PRESSIO-DEMOAPPS

Francesco Rizzi (NexGen Analytics), Patrick Blonigan (SNL),
Eric Parish (SNL), John Tencer (SNL),
Jorio Cocola (SNL), Marcin Wrobel (NGA)

SynS & ML Workshop @ ICML 2023

This work was funded by the Advanced Simulation and Computing program and the Laboratory Directed Research and Development program at Sandia National Laboratories, 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-NA-0003525

WHAT IS PRESSIO-DEMOAPPS?

A collection of 1D, 2D and 3D problems of varying complexity spanning multiple areas: from linear advection, to diffusion and reaction-diffusion, and compressible Euler (fluids).

Originally started as part of the Pressio* project to create a suite of benchmark problems to test model reduction methods.

Evolved into a self-contained framework: for example, it can be used for doing “standard” simulations, or one can just use the Python meshing scripts, or leverage the sample mesh capability to study function approximations.

* https://github.com/Pressio

MAIN FEATURES

- Grounded on well-established numerical methods for PDEs
- Offers both a C++ and Python API
  - Header-only, depends on: Python, C++, Eigen, simple to build
- Cell-centered finite volume discretization with various numerical schemes (e.g. for handling sharp gradients) and *exact Jacobians*
- Built-in support for sample mesh (more later)
- Development principles:
  - **simplicity:** focus on self-contained and well-defined problems
  - **quality assurance** (*every* problem is tested, github CI is active)
  - **extensibility**

CURRENTLY SUPPORTED PROBLEMS

1d Problems

- 1D Linear Advection
- 1D single-species reaction diffusion
- 1D Euler Smooth
- 1D Euler Sod
- 1D Euler Lax
- 1D Euler Shu-Osher

2d Problems

- 2D single-species reaction diffusion
- 2D Burgers (Periodic BCs)
- 2D Gray Scott reaction-diffusion
- 2D Shallow water equations
- 2D Euler Smooth
- 2D Euler Kelvin-Helmholtz
- 2D Euler Sedov Full
- 2D Euler Sedov (with symmetry)
- 2D Euler Riemann
- 2D Euler Normal Shock
- 2D Euler Double Mach Reflection

3d Problems

- 3D Euler Smooth
- 3D Euler Sedov (with symmetry)

How did we get to this set of problems?
This is just a start, but already quite broad:

- from smooth fields to sharp gradients and discontinuities
- key physics is covered: advection, diffusion, reaction, gas dynamics

HOW DOES IT WORK?
IT IS A 3-STEP PROCESS

Choose a problem: e.g. the Sod1D

Create the mesh:

```
python3 pressio-demoapps/meshing_scripts/create_full_mesh_for.py \
        --outdir $HOME/myTestMesh \
        --problem sod1d_s7 \
        -n 100
```

Instantiate the problem (Python snippet):

```
import pressiodemoapps as pda
# ...
meshObj = pda.load_cellcentered_uniform_mesh("/home/myTestMesh")
order   = pda.InviscidFluxReconstruction.Weno5;
problem = pda.create_problem(meshObj, pda.Euler1d.Sod, order)
```

Use the problem object: next slide

PROBLEM OBJECT API

Any problem in demo apps reduces to this semi-discrete form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t, \ldots)$$

y: state vector of the degrees-of-freedom at all cells

f: right-hand-side

t: time

```
1  class Problem:
2
3    # Constructs and returns an instance of
4    # the initial condition for the target problem.
5    def initialCondition(self):
6      return # numpy array
7
8    # Constructs and returns an instance of the rhs.
9    def createRightHandSide(self):
10     return # numpy array
11
12   # Constructs and returns an instance of the
13   # Jacobian's action applied to the operand.
14   def createApplyJacobianResult(self, operand):
15     return # numpy array
16
17   # Computes rhs = f(y,time,...)
18   def rightHandSide(self, y, time, rhs):
19     return;
20
21   # Computes  df/dy(y,time,...) * operand
22   def applyJacobian(self, y, operand, time, result):
23     return;
```

SAMPLE RESULTS (1D): SOD PROBLEM

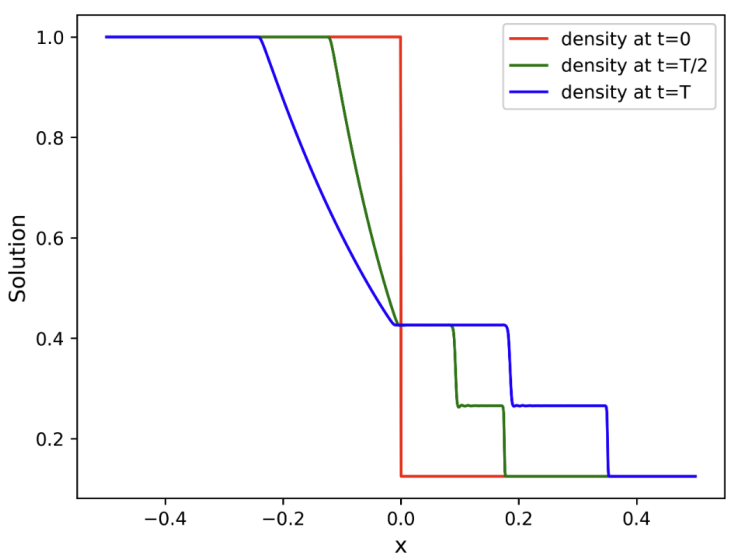
This problem solves the *1D conservative Euler equations* for the Sod1d problem.

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho u\\ \rho E\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho u\\ \rho u^2+p\\ u(E+p)\end{bmatrix}=0$$

where the pressure p is related to the conserved quantities through the equation of the state

$$p=(\gamma-1)(\rho E-\frac{1}{2}\rho u^2).$$

Representative plot for total simulation time $T=0.2$ showing density at selected time steps $t\in\{0,0.1,0.2\}$ obtained using time step $dt=10^{-4}$, Weno5, Runge-Kutta4 integration with a mesh of $N=1000$ cells.



SAMPLE RESULTS (2D): GRAY-SCOTT REACTION-DIFFUSION

This problem focuses on the following 2D reaction-diffusion system of PDE:

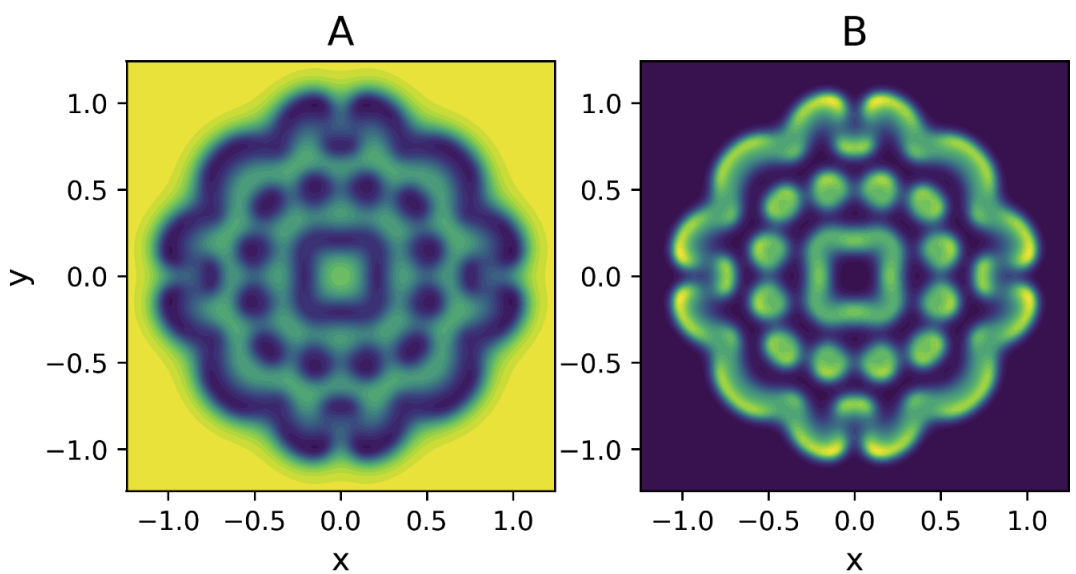
$$\frac{\partial A}{\partial t} = D_a \left(\frac{\partial^2 A}{\partial x^2} + \frac{\partial^2 A}{\partial y^2} \right) - AB^2 + F(1 - A)$$
$$\frac{\partial B}{\partial t} = D_b \left(\frac{\partial^2 B}{\partial x^2} + \frac{\partial^2 B}{\partial y^2} \right) + AB^2 - (F + K)B$$

- D_a, D_b, F, K can be provided to the constructor (more below)

Plots at t=1000 obtained using time step dt=0.5, Runge-Kutta4 integration, a mesh of `160x160`.

- $D_a = 2 \cdot 10^{-4}, D_b = D_a/4$
- $F = 0.042, K = 0.062$

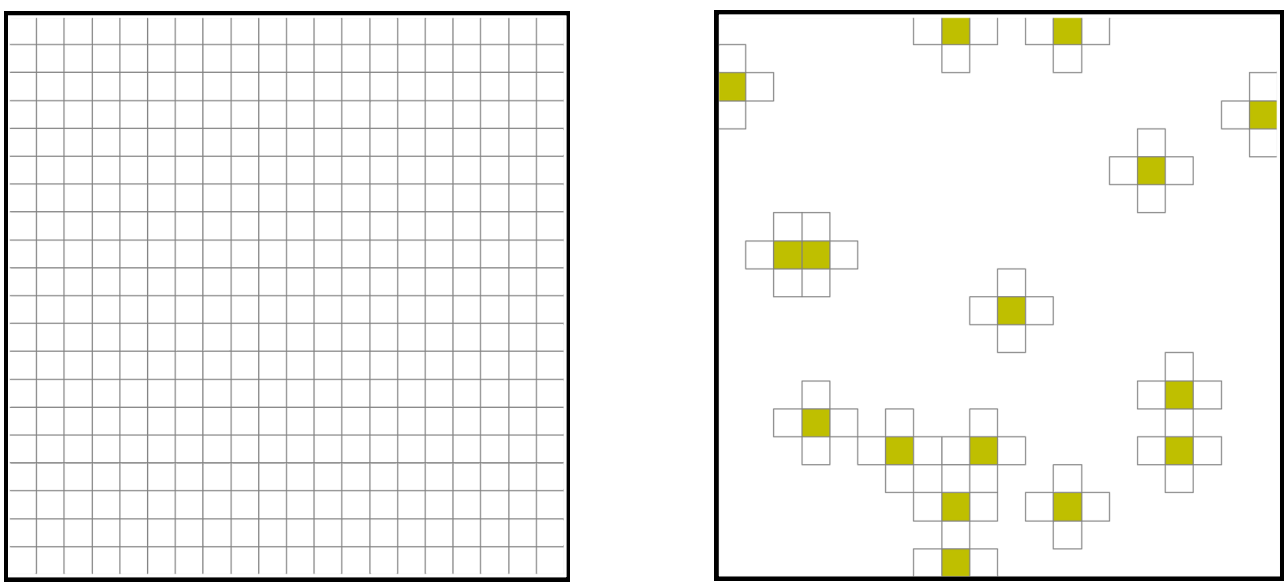
- Domain is $[-5/4, 5/4]^2$ with periodic BCs



SAMPLE MESH FEATURE

A key feature of pressio-demoapps is its native support for *sample mesh*. It is a disjoint collection of cells where the RHS (or residual) vector and Jacobian matrix of the target system are computed.

Can be used for several purposes, e.g., exploring discrete function approximation, hyper-reduction of nonlinear operators.



pressio.github.io/pressio-demoapps/meshsample.html

CONCLUSIONS

pressio-demoapps is a collection of well-defined problems
Implementation is grounded on well-established numerical methods

Covers a broad range of physics: conduction, convection, reaction, diffusion, etc.

Scientific reproducibility is critical

Can be used for several purposes, and can provide a well-defined framework to experiment with ML applications

francesco.rizzi@ng-analytics.com

https://pressio.github.io/pressio-demoapps/