DAEPACK

Supporting Libraries for the DAEPACK Numerical Components

Version 1.0

John E. Tolsma

December 2, 2000

Contents

This manual describes the additional supporting routines that must be linked into an application using the DAEPACK Numerical components. Some of these libraries use public domain code and are distributed with DAEPACK. Others, namely the libraries containing Harwell code, require the user to have a valid license in order to use the code. Consequently, it is left to the user to build these libraries prior to using DAEPACK. Finally, DAEPACK numerical solvers that use the Harwell routines to exploit sparsity are also provided in versions where the calls to Harwell code are replaced by dense linear algebra performed by similar Linpack routines. This allows the user without access to the Harwell code to use the DAEPACK components. However, this dramatically limits the size of problems that may be solved and reduces the performance on many other problems.

1 Overview

Most of the DAEPACK numerical components depend on additional supporting code for various calculations. This code, described in detail below, should be linked into the application using DAEPACK and the user is free to select how the additional code is made available to DAEPACK (e.g., linked in as object code or shared or static libraries). Much of this additional code is public domain and these routines are distributed in separate libraries with DAEPACK. However, some of the code, namely certain Harwell Library routines, require the user to have a valid license. Consequently, it is left to the user to build these libraries prior to using DAEPACK.

Four additional numerical libraries are required when using the DAEPACK numerical components BLKSLV [6], DSL48E [5], and DSL48S [4]. The four additional libraries are:

1. BLAS, containing certain basic linear algebra subroutines,
2. DSM, containing algorithms for determining structural orthogonal columns of the Jacobian matrix,
3. HARWELL, containing Harwell library sparse linear algebra, structural, and solver subroutines, and
4. LINPACK, containing certain linear algebra routines from the Linpack library.

Table 1 summarizes which supporting libraries are required by each of the DAEPACK numerical packages.

Table 1: Dependencies of DAEPACK numerical components with additional supporting libraries.

DAEPACK Package	Depends on Supporting Libraries
BLKSLV	BLAS, HARWELL, and LINPACK
BLKSLV (dense)	BLAS and LINPACK
DSL48S	BLAS, DSM, and HARWELL
DSL48S (dense)	BLAS, DSM, and LINPACK
DSL48E	DSL48S, BLAS, DSM, and HARWELL
DSL48E (dense)	DSL48S, BLAS, DSM, and LINPACK

2 Basic Linear Algebra Subroutines (BLAS)

Most computer platforms have tailored basic linear algebra subroutine (BLAS) libraries available. Consequently, the DAEPACK numerical codes exploit these subroutines wherever possible. If the BLAS library is available on the machine where DAEPACK is to be used, then it should be linked with DAEPACK into the main application (e.g., with the compiler option `-lblas`). If this library is not available, then DAEPACK provides a BLAS library containing only the subroutines actually used: `dcopy`, `dscal`, `daxpy`, and `dswap`. The BLAS library is named `libblas.sl` (shared library version) and `libblas.a` (static library version) and should be linked into the application using DAEPACK. Obviously, since these routines are not tailored to the specific architecture, the native library is preferred. This library is required for all of the DAEPACK numerical components.

3 Structurally Orthogonal Columns

DAEPACK component DSL48S will exploit Jacobian structure when evaluating partial derivatives with finite differences by partitioning the columns of the Jacobian into structurally orthogonal groups. This will, for many large sparse problems, substantially reduce the work required to evaluate the Jacobian (however, it is recommended to use the analytical derivatives provided by the DAEPACK symbolic components [3] instead of finite differences whenever possible). DSL48S uses the code DSM described in [2, 1] for this task. This code is distributed with DAEPACK as a shared library (`libdsm.sl`) and a static library (`libdsm.a`), which must be linked into applications using DSL48S directly or indirectly through DSL48E.

4 Sparse Linear Algebra and Structural Analysis

The DAEPACK symbolic library contains a component for generating automatically new Fortran source code for determining the sparsity pattern of a model from the original Fortran source [3]. This capability allows sparsity to be exploited for a wide variety of applications, including the algorithms in BLKSLV, DSL48E, and DSL48S. In particular, these DAEPACK components use subroutines in the Harwell library for structural analysis and linear algebra. Since this library requires a license, we have chosen to distribute DAEPACK without these subroutines, leaving it to the user to obtain the Harwell license and build the library (this simplifies the distribution and lowers the cost of DAEPACK). The following Harwell subroutines are required:

DGEMM	DGEMV	DTRSM	DTRSV	MA48AD	MA48BD	MA48CD
MA48DD	MA48ID	MA50AD	MA50BD	MA50CD	MA50DD	MA50ED
MA50FD	MA50GD	MA50HD	MA50ID	MC13D	MC13E	MC21A
MC21AD	MC21B	MC21BD	MC23AD	MC29AD	MC41AD	XERBLA

These subroutines must be available to an application using DAEPACK components BLKSLV, DSL48E, and DSL48S. The Harwell library distribution provides facilities allowing the user to create a library. However, if the user simply has the Fortran source then the library can be readily constructed with a Fortran compiler and the `ar` or `ld` command on UNIX/Linux. Similar options are available on Windows 98/NT. Alternatively, the Harwell subroutines may be compiled and linked into the application as object code.

As mentioned above, the DAEPACK components using Harwell code are also provided in versions that use dense linear algebra (with Linpack code) instead. In this case, the user need not have the Harwell library and can simply link in the Linpack library described below. However, significant performance increases can be achieved by exploiting sparsity and much larger problems can be solved. Moreover, the symbolic components of DAEPACK generate the additional code required to exploit sparsity. Thus, we recommend that a Harwell license be obtained to maximize the benefits of using DAEPACK.

5 Other Harwell Library Routines

In addition to the subroutines described in the section above, two other Harwell subroutines are required by DAEPACK. Subroutine `LA01BD` is a linear programming (LP) solver used by the DAEPACK component SLPSLV (potentially called from BLKSLV), the DAEPACK implementation of the *successive linear programming* (SLP) approach for solving nonlinear systems of equations. Subroutine `NB01AD` implements

the *bisection* algorithm for solving single equations and is used in both the DAEPACK BSCSLV (potentially called from BLKSLV) and DSL48E components. Both `LA01AD` and `NB01AD` should be compiled into a shared or static library and linked into applications using BLKSLV and/or DSL48E.

6 LINPACK

When using BLKSLV to solve a large sparse system of equations, some (or all) of the blocks may small enough so that no benefit is gained from using sparse linear algebra. Sparse linear algebra is also not appropriate if the block is not sufficiently sparse. In these situations, BLKSLV will use Linpack linear algebra subroutines to solve the linear systems that arise during Newton's method. The DAEPACK distribution contains a library named `liblinpack.sl` (shared library version) and `liblinpack.a` (static library version) which provides the Linpack subroutine `dgefa` for matrix factorization and subroutine `dgesl` for the subsequent backsubstitution. This library must also be linked into applications using the versions of the DAEPACK components that do not exploit sparsity.

7 Conclusion

Several additional libraries must be linked into applications using the DAEPACK numerical components. These supporting libraries are provided with the DAEPACK distribution with the exception of libraries containing Harwell code, which must be provided by the user. Additional information can be found on the DAEPACK website `yoric.mit.edu/daepack/daepack.html`.

References

[1] T. F. Coleman, B. S. Garbow, and J. J. Moré, *Fortran subroutines for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 346–347.

[2] ——, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 329–345.

[3] J. E. Tolsma, *DAEPACK code generation manual*, Tech. Rep. DAEPACK Documentation. Version 1.0, Massachusetts Institute of Technology, 2000. http://yoric.mit.edu/daepack/daepack.html.

[4] ——, *Large-scale numerical integration and parametric sensitivity analysis of DAEs. DSL48S manual*, Tech. Rep. DAEPACK Documentation. Version 1.0, Massachusetts Institute of Technology, 2000. http://yoric.mit.edu/daepack/daepack.html.

[5] ——, *Numerical integration with robust state event location. DSL48E manual*, Tech. Rep. DAEPACK Documentation. Version 1.0, Massachusetts Institute of Technology, 2000. http://yoric.mit.edu/daepack/daepack.html.

[6] ——, *Solution of large-scale sparse nonlinear algebraic systems of equations. Block Solver manual*, Tech. Rep. DAEPACK Documentation. Version 1.0, Massachusetts Institute of Technology, 2000. http://yoric.mit.edu/daepack/daepack.html.