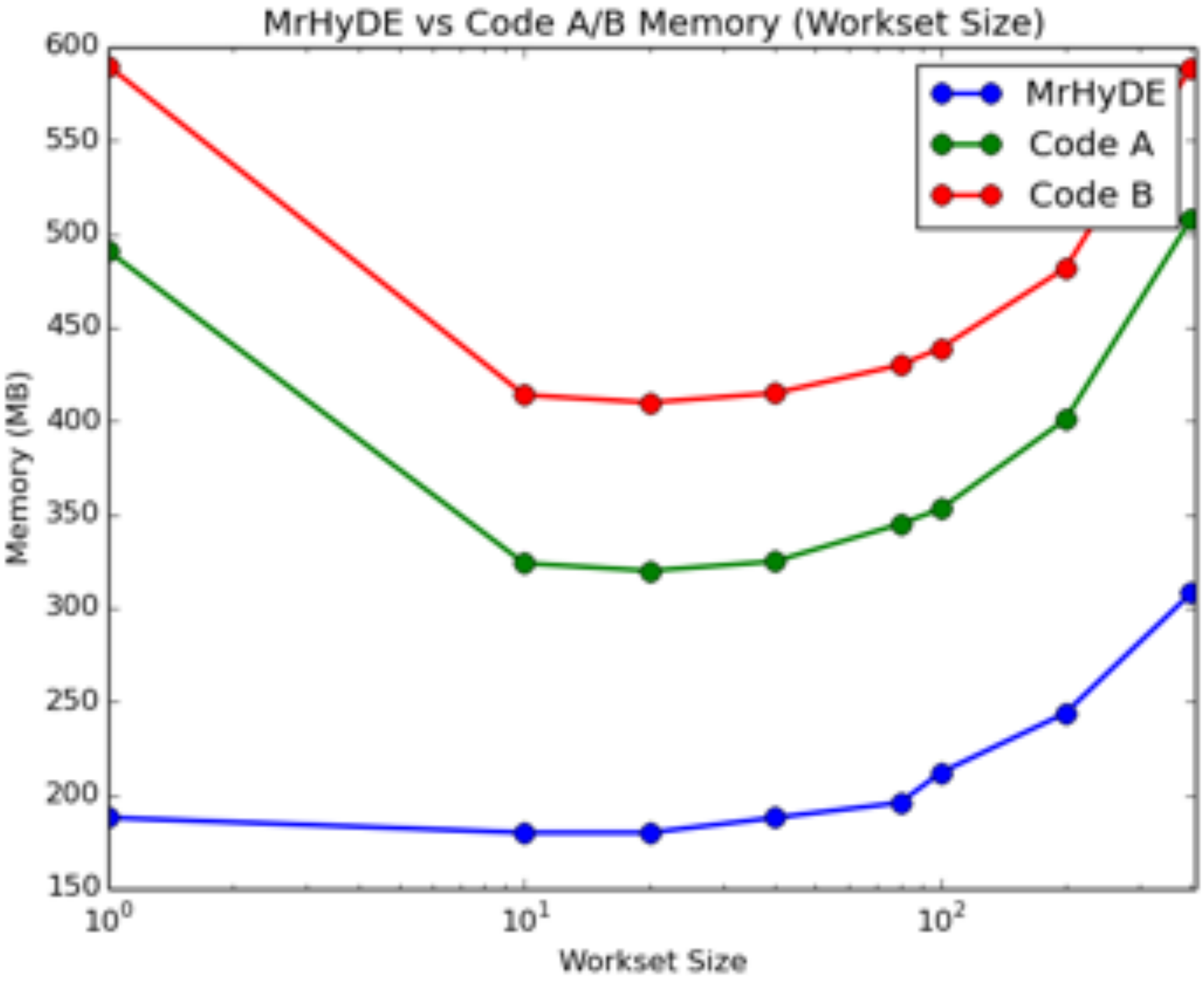
Figure: Comparison of total physics/assembly time between MILO-2019 and MrHyDE-2020 for transient nonlinear system with 40,000 elements and 300 linear systems.

A workset is basically a group of elements that get processed together.

Performance Gains

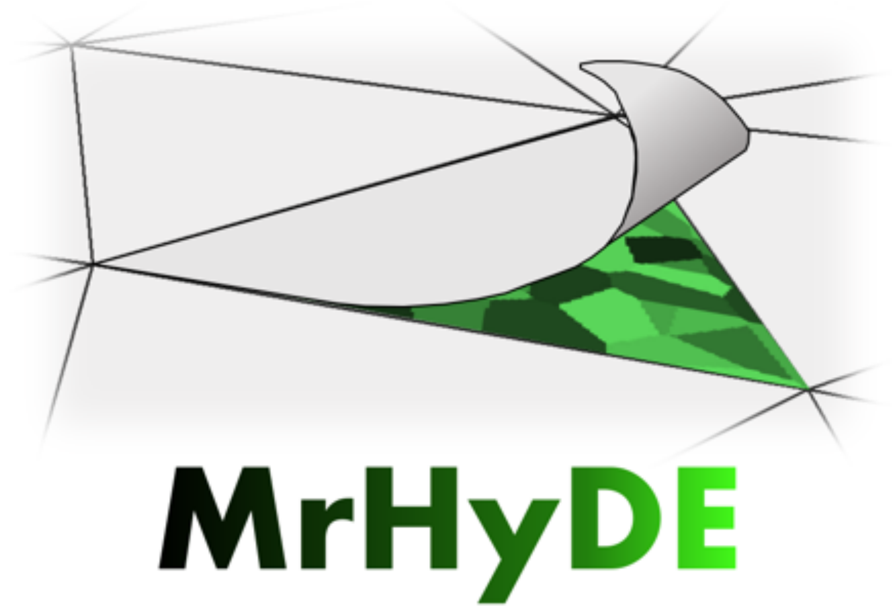


Memory Usage vs Another Trilinos-Based Code



What is MrHyDE?

- A C++ framework designed and optimized for solving multi-resolution hybridized differential equations.
- Provides an interface to powerful Trilinos tools within a user-friendly framework
- Portability with performance from laptops, to MPI-based clusters, to heterogeneous nodes, to MPI+X
- Ability to extract and inject data to develop data-informed physics-based simulations
- A modular and flexible environment for solving transient nonlinear multiphysics and multiscale systems in 1,2,3D
- Extensive set of examples/regression tests to maintain software quality and guide new users



How to Obtain and Build MrHyDE

- If you haven’t done so already, clone the MrHyDE repository

```
git clone https://github.com/TimWildey/MrHyDE.git
```

- Create a build directory (suggest MrHyDE/build)

```
cd MrHyDE
mkdir build
cd build
```

- Copy one of the CMake configure scripts from MrHyDE/scripts/configure-MrHyDE

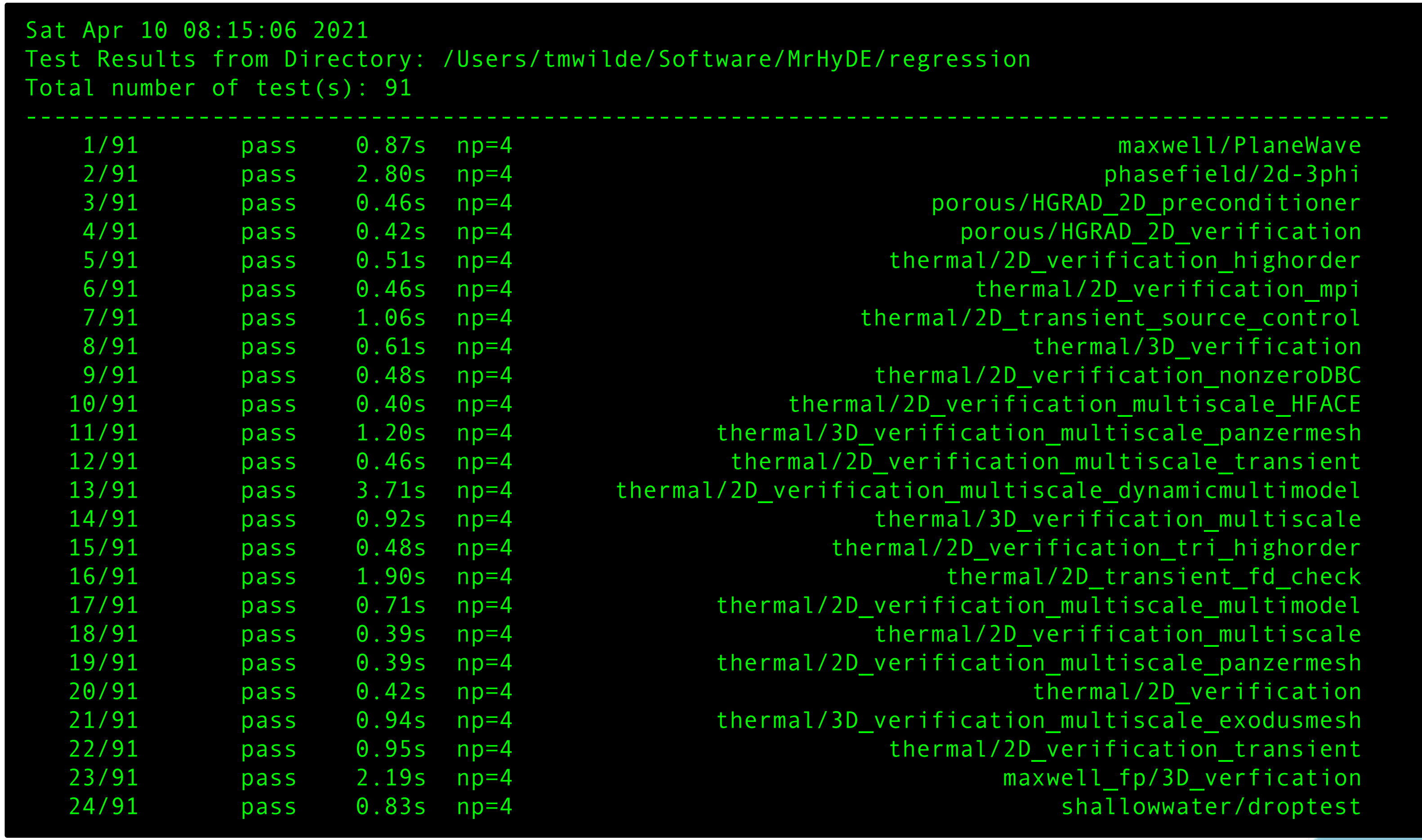
```
cp ../scripts/configure-MrHyDE/configure-MrHyDE-mac-catalina-serial configure
```

- Edit the Trilinos and MrHyDE paths in the configure file, then run

```
./configure
ninja
```

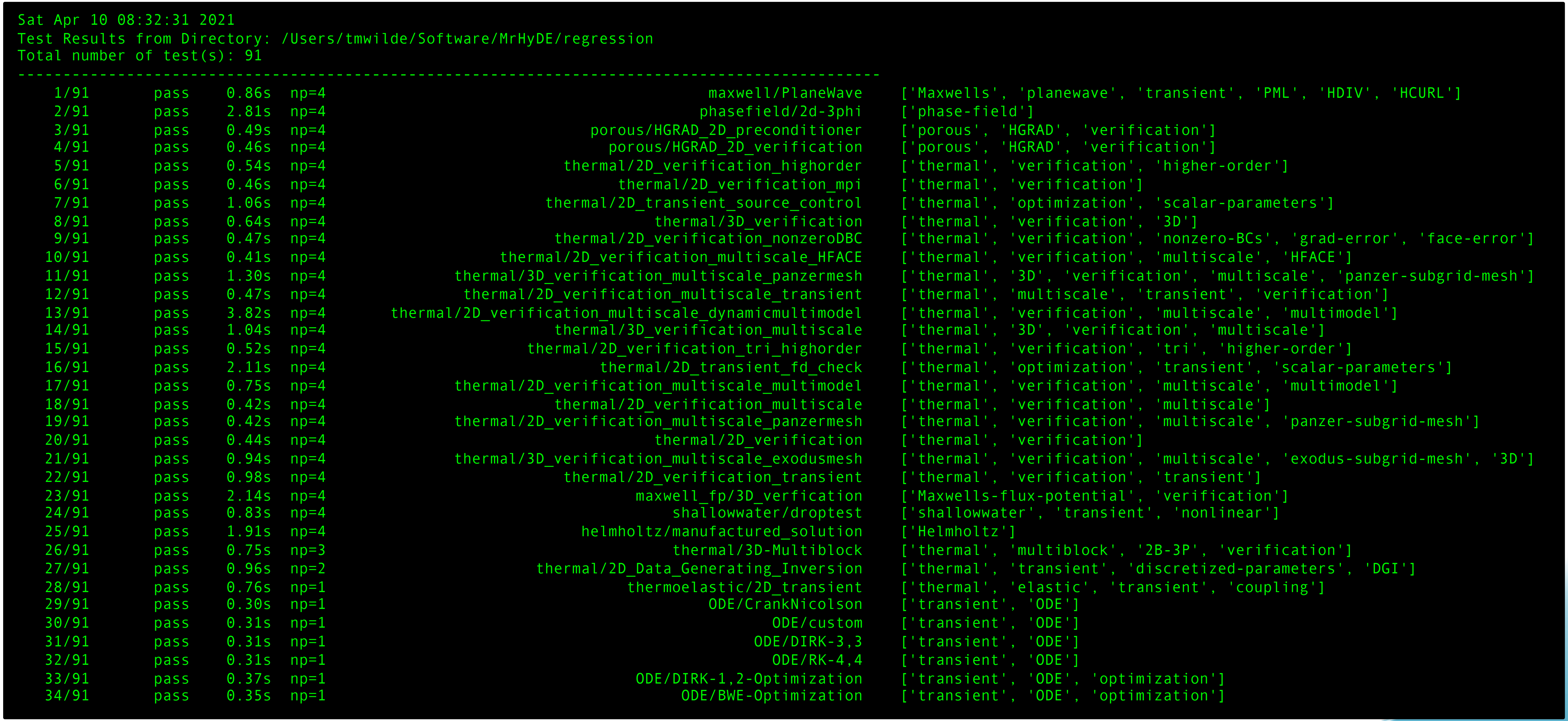
Regression Testing

- Python-based testing framework adapted from DGM by BvB/TS/TW
- Currently uses python2
  - Upgrade to python3 coming soon
- Currently 91 tests that also serve as a library of examples
- All are small tests that run in less than 5 seconds (on a mac)
- Easy to take one and scale up to 1000s of cores or heterogeneous nodes
- General guidelines for code contributions:
  - Run the tests before checking in code
  - If you add a capability, add a test that covers it



```
Sat Apr 10 08:15:06 2021
Test Results from Directory: /Users/tmwilde/Software/MrHyDE/regression
Total number of test(s): 91
-----------------------------------------------------------------------------------------------
     1/91      pass    0.87s   np=4                                        maxwell/PlaneWave
     2/91      pass    2.80s   np=4                                       phasefield/2d-3phi
     3/91      pass    0.46s   np=4                           porous/HGRAD_2D_preconditioner
     4/91      pass    0.42s   np=4                             porous/HGRAD_2D_verification
     5/91      pass    0.51s   np=4                        thermal/2D_verification_highorder
     6/91      pass    0.46s   np=4                              thermal/2D_verification_mpi
     7/91      pass    1.06s   np=4                      thermal/2D_transient_source_control
     8/91      pass    0.61s   np=4                                  thermal/3D_verification
     9/91      pass    0.48s   np=4                       thermal/2D_verification_nonzeroDBC
    10/91      pass    0.40s   np=4                 thermal/2D_verification_multiscale_HFACE
    11/91      pass    1.20s   np=4            thermal/3D_verification_multiscale_panzermesh
    12/91      pass    0.46s   np=4             thermal/2D_verification_multiscale_transient
    13/91      pass    3.71s   np=4     thermal/2D_verification_multiscale_dynamicmultimodel
    14/91      pass    0.92s   np=4                       thermal/3D_verification_multiscale
    15/91      pass    0.48s   np=4                    thermal/2D_verification_tri_highorder
    16/91      pass    1.90s   np=4                            thermal/2D_transient_fd_check
    17/91      pass    0.71s   np=4            thermal/2D_verification_multiscale_multimodel
    18/91      pass    0.39s   np=4                       thermal/2D_verification_multiscale
    19/91      pass    0.39s   np=4            thermal/2D_verification_multiscale_panzermesh
    20/91      pass    0.42s   np=4                                  thermal/2D_verification
    21/91      pass    0.94s   np=4            thermal/3D_verification_multiscale_exodusmesh
    22/91      pass    0.95s   np=4                        thermal/2D_verification_transient
    23/91      pass    2.19s   np=4                                maxwell_fp/3D_verfication
    24/91      pass    0.83s   np=4                                    shallowwater/droptest
```

Regression Testing



```
Sat Apr 10 08:32:31 2021
Test Results from Directory: /Users/tmwilde/Software/MrHyDE/regression
Total number of test(s): 91
-----------------------------------------------------------------------------------------------
   1/91       pass    0.86s  np=4                                     maxwell/PlaneWave    ['Maxwells', 'planewave', 'transient', 'PML', 'HDIV', 'HCURL']
   2/91       pass    2.81s  np=4                                    phasefield/2d-3phi    ['phase-field']
   3/91       pass    0.49s  np=4                        porous/HGRAD_2D_preconditioner    ['porous', 'HGRAD', 'verification']
   4/91       pass    0.46s  np=4                          porous/HGRAD_2D_verification    ['porous', 'HGRAD', 'verification']
   5/91       pass    0.54s  np=4                     thermal/2D_verification_highorder    ['thermal', 'verification', 'higher-order']
   6/91       pass    0.46s  np=4                           thermal/2D_verification_mpi    ['thermal', 'verification']
   7/91       pass    1.06s  np=4                   thermal/2D_transient_source_control    ['thermal', 'optimization', 'scalar-parameters']
   8/91       pass    0.64s  np=4                               thermal/3D_verification    ['thermal', 'verification', '3D']
   9/91       pass    0.47s  np=4                    thermal/2D_verification_nonzeroDBC    ['thermal', 'verification', 'nonzero-BCs', 'grad-error', 'face-error']
  10/91       pass    0.41s  np=4              thermal/2D_verification_multiscale_HFACE    ['thermal', 'verification', 'multiscale', 'HFACE']
  11/91       pass    1.30s  np=4         thermal/3D_verification_multiscale_panzermesh    ['thermal', '3D', 'verification', 'multiscale', 'panzer-subgrid-mesh']
  12/91       pass    0.47s  np=4          thermal/2D_verification_multiscale_transient    ['thermal', 'multiscale', 'transient', 'verification']
  13/91       pass    3.82s  np=4  thermal/2D_verification_multiscale_dynamicmultimodel    ['thermal', 'verification', 'multiscale', 'multimodel']
  14/91       pass    1.04s  np=4                    thermal/3D_verification_multiscale    ['thermal', '3D', 'verification', 'multiscale']
  15/91       pass    0.52s  np=4                 thermal/2D_verification_tri_highorder    ['thermal', 'verification', 'tri', 'higher-order']
  16/91       pass    2.11s  np=4                         thermal/2D_transient_fd_check    ['thermal', 'optimization', 'transient', 'scalar-parameters']
  17/91       pass    0.75s  np=4         thermal/2D_verification_multiscale_multimodel    ['thermal', 'verification', 'multiscale', 'multimodel']
  18/91       pass    0.42s  np=4                    thermal/2D_verification_multiscale    ['thermal', 'verification', 'multiscale']
  19/91       pass    0.42s  np=4         thermal/2D_verification_multiscale_panzermesh    ['thermal', 'verification', 'multiscale', 'panzer-subgrid-mesh']
  20/91       pass    0.44s  np=4                               thermal/2D_verification    ['thermal', 'verification']
  21/91       pass    0.94s  np=4         thermal/3D_verification_multiscale_exodusmesh    ['thermal', 'verification', 'multiscale', 'exodus-subgrid-mesh', '3D']
  22/91       pass    0.98s  np=4                     thermal/2D_verification_transient    ['thermal', 'verification', 'transient']
  23/91       pass    2.14s  np=4                             maxwell_fp/3D_verfication    ['Maxwells-flux-potential', 'verification']
  24/91       pass    0.83s  np=4                                 shallowwater/droptest    ['shallowwater', 'transient', 'nonlinear']
  25/91       pass    1.91s  np=4                       helmholtz/manufactured_solution    ['Helmholtz']
  26/91       pass    0.75s  np=3                                 thermal/3D-Multiblock    ['thermal', 'multiblock', '2B-3P', 'verification']
  27/91       pass    0.96s  np=2                  thermal/2D_Data_Generating_Inversion    ['thermal', 'transient', 'discretized-parameters', 'DGI']
  28/91       pass    0.76s  np=1                            thermoelastic/2D_transient    ['thermal', 'elastic', 'transient', 'coupling']
  29/91       pass    0.30s  np=1                                     ODE/CrankNicolson    ['transient', 'ODE']
  30/91       pass    0.31s  np=1                                            ODE/custom    ['transient', 'ODE']
  31/91       pass    0.31s  np=1                                          ODE/DIRK-3,3    ['transient', 'ODE']
  32/91       pass    0.31s  np=1                                            ODE/RK-4,4    ['transient', 'ODE']
  33/91       pass    0.37s  np=1                             ODE/DIRK-1,2-Optimization    ['transient', 'ODE', 'optimization']
  34/91       pass    0.35s  np=1                                  ODE/BWE-Optimization    ['transient', 'ODE', 'optimization']
```

Exercise: Run the Regression Tests

- Go into MrHyDE/regression
- Assume your build directory is in MrHyDE/build, create a soft link in the regression folder

```
ln -s ../build/src/mrhyde
```

- Now, just run the tests

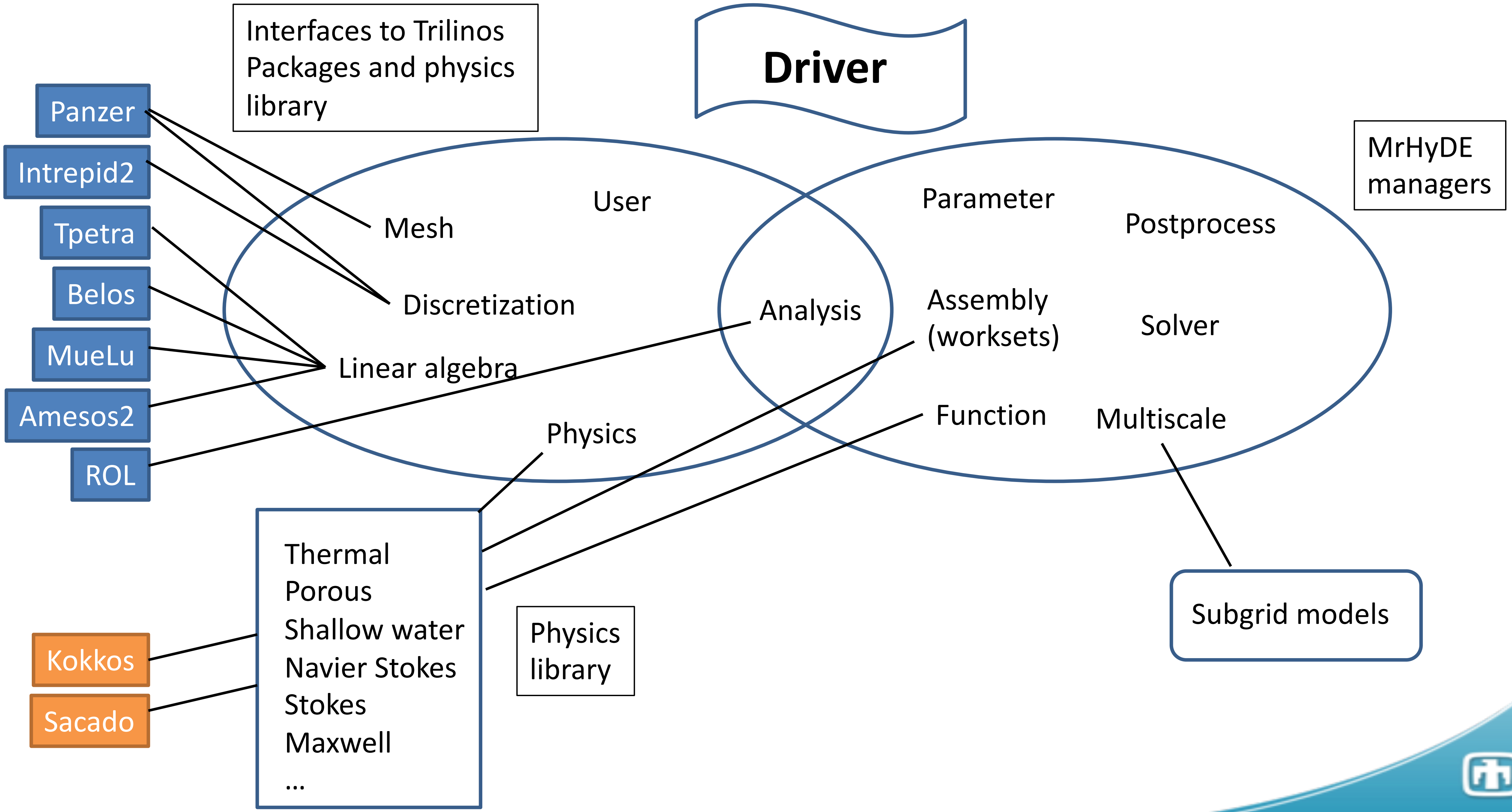
```
python runtests.py
```

- Go into one of the tests and modify the input file to do something else
- Run the tests again and you should see a failure
- To visualize the solutions, add the following in the Postprocess sublist:

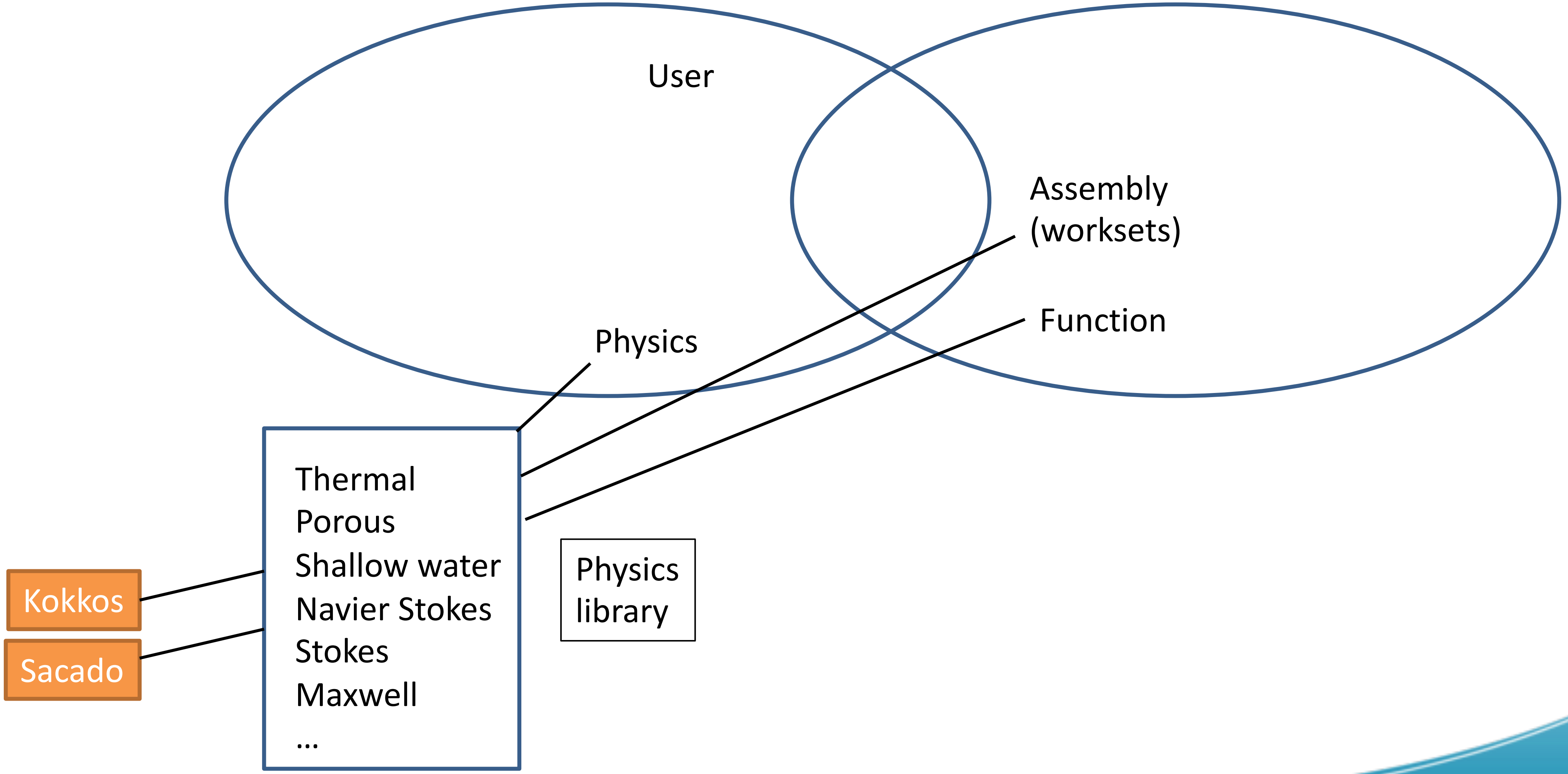
```
write solution: true
```

- This will create an exodus file. To visualize this, use ParaView
- If you work at Sandia: https://onestop.sandia.gov/paraview
- Otherwise: https://www.paraview.org/download/

MrHyDE Organization



MrHyDE Organization


User

Physics

Assembly (worksets)

Function

Thermal
Porous
Shallow water
Navier Stokes
Stokes
Maxwell
…

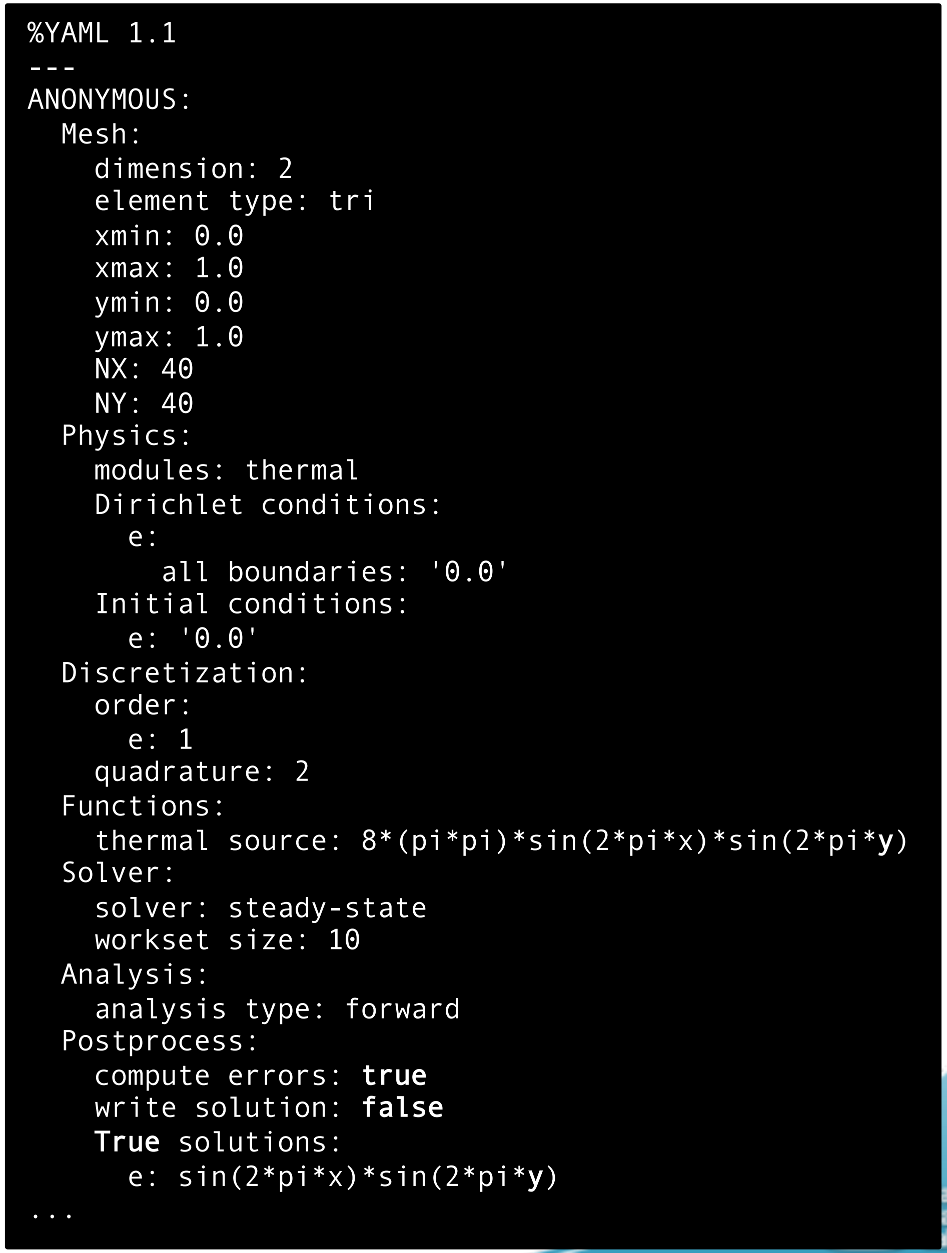
Physics library

Kokkos

Sacado

Navigating the User Interface and Input File

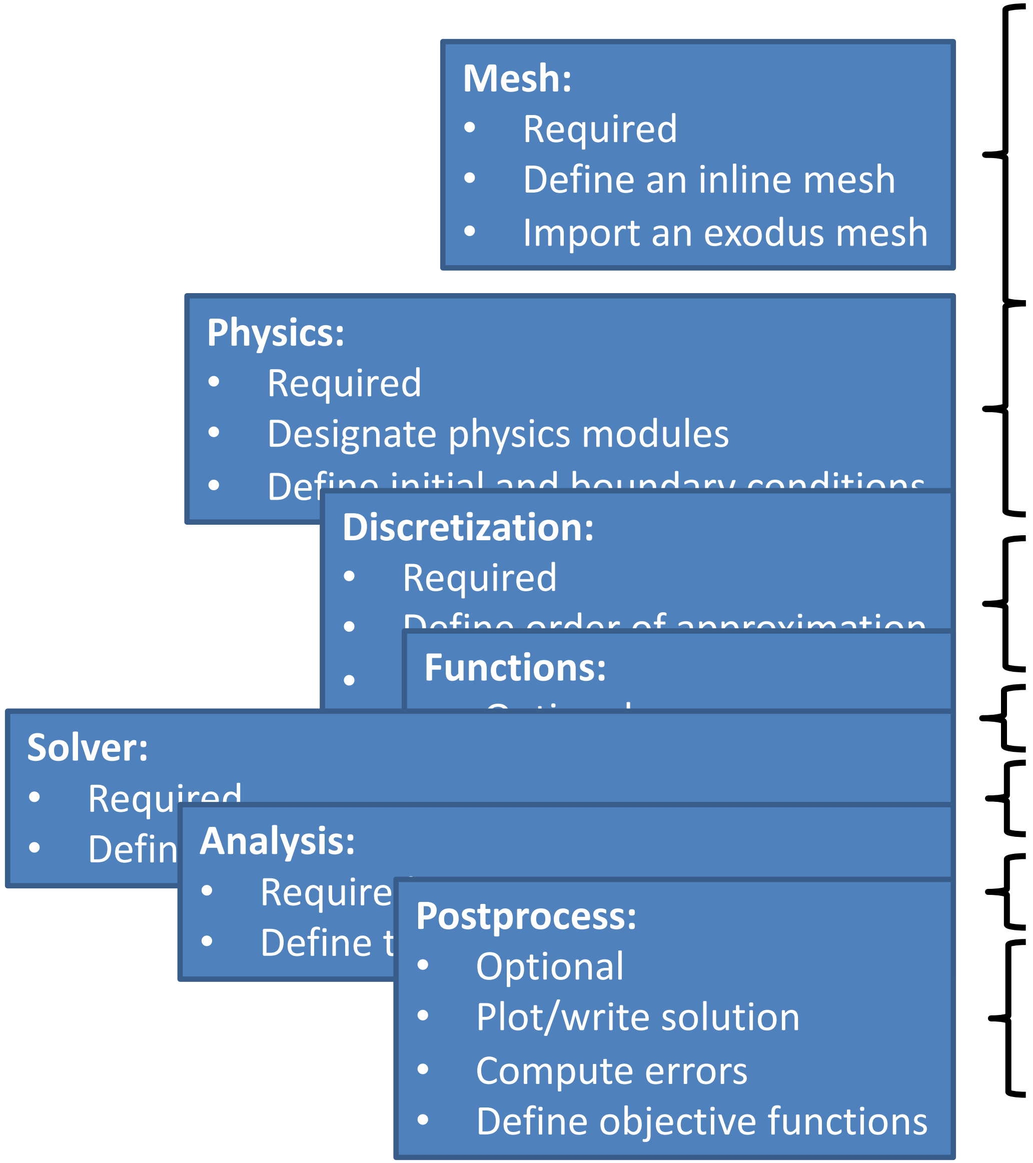
- MrHyDE primarily uses the YAML format
  - XML is also an option
- **YAML** (a recursive acronym for "YAML Ain't Markup Language") is a human-readable data-serialization language. [1]
- The user interface looks for input.yaml
- For examples, see the regression tests
- Also, see `MrHyDE/scripts/input-files` for all the available options.
- There are 5 required blocks and 4 options blocks
- Automatically determines data type
  - `int, double, char`
- Can force a char by using single quotes



```
%YAML 1.1
---
ANONYMOUS:
  Mesh:
    dimension: 2
    element type: tri
    xmin: 0.0
    xmax: 1.0
    ymin: 0.0
    ymax: 1.0
    NX: 40
    NY: 40
  Physics:
    modules: thermal
    Dirichlet conditions:
      e:
        all boundaries: '0.0'
    Initial conditions:
      e: '0.0'
  Discretization:
    order:
      e: 1
    quadrature: 2
  Functions:
    thermal source: 8*(pi*pi)*sin(2*pi*x)*sin(2*pi*y)
  Solver:
    solver: steady-state
    workset size: 10
  Analysis:
    analysis type: forward
  Postprocess:
    compute errors: true
    write solution: false
    True solutions:
      e: sin(2*pi*x)*sin(2*pi*y)
...
```

[1] Source: wikipedia

Navigating the User Interface and Input File



Mesh:
- Required
- Define an inline mesh
- Import an exodus mesh

Physics:
- Required
- Designate physics modules
- Define initial and boundary conditions

Discretization:
- Required
- Define order of approximation
-

Functions:
- Optional

Solver:
- Required
- Defin

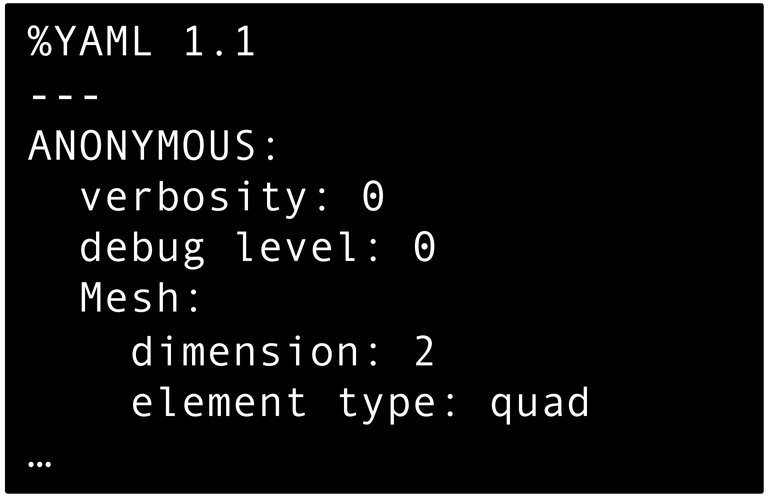
Analysis:
- Require
- Define t

Postprocess:
- Optional
- Plot/write solution
- Compute errors
- Define objective functions

```
%YAML 1.1
---
ANONYMOUS:
  Mesh:
    dimension: 2
    element type: tri
    xmin: 0.0
    xmax: 1.0
    ymin: 0.0
    ymax: 1.0
    NX: 40
    NY: 40
  Physics:
    modules: thermal
    Dirichlet conditions:
      e:
        all boundaries: '0.0'
    Initial conditions:
      e: '0.0'
  Discretization:
    order:
      e: 1
    quadrature: 2
  Functions:
    thermal source: 8*(pi*pi)*sin(2*pi*x)*sin(2*pi*y)
  Solver:
    solver: steady-state
    workset size: 10
  Analysis:
    analysis type: forward
  Postprocess:
    compute errors: true
    write solution: false
    True solutions:
      e: sin(2*pi*x)*sin(2*pi*y)
...
```

How to Get More Output

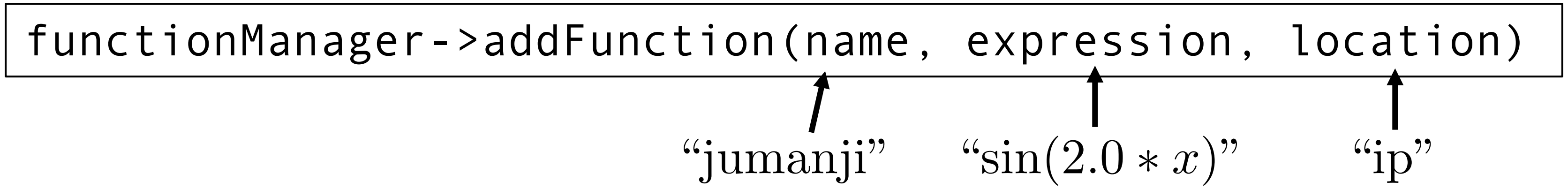
- verbosity
  - Default: 0
  - >0 print time step information
  - >1 print nonlinear solver information
  - >8 print linear solver information
  - >=10 print Teuchos timer information (very useful)
  - >20 MueLu preconditioner information
- debug level
  - Default: 0
  - 1: print status when going into manager/interface constructors and some functions that are only called once during setup phase
  - 2: also print status when going into other functions that are called many times during setup or run
  - 3: also print vectors, matrices, some Views



```
%YAML 1.1
---
ANONYMOUS:
  verbosity: 0
  debug level: 0
  Mesh:
    dimension: 2
    element type: quad
…
```

The Function Manager

- One of the most important pieces of MrHyDE for a user to understand
- Similar to Phalanx – builds Directed Acyclic Graphs (DAGs)
- Distinguishing feature: an interpreter that turns strings into DAGs
- Can be thought of as an auto-generator of evaluators (although it doesn't use PHX::evaluator)
- To add a function:



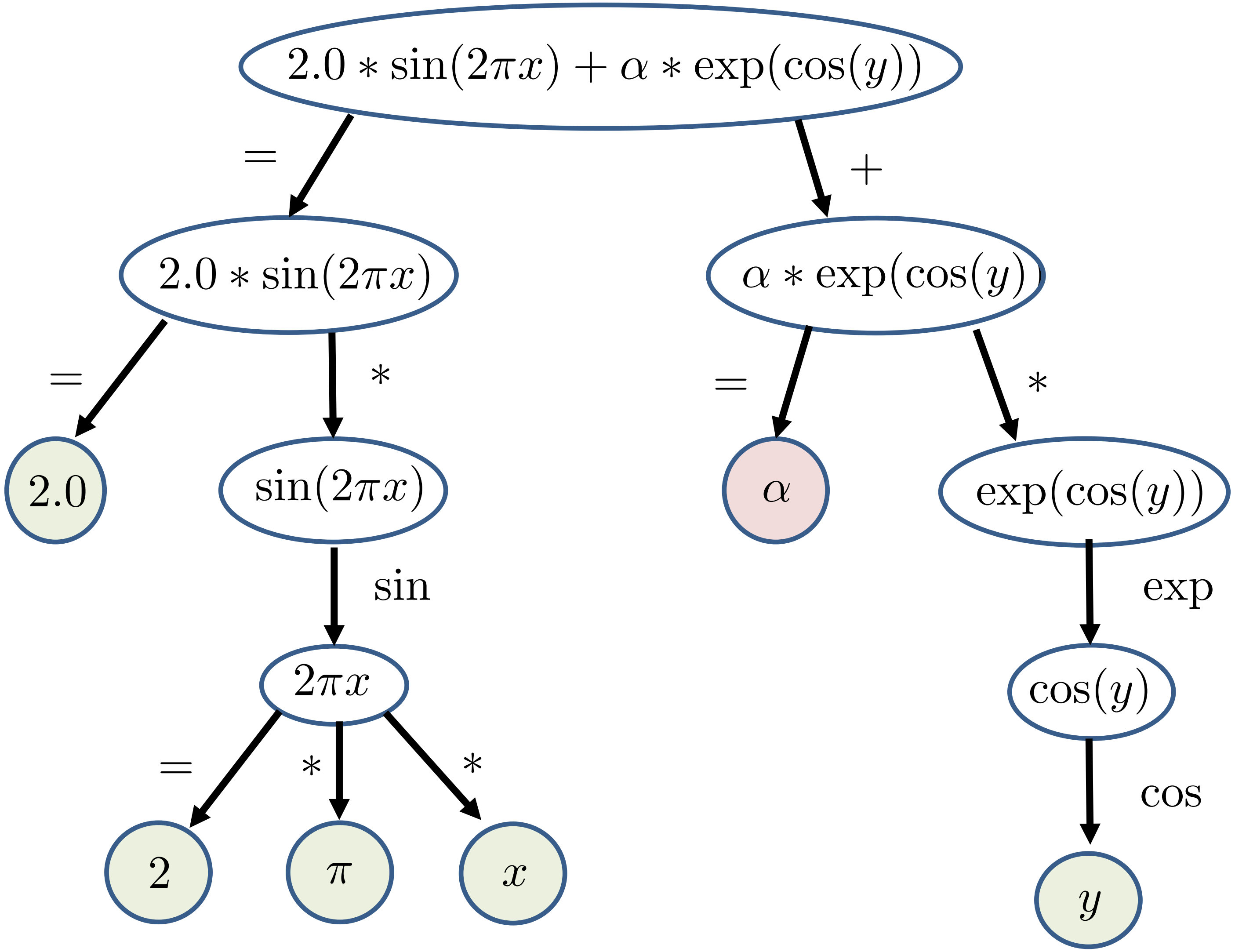
```
functionManager->addFunction(name, expression, location)
```

- To evaluate a function:

```
auto data = functionManager->evaluate(name, location)
```

- But, how does it decompose a function into a DAG?

Directed Rooted Trees
Special Case of a Directed Acyclic Graph (DAG)

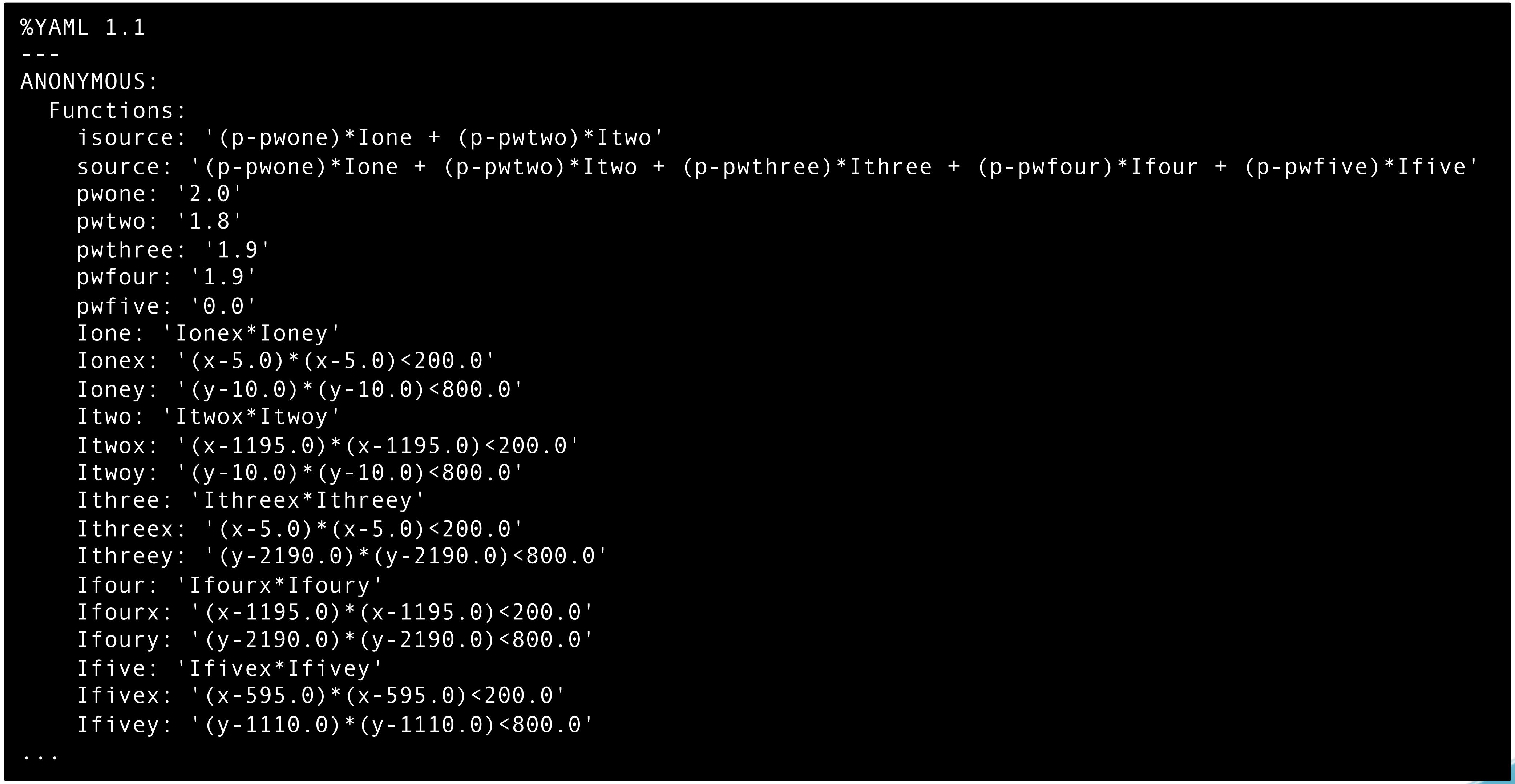


What can be used in expressions?

Function managers are aware of a few types of variables:

- Spatial variables: x, y, z
- Time: t
- The value of \pi: pi
- Scalars: 2.0, 1.3e-4, 2.2E-3, 0.0000001, etc.
- Scalar parameters: lambda
- Components of vector parameters: eta[1]
- Discretized parameters: mu
- Solution variables: u, dx, pr, Hu, etc.
- Components of vector variables: B[x]
- Components of grad of HGRAD variables: grad(dx)[x]
- Components of curl of HCURL variables: curl(E)[y]
- Divergence of HDIV variables: div(B)
- Names of other functions, responses, objectives: source, obj0
- Normals on sides/faces: nx, ny, nz

Functions Sublist can be Arbitrarily Complicated[1]



```
%YAML 1.1
---
ANONYMOUS:
  Functions:
    isource: '(p-pwone)*Ione + (p-pwtwo)*Itwo'
    source: '(p-pwone)*Ione + (p-pwtwo)*Itwo + (p-pwthree)*Ithree + (p-pwfour)*Ifour + (p-pwfive)*Ifive'
    pwone: '2.0'
    pwtwo: '1.8'
    pwthree: '1.9'
    pwfour: '1.9'
    pwfive: '0.0'
    Ione: 'Ionex*Ioney'
    Ionex: '(x-5.0)*(x-5.0)<200.0'
    Ioney: '(y-10.0)*(y-10.0)<800.0'
    Itwo: 'Itwox*Itwoy'
    Itwox: '(x-1195.0)*(x-1195.0)<200.0'
    Itwoy: '(y-10.0)*(y-10.0)<800.0'
    Ithree: 'Ithreex*Ithreey'
    Ithreex: '(x-5.0)*(x-5.0)<200.0'
    Ithreey: '(y-2190.0)*(y-2190.0)<800.0'
    Ifour: 'Ifourx*Ifoury'
    Ifourx: '(x-1195.0)*(x-1195.0)<200.0'
    Ifoury: '(y-2190.0)*(y-2190.0)<800.0'
    Ifive: 'Ifivex*Ifivey'
    Ifivex: '(x-595.0)*(x-595.0)<200.0'
    Ifivey: '(y-1110.0)*(y-1110.0)<800.0'
...
```

[1] Within reason. There is a limit on the number of recursions, but this can be increased.

Exercise: Create a New Physics Module

- We are going to implement the following PDE:

$$-\Delta \{\text{varname}\} + c\,\{\text{varname}\} = s(x)$$

- Step 1: choose a name for your variable, e.g., “llama” or “coconut”
- Step 2: choose a name for the physics module, e.g., “coconuts”
- Step 3: copy the template (newmodule.hpp) from MrHyDE/doc/Tutorial/Example into MrHyDE/src/physics

```
cd MrHyDE/src/physics
cp ../../doc/Tutorial/Example/newmodule.hpp coconuts.hpp
```

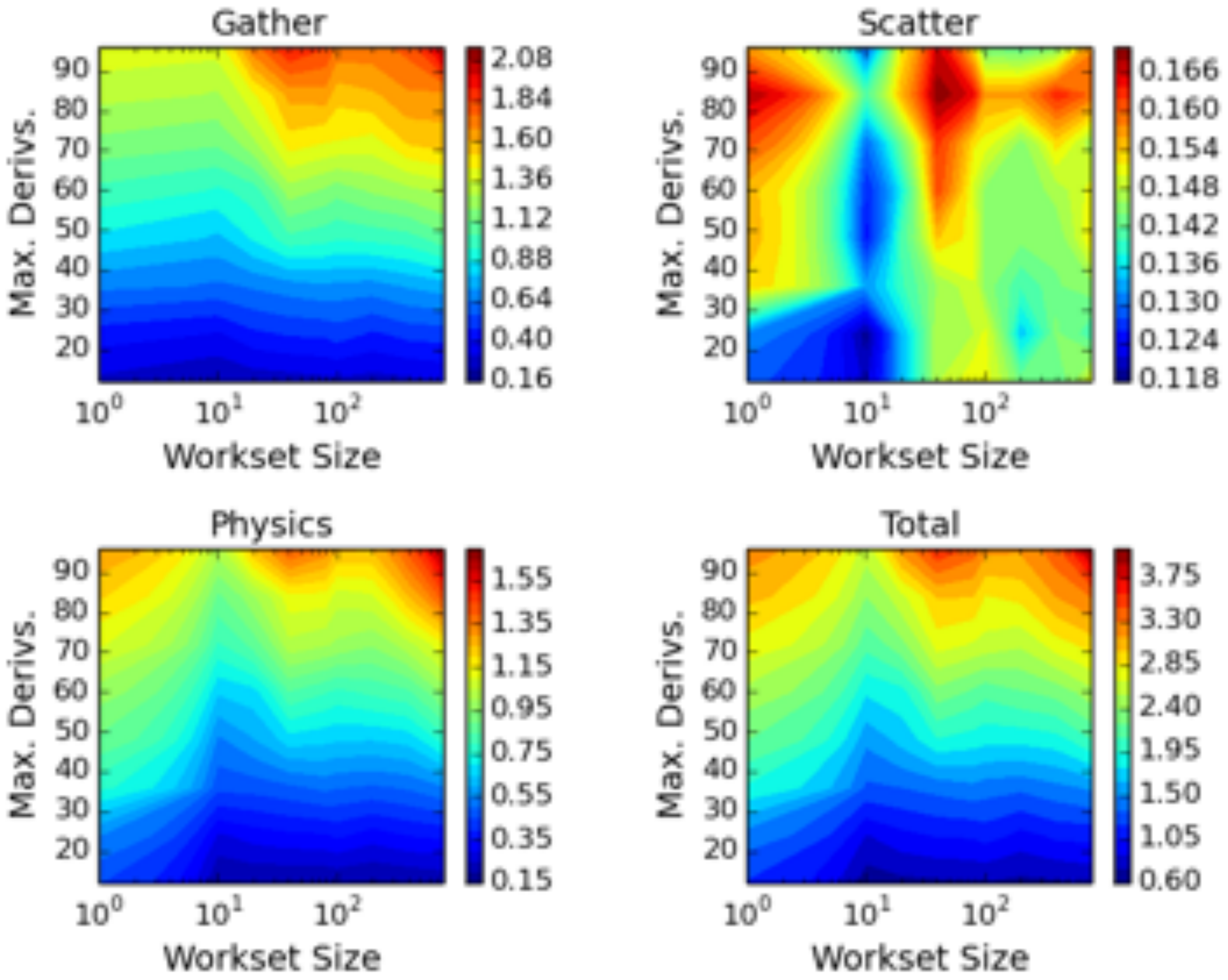
- Step 4: open the file and edit the variable and module names. The weak residual is already there.
- Step 5: make MrHyDE aware of the new module. Open physicsImporter.cpp and add the header and the constructor for your new module (just copy, paste and edit an existing one)
- Step 6: recompile MrHyDE
- Step 7: go back to MrHyDE/doc/Tutorial/Example and edit input.yaml
- Step 8: Create a soft link to MrHyDE/build/src/mrhyde
- Step 9: Run your new module
- Step 10: visualize the results, change the PDE from the input file, couple with other modules, etc.

Boosting Performance

- The default settings are suboptimal to allow all the regression tests to run
- For a specific problem, performance can be optimized without modifying the code
- The default workset size is 100.
  - This is a runtime option to define the number of elements that get processed together
  - 1 is almost never optimal
  - Larger values increase memory requirements
  - Even without threading, adjusting this can improve performance
  - Optimal number is problem dependent
- The default number of derivatives for the SFAD AD objects is 64
  - This is a compile time option (due to Sacado SFAD)
  - Tailoring this can significantly improve performance
  - Minimum value is the maximum of: DOFs per element, active parameters, discretized parameter DOFs per element
  - For example, for the shallow water equations in 2D using linear basis, maxDerivs = 12
  - Adjusted in the MrHyDE configure script:

```
-D MrHyDE_MAX_DERIVS=64 \
```

Impact of Performance Tuning



Preview of Tomorrow

Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization