ABACUSS Project Report **No. 97/1**

DSL48S
A Large-Scale Differential-Algebraic and Parametric Sensitivity Solver

Paul I. Barton, Russell J. Allgor and William F. Feehery
Department of Chemical Engineering
Massachusetts Institute of Technology
Cambridge MA 02139

22nd January 1997

DSL48S is a package of FORTRAN 77 subroutines that can be called by a user provided main program to obtain numerical solutions to Initial Value Problems (IVPs) in Differential-Algebraic Equations (DAEs). The code works with DAEs in the general nonlinear form:

$$f(\dot{y}, y, t) = 0 \tag{1}$$

where $y \in \mathbf{R}^n$ are the state variables, $\dot{y} \in \mathbf{R}^n$ are the first time derivatives of y with respect to the independent variable time t, and $f : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R} \rightarrow \mathbf{R}^n$. DAEs are distinguished from Ordinary Differential Equations by the fact that the Jacobian matrix $\partial f/\partial \dot{y}$ is singular everywhere. Given consistent initial values, $y(0)$ and $\dot{y}(0)$, the code will calculate a numerical approximation to the solution $y(t)$ of (1) over a specified time interval $(0, t_f]$. This approximation will satisfy a user specified truncation error tolerance.

DSL48S is particularly suitable for large-scale problems (e.g., n = 1,000–100,000+). In order to set up a problem, the user must provide:

- a main program that calls DSL48S and co-ordinates solution of the problem the user wants to solve.
- a subroutine that will calculate the vector of values for f given values for y, $\dot{y}$ and t (often known as a residual evaluator).
- the pattern of nonzero elements in the Jacobian matrices $\partial f/\partial y$ and $\partial f/\partial \dot{y}$.
- an *optional* subroutine that can return analytical values for some or all of the nonzero elements of the Jacobian matrices $\partial f/\partial y$ and $\partial f/\partial \dot{y}$ given values for y, $\dot{y}$ and t (often known as a Jacobian evaluator).

In addition, if the DAEs are stated in terms of a vector of time invariant parameters $v \in \mathbf{R}^p$:

$$f(\dot{y}, y, v, t) = 0 \tag{2}$$

it is possible to calculate the parametric sensitivity trajectories simultaneously with the state trajectories. The parametric sensitivities are defined by:

$$\frac{\partial f}{\partial \dot{y}}\dot{s}_i + \frac{\partial f}{\partial y}s_i + \frac{\partial f}{\partial v_i} = 0 \qquad \forall i = 1 \ldots p \tag{3}$$

where $s_i \equiv \partial y / \partial v_i$. Residuals for the sensitivity equations (3) can be evaluated either a) numerically using a directional derivative finite difference approximation [1], or b) analytically. The analytical option requires an extra user supplied routine, but has been shown to be more efficient [2].

It is also important to be aware of what DSL48S *cannot* do, which includes:

- DSL48S cannot calculate consistent initial values $\{y(0), \dot{y}(0)\}$. DSL48S requires the initial values $\{y(0), \dot{y}(0)\}$ to be consistent within a reasonable tolerance. Further, DSL48S cannot check if these initial values are consistent: if inconsistent values are passed to DSL48S, the code will either fail, or jump to an arbitrary consistent initial point and solve an IVP from this point. See [3] for a detailed discussion of the pitfalls associated with passing a code like DSL48S inconsistent initial values.
- DSL48S cannot solve DAEs with differential index [4] greater than unity. Further, since DSL48S is unable to check the differential index of a system, if passed a system with differential index greater than unity, the code will either fail unpredictably, or calculate results that are absolute garbage.
- DSL48S will not respond well to functions f containing discontinuities. It is recommended that discontinuous functions f be handled explicitly in the user provided main program that calls DSL48S as a subproblem. A modern approach to discontinuity handling is described in [5].

DSL48S is a derivative work of the public domain code DASSL, developed by Linda Petzold [6]. As such, it shares with DASSL the entire basic numerical integration algorithm, with its *excellent* heuristics. The most significant additional features of DSL48S created by work at MIT include:

- the original linear algebra routines have been removed and replaced by the sparse unsymmetric linear equation solver MA48 [7], developed by I.S. Duff and J.K. Reid. MA48 is marketed by AEA Technology in the UK, and MIT has obtained permission to distribute code with MA48 embedded provided the licensee obtains a MA48 license from AEA Technology. This feature enables DSL48S to solve very large-scale problems involving sparse, unstructured equation systems. DSL48S has currently been tested on problems with upwards of 60,000 simultaneous equations. Our numerical experiments comparing the older MA28 with MA48 indicate that on average, a 20–30% reduction in solution times can be achieved through the use of MA48. Discussions with colleagues who have solved really large problems with MA48 (100,000+ equations) indicate that this advantage becomes much larger as problem size increases. A further useful feature of MA48 is that is will automatically decompose solution of the linear system into a sequence of smaller linear systems (known as *block decomposition*) if this is possible. We have found that this is the most effective strategy for exploiting any block triangular structure of the original DAE system (1).

  Please note that the original linear equation solvers embedded in DASSL have been removed and are not supported in DLS48S. However, MA48 will analyze a particular problem and will revert to a dense linear solver if it judges that this is a more appropriate strategy.
- parametric sensitivities are calculated by a novel *staggered corrector* algorithm developed at MIT [2]. The algorithm embedded in DSL48S is more efficient than all other approaches currently reported in the literature [2]. Further, the advantages of our approach are particularly noticeable for large sparse equation systems. There are also a number of useful options that control how the sensitivities are calculated that can greatly improve the efficiency of certain applications.

- an algorithm that will guarantee that the solution trajectory lies within bounds on the state variables specified by the user.
- the ability to use a sparse hybrid Jacobian for the corrector iteration. This is a Jacobian matrix for which a subset of the nonzero elements are calculated analytically using a user supplied subroutine, and the remainder of the elements are calculated using numerical finite differences. Further, the number of residual evaluations required for the finite differences is minimized through use of a special algorithm [8, 9].
- modifications to allow use of DSL48S as part of a code to solve a broad class of high index DAEs [10]. However, in itself DSL48S *cannot* solve high index DAEs.
- the option to use a novel algorithm developed at MIT that will automatically scale the matrix used in the corrector iteration [11, 12]. This option is recommended by default because: a) if the corrector matrix is ill-conditioned, the scaling will mitigate the numerical problems that may occur, b) in general, across the board, empirical evidence indicates that the scaling leads to shorter solution times, and c) the code will automatically warn the user if the problem is so poorly conditioned that it cannot be solved to the specified tolerance with the current machine precision.
- the option for DSL48S to use a novel algorithm developed at MIT [13] to calculate an 'optimal' initial step size for the numerical integration. This is based on information from a consistent initialization calculation, which must be passed to DSL48S by the user. This feature is particularly useful for combined discrete/continuous (or hybrid) simulations with frequent discontinuities, which require frequent restarts of the numerical integrator. The default initial step size selection heuristics tend to be overly conservative, leading to many small steps following a discontinuity before confidence in the solution is built up again, which slows down the code considerably if discontinuities occur frequently. Our algorithm is an attempt to select an initial step size that is as large as possible considering the state of excitation of the dynamic system following a discontinuity, while avoiding a truncation error test failure on the first step.

MIT is pleased to offer DSL48S licenses to both industry and academia. Technical queries concerning DSL48S should be directed to:

Prof. Paul I. Barton
Hoyt C. Hottel Assistant Professor
Department of Chemical Engineering
Massachusetts Institute of Technology 66–464
77 Massachusetts Avenue
Cambridge
MA 02139
Phone: +1–617–253–6526
Fax: +1–617–258-5042
e-mail: pib@mit.edu

Licensing and legal queries should be directed to:

Mr. Michael J. Hegarty
Technology Licensing Office
Massachusetts Institute of Technology NE25–230
77 Massachusetts Avenue

Cambridge
MA 02139
Phone: +1–617–253–6966
e-mail: mhegarty@mit.edu

References

[1] T. Maly and L. R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Appl. Num. Math.*, 20:57–79, 1996.

[2] W. F. Feehery, J. E. Tolsma, and P. I. Barton. Efficient sensitivity analysis of large-scale differential-algebraic systems. *Appl. Num. Math.*, 1997. in press.

[3] A. Kröner, W. Marquardt, and E. D. Gilles. Computing consistent initial values for differential-algebraic equations. *Computers chem. Engng*, 16(S):131–138, 1992.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. North-Holland, 1989.

[5] T. Park and P. I. Barton. State event location in differential-algebraic models. *ACM Trans. Mod. Comput. Sim.*, 6(2):137–165, 1996.

[6] L. Petzold. A description of DASSL: a differential-algebraic equation solver. In *Proceedings 10th IMACS World Congress*, Montreal, Canada, 1982.

[7] I. S. Duff and J. K. Reid. MA48, a Fortran code for direct solution of sparse unsymmetric linear systems of equations. Technical Report RAL–930–072, Rutherford Appleton Laboratory, October 1993.

[8] T.F. Coleman, B. S. Garbow, and J. J. Moré. Software for estimating sparse Jacobian matrices. *ACM Trans. Math. Soft.*, 10(3):329–345, 1984.

[9] T.F. Coleman, B. S. Garbow, and J. J. Moré. Algorithm 618 FORTRAN subroutines for estimating sparse Jacobian matrices. *ACM Trans. Math. Soft.*, 10(3):346–347, 1984.

[10] W. F. Feehery and P. I. Barton. A differentiation-based approach to simulation and dynamic optimization with high-index differential-algebraic equations. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation Techniques, Applications, and Tools*. SIAM, Philadelphia, 1996.

[11] R. J. Allgor and P. I. Barton. Accurate solution of batch distillation models: Implications of ill-conditioned corrector iterations. In *AIChE Annual Meeting*, Miami, Floirda, 1995.

[12] R. J. Allgor and P. I. Barton. Automated scaling of differential-algebraic equations. *Applied Numerical Mathematics*, 1997. in preparation for submission.

[13] R. J. Allgor and P. I. Barton. Initial step size selection for differential-algebraic systems. *ACM Transactions on Modeling and Computer Simulation*, 1997. submitted.