SUNDIALS: SUite of Nonlinear and DIfferential-ALgebraic Solvers

SUNDIALS Team: Carol S. Woodward, Daniel R. Reynolds, Alan C. Hindmarsh, and Lawrence E. Banks

SUNDIALS provides robust and efficient nonlinear solvers and time integrators for incorporation in large-scale scientific application codes

Overview of SUNDIALS

- Suite of time integrators and nonlinear solvers
  - ODE and DAE time integrators with forward and adjoint sensitivity capabilities, Newton-Krylov nonlinear solver
  - Written in C with interfaces to Fortran and Matlab
  - Designed to be incorporated into existing codes
- Modular implementation: users can supply own data structures
  - Vector structures – core data structure for all the codes
  - Linear solvers / preconditioners
  - Supplied with serial, distributed (MPI) and shared memory parallel (OpenMP & Pthreads) structures
- Freely available (BSD license)
- Contains six packages:
  - CVODE[1,3], CVODES[2,4], ARKode[1,3]
  - IDA[1,3], IDAS[2,4]
  - KINSOL[3,4]

[1] Rootfinding capabilities
[2] Forward and adjoint sensitivity analysis capabilities
[3] Fortran interfaces
[4] Matlab interfaces

Time Integrators

ODE Integration

- CVODE(S) – variable order, variable step linear multistep methods

$$\dot{y} = f(t,y) \qquad \sum_{j=0}^{K_1} \alpha_{n,j} y_{n-j} + \Delta t_n \sum_{j=0}^{K_2} \beta_{n,j} \dot{y}_{n-j} = 0$$

- ARKode – variable step, additive Runge Kutta multistage methods

$$M\dot{y} = f_E(t,y) + f_I(t,y)$$

$$Mz_i = My_{n-1} + h_n \sum_{j=0}^{i-1} A^E_{i,j} f_E(t_{n-1} + c_j h_n, z_j) + h_n \sum_{j=0}^{i} A^I_{i,j} f_I(t_{n-1} + c_j h_n, z_j),$$

$$My_n = My_{n-1} + h_n \sum_{i=0}^{s} b_i \left(f_E(t_{n-1} + c_i h_n, z_i) + f_I(t_{n-1} + c_i h_n, z_i) \right),$$

$$M\tilde{y}_n = My_{n-1} + h_n \sum_{i=0}^{s} \tilde{b}_i \left(f_E(t_{n-1} + c_i h_n, z_i) + f_I(t_{n-1} + c_i h_n, z_i) \right).$$

DAE Integration

- IDA(S) – variable order, variable step BDF (linear multistep) methods

$$F(t, \dot{y}, y) = 0$$
$$y(t_0) = y_0$$

Nonlinear solvers

KINSOL – nonlinear solvers (Newton & Fixed-point)

- Modified Newton method (serial and SMP applications)
- Inexact Newton-Krylov (serial, SMP and DMP applications)
- Picard and fixed-point with Anderson acceleration
- Krylov, sparse-direct, and dense/band linear solvers
- Globalization: linesearch, inexact Newton

Nonlinear solver components used within implicit time integrators

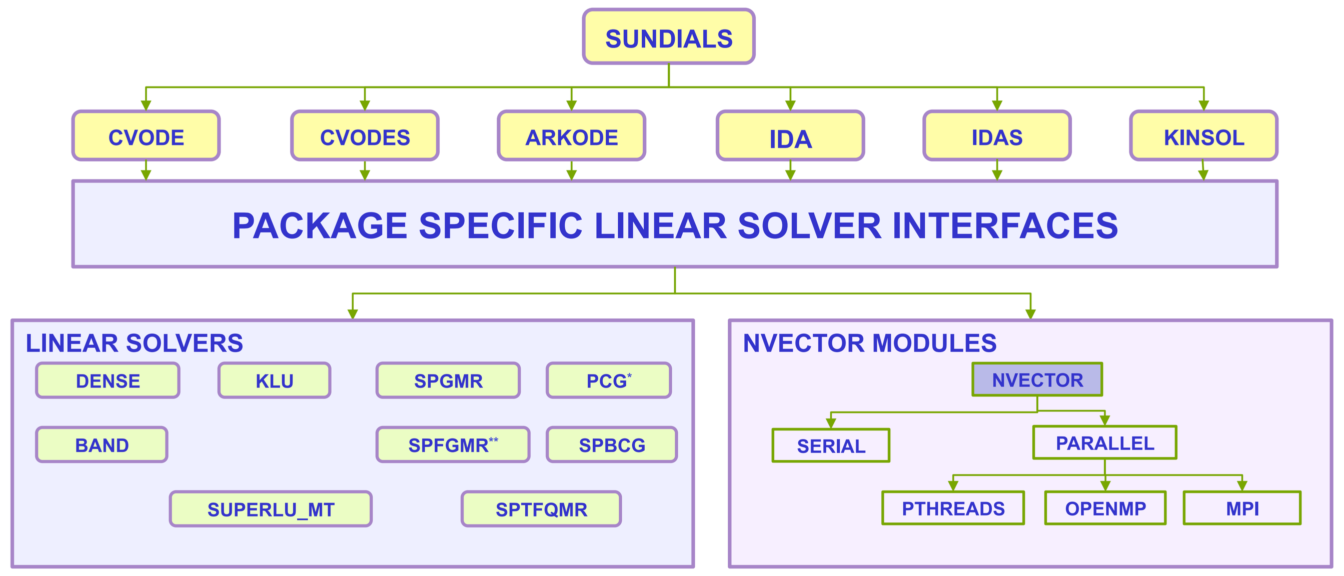
Flexible design – data structures/solvers

Serial, shared-memory and distributed-memory parallel vector structures are provided, or may be application specific. Requires:

- `Content` structure specifies the actual data and information needed to make new vectors (problem or grid data)
- Implementations of vector operations on supplied structure
- Routines to clone vectors for use within SUNDIALS

All parallelism resides in vector operations: dot products, norms, etc.

Solvers are reused between different SUNDIALS packages:



New Features

- ARKode package with explicit, implicit, and additive Runge-Kutta methods
- Interfaces for two sparse direct linear solvers, KLU and SuperLU_MT
- OpenMP and Pthreads vector kernels
- Fixed-point and Picard nonlinear solvers (optionally accelerated with Anderson acceleration) in KINSOL; FP/AA in ARKode
- FGMRES support in KINSOL and ARKODE; PCG support in ARKODE
- Removal of autotools configuration support. Now exclusively uses CMake

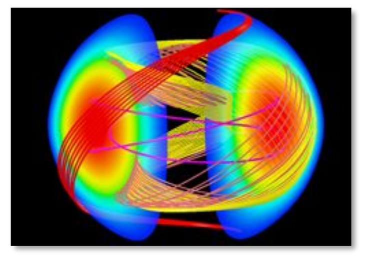
Application Skeleton (CVODE, GMRES, parallel)

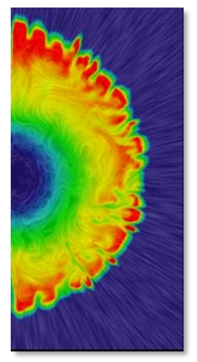
```
#include “cvode.h”
#include “cvode_spgmr.h”
#include “nvector_parallel.h”
…
y = N_VNew_Parallel(comm, local_n, NEQ);  // initialize vector
cvmem = CVodeCreate(CV_BDF,CV_NEWTON);    // create CVODE solver
flag = CVodeSet*(…);                      // set solver options
flag = CVodeInit(cvmem,rhs,t0,y,…);       // initialize solver
flag = CVSpgmr(cvmem,…);                  // pick linear solver
flag = CVSpilsSet*(cvmem, …);             // lin. solve options
for(tout = …) {                           // time-stepping loop
   flag = CVode(cvmem, …,y,…);  }
…
NV_Destroy(y);                            // free vector memory
CVodeFree(&cvmem);                        // free solver memory
```

Application Impact

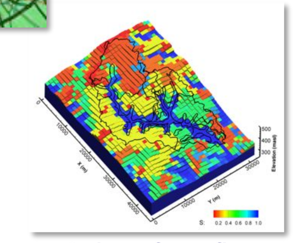
Used worldwide in applications from research and industry:

- Power grid modeling (RTE France, ISU)
- Simulation of clutches and power train parts (LuK GmbH & Co.)
- 3D parallel fusion (SMU, U. York, LLNL)
- Implicit hydrodynamics in core collapse supernovae (Stony Brook)
- Dislocation dynamics (LLNL)
- Sensitivity analysis of chemically reacting flows (Sandia)
- Large-scale subsurface flows (CO Mines, LLNL)
- Optimization in simulation of energy-producing algae (NREL)
- Micromagnetic simulations (U. Southampton)


Magnetic reconnection


Core collapse supernovae


Dislocation dynamics


Subsurface flow

>3,500 downloads/year

Future Plans

- Enhanced support for accelerators/many-core vector kernels
- Incorporation of *communication-avoiding* linear solvers
- Native interface to HYPRE for linear solvers
- Expansion of SUNDIALS/PETSc interfaces beyond CVODE
- Extension of ODE solvers to *multi-rate* problems (enabling components with varying step size)

https://computation.llnl.gov/casc/sundials

More Information: http://www.fastmath-scidac.org or contact Carol S. Woodward, LLNL, woodward6@llnl.gov, 925-424-6013