Capabilities: Time Integration

FASTMath Team Members: Cody J. Balos, David J. Gardner, Daniel R. Reynolds, Steven B. Roberts, and Carol S. Woodward

SUNDIALS provides robust and efficient adaptive time integrators with sensitivity capabilities for ODEs and DAEs along with an efficient nonlinear solver package for incorporation in large-scale scientific application codes

SUNDIALS Overview

Suite of adaptive ODE and DAE time integrators and nonlinear solvers

- Adaptive time integrators with forward and adjoint sensitivity analysis capabilities
- Written in C with interfaces to Fortran and C++

Designed to be easily incorporated into existing codes

- Packages are built on shared vector, matrix, and solver abstract classes
- Users can supply their own data structures and solvers or use SUNDIALS supplied modules

Availability and support

- Used worldwide with more than *160,000 downloads* in 2022
- Available from the LLNL software site, GitHub, and Spack under the BSD 3-Clause license
- Extensive user documentation and user support email list with an archive on Google Groups

SUNDIALS Packages

The SUNDIALS library consists of six packages:

ARKODE: adaptive step, explicit, implicit, additive, and multirate Runge-Kutta methods for **ODEs**

$$M(t)\frac{dy}{dt} = f_1(t,y) + f_2(t,y), \qquad y(t_0) = y_0$$

CVODE(S): adaptive order and step implicit linear multistep methods for **ODEs**

$$\frac{dy}{dt} = f(t,y), \qquad y(t_0) = y_0$$

(S) Variant with forward and adjoint sensitivity analysis capabilities

IDA(S): adaptive order and step BDF (implicit linear multistep) methods for **DAEs**

$$F(t,y,y') = 0, \qquad y(t_0) = y_0, \qquad y'(t_0) = y_0'$$

KINSOL: Newton and accelerated Picard and fixed-point methods for **nonlinear systems**

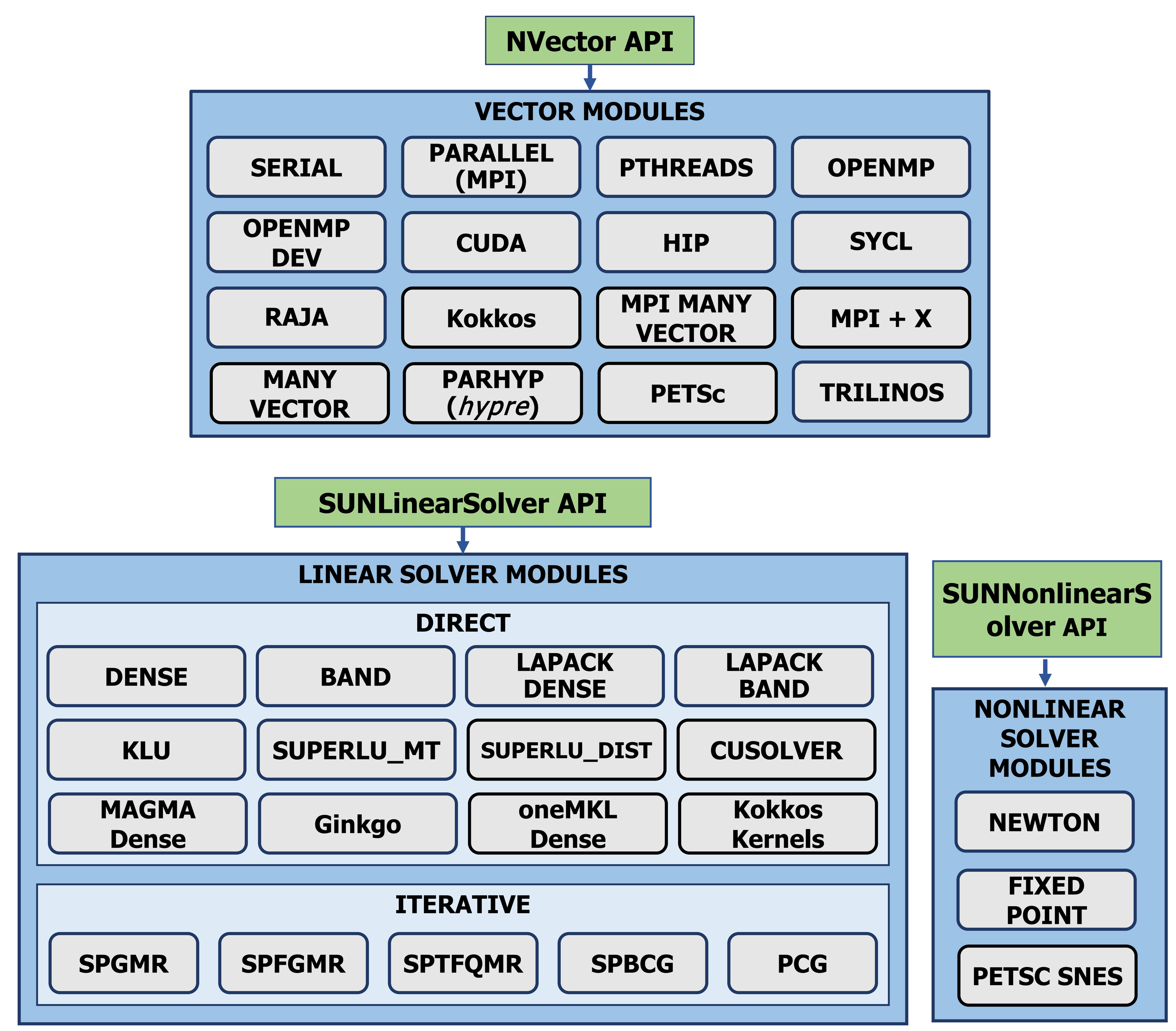
$$F(u) = 0 \qquad G(u) = u$$

Flexible Design and Data Structures

SUNDIALS packages are built on common APIs for vectors, matrices, and algebraic solvers making it straightforward to switch between options or provide application-specific data structures and optimized solvers.

The NVector, SUNMatrix, SUNLinearSolver, and SUNNonlinearSolver APIs define modules following the same object oriented design:

- Module-specific content structure is stored as a "black-box" ptr
- Module operations defined by API are implemented at user-level
- Function pointers to operations are stored in module ops struct
- Only a subset of operations are required for each package
- SUNDIALS supplies several optional module implementations, or users can supply their own
- User guides document the APIs



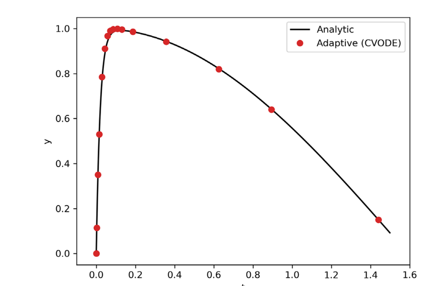
The choice of algebraic solvers used underneath SUNDIALS is critical to parallel scalability.

Adaptive Methods

SUNDIALS provides highly efficient integrators through use of adaptivity

- Estimate the time step error, E(Δt), using an embedded method of one lower order (RK) or direct error estimate (LMM)
- Accept step if $||E(\Delta t)||_{WRMS} < 1$; Reject it otherwise

$$\|y\|_{\mathrm{wrms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(w_i\ y_i)^2} \qquad w_i = \frac{1}{RTOL|y_i| + ATOL_i}$$



Adaptivity can give much more efficient (and accurate) results

- Choose next step so that $||E(\Delta t')||_{WRMS}$ is expected to be small
- CVODE(S) and IDA(S) also allow order adaption
- ARKODE supplies advanced "error controllers" which adapt steps to meet other criteria (e.g., minimize failed steps, smooth transitions in step sizes)

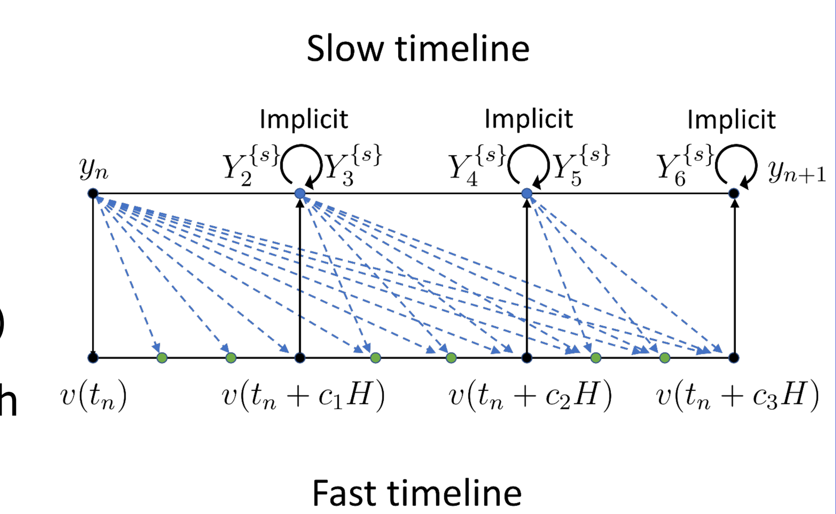
Multirate Methods

New SUNDIALS development has focused on flexible and high-order multirate methods that evolve different processes with different step sizes. For an IVP

$$\dot{y}(t) = f^I(t,y) + f^E(t,y) + f^F(t,y), \qquad t \in [t_0, t_f], \qquad y(t_0) = y_0.$$

a single step $y_n \rightarrow y_{n+1}$ of macroscale size $H = t_{n+1} - t_n$ proceeds as:

1. Let: $z_1 = y_n$
2. For each slow stage $z_i,\ i = 2, \dots, s$
   a. Define: $r_i(\tau) = \sum_{j=1}^{i} \gamma_{i,j}(\tau) f^I(t_n + c_j H, z_j) + \sum_{j=1}^{i-1} \omega_{i,j}(\tau) f^E(t_n + c_j H, z_j)$.
   b. Evolve: $\dot{v}(\tau) = f^F(\tau, v) + r_i(\tau),\ \tau \in [\tau_{0,i}, \tau_{f,i}],\ v(\tau_{0,i}) = z_{i-1}$.
   c. Let: $z_i = v(\tau_{f,i})$.
3. Let: $y_{n+1} = z_s$.



- Step 2b may use any applicable algorithm of sufficient accuracy (including another multirate method)
- Adaptivity may be performed for both H and for the fast IVPs.
- New support for ImEx at slow scale

For More Information

- https://computing.llnl.gov/projects/sundials
- Carol S. Woodward, LLNL, woodward6@llnl.gov
- Daniel R. Reynolds, SMU, reynolds@smu.edu

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and by Southern Methodist University under award DE-SC0021354. LLNL-POST-853941.

More FASTMath Information: http://scidac5-fastmath.lbl.gov

Argonne National Laboratory · Berkeley Lab · Lawrence Livermore National Laboratory · NREL National Renewable Energy Laboratory · Oak Ridge National Laboratory · Sandia National Laboratories · Massachusetts Institute of Technology · Rensselaer · SMU · USC University of Southern California