Kokkos Kernels: ODE

Luc Berger-Vergiat, Sandia National Laboratories

Kokkos User Group Meeting 2023

December 12, 2023

Sandia National Laboratories is a multi-mission 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.
SAND2023-14632C

Recall the structure of ODEs

$$\frac{\partial y}{\partial t} = f(y, t), \quad y(0) = y_0$$

Two major families of solvers can be used

1. Explicit time integrators
   - Only require function evaluations and vector operations
   - Easy to implement, flexible order of integration
   - Require small time step for appropriate convergence (CFL condition)
   - Can easily support adaptive time stepping
2. Implicit time integrators
   - Requires Jacobian evaluation and non-linear solver
   - Can be time-step and order adaptive but needs a complex control flow
   - Uses larger time-steps and can solve stiff system of equations

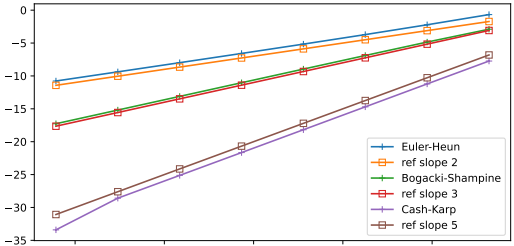
Primary goal is to support solution of many ODEs concurrently.
Support both explicit and implicit adaptive solvers

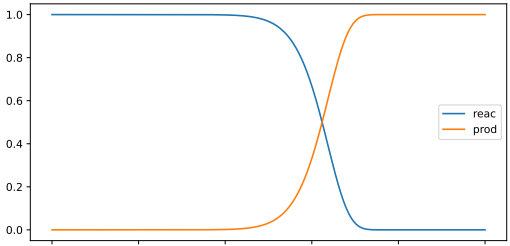
- Explicit Runge-Kutta order 1 to 5
- Implicit Backward Difference Formula, order 1 to 5
- Implicit Adams-Moulton orders 1 to 6 (maybe up to 12)

Single problem and batched problems interfaces

- Explicit and implicit solvers will both have single problem interface on device (callable within inner most parallel level)
- Explicit algorithms will have batched interface, implicit ones need more evaluation for batched approach

- Implementation of single problem interface complete
- All classic variants from order 1 to 5 implemented (Fehlberg, Bogacki-Shampine, Cash-Karp, Dormand-Prince)
- Batched interface and implementation underway
- Dormand-Prince 8-5-3 variant could be next feature?
- Symplectic algorithms maybe considered?





- Implementation of single problem complete
- Performance optimization still underway
  - memory foot print reduction
  - Jacobian calculation and factorization reuse
- Only dense Jacobian supported, sparse coming next