UCRL-JRNL-227722

SUNDIALS Equation Solvers

A. C. Hindmarsh, R. Serban

February 2, 2007

Scholarpedia

UCRL-JRNL-227722

6 February 2007

SUNDIALS Equation Solvers

Alan C. Hindmarsh and Radu Serban

Lawrence Livermore National Laboratory

Livermore, California

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

== Introduction ==

SUNDIALS, a SUite of Nonlinear and DIfferential/Algebraic equation Solvers, addresses three problem types:

* Ordinary Differential Equation (ODE) systems (initial value problems),

* Nonlinear algebraic systems of equations, and

* Differential-algebraic equation (DAE) systems (initial value problems).

All are written in ANSI C. For each problem type, there is a basic solver. For ODE and DAE systems, there are extensions that perform sensitivity analysis. All five solvers run in either a serial or a parallel machine setting. The implicit methods used lead to linear systems, for which a variety of direct and Krylov iterative methods are available. Two of the solvers include support for Fortran applications, and three include interfaces to Matlab. Full details (except for recent modifications) are available in Hindmarsh et al. (2005).

== CVODE, for ODE Systems ==

CVODE solves ODE initial value problems, in real N-space, written as

$$\dot{y} = f(t, y), \quad y(t_0) = y_0 \quad (y \in R^N).$$

The user first selects one of two variable-order, variable-step linear multistep method families – implicit Adams methods (up to order 12) or Backward Differentiation Formula (BDF) methods (up to order 5). Then the user specifies either functional or Newton iteration for the treatment of the implicit nonlinear equations, and in the Newton case, one of six algorithms (three direct, three Krylov) to solve the $N \times N$ linear systems that arise. These are:

* a dense direct solver (serial version only),

* a band direct solver (serial version only),

* a diagonal approximate Jacobian solver,

* Generalized Minimal Residual (GMRES) iteration,

* Bi-Conjugate Gradient Stable iteration (BiCGStab), and

* Transpose-Free Quasi-Minimal Residual (TFQMR) iteration.

For the dense and band solvers, the Jacobian matrix can be either supplied by the user or internally approximated. The three Krylov iterative methods are all scaled and include (left or right) preconditioning. For nonstiff systems, Adams method with functional iteration is sufficient. For stiff systems, characterized by at least one rapid decay mode (with time constant much smaller than the solution time scale), one must choose Newton iteration and a linear system solver that is appropriate to the problem. CVODE chooses stepsizes and method orders automatically and dynamically, so as to keep integration errors within user-supplied tolerances. The user may also specify that CVODE find (and stop at) the roots of given functions during the integration of the ODEs.

== KINSOL, for Nonlinear Algebraic Systems ==

KINSOL solves algebraic systems in real N-space, written as

$$F(u) = 0, \quad F : R^N \rightarrow R^N,$$

given an initial guess u_0. A modified or inexact Newton iteration is performed, using either a direct (dense or banded) solver (serial version only), or a right-preconditioned Krylov iterative solver (GMRES, BiCGStab, or TFQMR) for the linear systems. The user can choose full Newton corrections, or activate a linesearch globalization strategy. Also, the user can specify a scaling on the u vector and/or on the F vector, and tolerances on the steps δu and/or on $F(u)$, for use in the stopping tests. There are also three choices for the stopping tests in the Krylov iterations. KINSOL allows the user to enforce inequality constraints on each component of u – that it be positive, negative, non-negative, or non-positive.

== IDA, for DAE Systems ==

IDA solves real differential-algebraic systems in N-space, in the general form

$$F(t, y, \dot{y}) = 0, \quad y(t_0) = y_0, \quad \dot{y}(t_0) = \dot{y}_0.$$

If the supplied initial conditions are not consistent with the equations, IDA has an option to correct them, for some cases. The integration of the system uses variable-step variable-order BDF methods, up to order 5. In the Newton iteration at each step, the linear systems are solved by one of 5 methods – two direct (dense or band; serial version only) and three Krylov (GMRES, BiCGStab, or TFQMR). The direct methods allow either a user-supplied Jacobian or an internal approximation, and the Krylov methods include scaling and left preconditioning. As with CVODE, IDA chooses stepsizes and orders dynamically to control local errors according to user tolerances, and the user can have the integration stop at roots of given functions. As with KINSOL, the user may specify inequality constraints on the components of y.

== Preconditioning ==

The Krylov linear system solvers (GMRES, BiCGStab, and TFQMR) are "matrix-free", but to be effective they usually require preconditioning. A preconditioner matrix P is one that captures the dominant parts of the system Jacobian, but yet permits a reasonably economical solution of systems $Px = b$. All of the SUNDIALS solvers allow the user to supply preconditioning, in the form of two routines – one to preprocess P, and another to solve $Px = b$.

Alternatively, the user can call on one of two preconditioner modules included in SUNDIALS. The serial versions of CVODE and CVODES offer a preconditioner that generates banded difference-quotient approximations to the Jacobian. Also, in a parallel environment, all the solvers offer a block-diagonal preconditioner with banded blocks (one per processor).

== Sensitivity Analysis ==

When the problem-defining function (f or F), and possibly also the initial value vector(s), involve a set of parameters p, it is often desirable to compute the derivative with respect to p of the solution y (or of some output function of y). Code extensions CVODES (available now) and IDAS (in development) compute these sensitivities, using either of two approaches. In forward sensitivity analysis, an ODE or DAE for $\partial y/\partial p$ is generated and solved. In adjoint sensitivity analysis, more suitable when the number of parameters is large, the system is integrated forward and the solution saved at selected checkpoints, and another auxiliary equation (the adjoint system) is generated and integrated backwards (along with partial forward integrations between checkpoints). Then the desired sensitivities are computed in terms of the adjoint solution using a quadrature. The codes offer various options for the way in which the auxiliary equations are generated, and their solutions advanced along with the state vector y.

== Software Features ==

SUNDIALS was designed in a modular manner with careful attention to flexibility and avoidance of duplication. For example, code modules for the various generic linear solvers (direct and Krylov), which are independent of the top-level solvers, are accessed through interface functions that are specific to the various solvers. In addition, for each top-level solver, these interfaces conform to a standard which permits a user to supply his/her own linear solver module.

Operations on N-vectors (linear sums, dot products, norms, etc.) are also isolated in an NVECTOR module. The generic NVECTOR module defines abstract vector operations, and links to an actual NVECTOR implementation. The latter can be the serial or parallel (MPI) implementation supplied with SUNDIALS, or one supplied by the user. The rest of SUNDIALS is independent of the NVECTOR implementation.

The user interface to the SUNDIALS solvers consists of various function calls that initialize modules, allocate memory, specify user functions, specify required and optional input parmameters, compute and obtain solutions, get optional outputs, and free memory. When there is a default for an optional input, the user can either do nothing or call a Set function to supply a non-default value. Similar Get functions return optional outputs to the user.

For the CVODE, KINSOL, and IDA solvers, SUNDIALS includes modules that interface with Fortran applications. Thus a user's Fortran program can call routines that interface to one of these solvers, and can supply Fortran subroutines that are called by the solver through its interfaces.

== Reference ==

* Hindmarsh, A. C., et al., SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Trans. Math. Softw., 31:363-396, 2005.

== External Link ==

* http://www.llnl.gov/CASC/sundials