


GALAHAD SILS

USER DOCUMENTATION GALAHAD Optimization Library version 5.0

1 SUMMARY

This package **solves a sparse symmetric system of linear equations.** Given a sparse symmetric matrix $\mathbf{A} = \{a_{ij}\}_{n\times n}$, and an n-vector $\mathbf{b}$ or a matrix $\mathbf{B} = \{b_{ij}\}_{n\times r}$, this subroutine solves the system $\mathbf{Ax} = \mathbf{b}$ or the system $\mathbf{AX} = \mathbf{B}$. The matrix $\mathbf{A}$ need not be definite. There is an option for iterative refinement.

The method used is a direct method based on a sparse variant of Gaussian elimination and is discussed further by Duff and Reid, *ACM Trans. Math. Software* **9** (1983), 302-325.

ATTRIBUTES — Versions: `GALAHAD_SILS_single`, `GALAHAD_SILS_double`. **Remark:** `GALAHAD_SILS` is a Fortran 90 encapsulation of the HSL Fortran 77 package `MA27`, that offers some additional facilities. The user interface is designed to be equivalent to the more recent HSL package `HSL_MA57`, so the two packages may be used interchangeably. **Uses:** `GALAHAD_SMT`, `MA27`. **Date:** April 2001. **Origin:** Interface by N. I. M. Gould, Rutherford Appleton Laboratory, documentation follows that of I.S. Duff and J.K. Reid, ibid. **Language:** Fortran 95 + TR 15581 or Fortran 2003.

2 HOW TO USE THE PACKAGE

The package is available using both single and double precision reals, and either 32-bit or 64-bit integers. Access to the 32-bit integer, single precision version requires the `USE` statement

```
USE GALAHAD_SILS_single
```

with the obvious substitution `GALAHAD_SILS_double`, `GALAHAD_SILS_single_64` and `GALAHAD_SILS_double_64` for the other variants.

If it is required to use more than one of the modules at the same time, the derived types `SMT_TYPE`, `SILS_CONTROL`, `SILS_AINFO`, `SILS_FINFO`, `SILS_SINFO`, and `SILS_FACTORS`, (Section 2.2) and the subroutines `SILS_INITIALIZE`, `SILS_ANALYSE`, `SILS_FACTORIZE`, `SILS_SOLVE`, `SILS_FINALIZE`, (Section 2.3) `SILS_ENQUIRE`, `SILS_ALTER_D`, and `SILS_PART_SOLVE` (Section 2.5) must be renamed on one of the `USE` statements.

There are five principal subroutines for user calls (see Section 2.5 for further features):

The subroutine `SILS_INITIALIZE` must be called to initialize the structure for the factors. It may also be called to set default values for the components of the control structure. If non-default values are wanted for any of the control components, the corresponding components should be altered after the call to `SILS_INITIALIZE`.

`SILS_ANALYSE` accepts the pattern of $\mathbf{A}$ and chooses pivots for Gaussian elimination using a selection criterion to preserve sparsity. It subsequently constructs subsidiary information for actual factorization by `SILS_FACTORIZE`. The user may provide the pivot sequence, in which case only the necessary information for `SILS_FACTORIZE` will be generated.

`SILS_FACTORIZE` factorizes a matrix $\mathbf{A}$ using the information from a previous call to `SILS_ANALYSE`. The actual pivot sequence used may differ from that of `SILS_ANALYSE` if $\mathbf{A}$ is not definite.

`SILS_SOLVE` uses the factors generated by `SILS_FACTORIZE` to solve a system of equations with one ($\mathbf{Ax} = \mathbf{b}$) or several ($\mathbf{AX} = \mathbf{B}$) right-hand sides, or to improve a given solution or set of solutions by iterative refinement.

`SILS_FINALIZE` reallocates the arrays held inside the structure for the factors to have size zero. It should be called when all the systems involving its matrix have been solved unless the structure is about to be used for the factors of another matrix.

2.1 Real and integer kinds

We use the terms integer and real to refer to the fortran keywords REAL(rp_) and INTEGER(ip_), where rp_ and ip_ are the relevant kind values for the real and integer types employed by the particular module in use. The former are equivalent to default REAL for the single precision versions and DOUBLE PRECISION for the double precision cases, and correspond to rp_ = real32 and rp_ = real64, respectively, as supplied by the fortran iso_fortran_env module. The latter are default (32-bit) and long (64-bit) integers, and correspond to ip_ = int32 and ip_ = int64, respectively, again from the iso_fortran_env module.

2.2 The derived data types

Six derived data types are accessible from the package.

2.2.1 The derived data type for holding the matrix

The derived data type SMT_TYPE is used to hold the matrix **A**. The components of SMT_TYPE are:

N is a scalar variable of type INTEGER(ip_), that holds the order n of the matrix **A**. **Restriction:** N $\geq$ 1.

NE is a scalar variable of type INTEGER(ip_), that holds the number of matrix entries. **Restriction:** NE $\geq$ 0.

VAL is a rank-one allocatable array of type REAL(rp_), and dimension at least NE, that holds the values of the entries. Each pair of off-diagonal entries $a_{ij} = a_{ji}$ is represented as a single entry. Duplicated entries are summed.

ROW is a rank-one allocatable array of type INTEGER(ip_), and dimension at least NE, that holds the row indices of the entries.

COL is a rank-one allocatable array of type INTEGER(ip_), and dimension at least NE, that holds the column indices of the entries.

2.2.2 The derived data type for holding control parameters

The derived data type SILS_CONTROL is used to hold controlling data. Default values may be obtained by calling SILS_initialize (see Section 2.3.1). The components of SILS_CONTROL are:

LP is an INTEGER(ip_) scalar used by the subroutines as the output stream for error messages. If it is negative, these messages will be suppressed. The default value is 6.

WP is an INTEGER(ip_) scalar used by the subroutines as the output stream for warning messages. If it is negative, these messages will be suppressed. The default value is 6.

MP is an INTEGER(ip_) scalar used by the subroutines as the output stream for diagnostic printing. If it is negative, these messages will be suppressed. The default value is 6.

SP is an INTEGER(ip_) scalar used by the subroutines as the output stream for statistics. If it is negative, these messages will be suppressed. The default value is -1.

LDIAG is an INTEGER(ip_) scalar used by the subroutines to control diagnostic printing. If LDIAG is less than 1, no messages will output. If the value is 1, only error messages will be printed. If the value is 2, then error and warning messages will be printed. If the value is 3, scalar data and a few entries of array data on entry and exit from each subroutine will be printed. If the value is greater than 3, all data will be printed on entry and exit. The default value is 2.

LA is an INTEGER(ip_) scalar used by SILS_FACTORIZE. If LA $\geq$ NRLNEC (see Section 2.2.3), the real array that holds data for the factors is reallocated to have size LA. Otherwise, the array is not reallocated unless its size is less than NRLNEC, in which case it is reallocated with size NRLTOT (see Section 2.2.3). The default value is 0.

LIW is an INTEGER(ip_) scalar used by SILS_FACTORIZE. If LIW $\geq$ NIRNEC (see Section 2.2.3), the integer array that holds data for the factors is reallocated to have size LIW. Otherwise, the array is not reallocated unless its size is less than NIRNEC, in which case it is reallocated with size NIRTOT (see Section 2.2.3). The default value is 0.

MAXLA is an INTEGER(ip_) scalar used by SILS_FACTORIZE. An error return occurs if the real array that holds data for the factors is too small and reallocating it to have size changed by the factor MULTIPLIER would make its size greater than MAXLA. The default value is HUGE(0).

MAXLIW is an INTEGER(ip_) scalar used by SILS_FACTORIZE. An error return occurs if the integer array that holds data for the factors is too small and reallocating it to have size changed by the factor MULTIPLIER would make its size greater than MAXLIW. The default value is HUGE(0).

MULTIPLIER is a REAL(rp_) scalar used by SILS_FACTORIZE when a real or integer array that holds data for the factors is too small. The array is reallocated with its size changed by the factor MULTIPLIER. The default value is 2.0.

REDUCE is a REAL(rp_) scalar that reduces the size of previously allocated internal workspace arrays if they are larger than currently required by a factor of REDUCE or more. The default value is 2.0.

NEMIN is an INTEGER(ip_) scalar used by SILS_ANALYSE for the minimum number of eliminations in a step that is automatically accepted. If two adjacent steps can be combined and each has fewer eliminations, they are combined. The default value is 1.

THRESH is an INTEGER(ip_) scalar used by SILS_ANALYSE to identify dense rows during pivot selection. It is the percentage density for a row to be regarded as dense. The default value is 100.

ORDERING is a scalar variable of type INTEGER(ip_), that controls the initial order of the rows when performing the factorization. Possible values are:

- 0 The ordering will be chosen by the Approximate Minimum Degree method without provisions for "dense" rows/columns.
- 1 The ordering specified in the argument PERM for SILS_analyse will be used.
- 2 The ordering will be chosen by the Approximate Minimum Degree method with provisions for "dense" rows/columns.
- 3 The ordering will be chosen by the Minimum Degree method.
- 4 The ordering will be chosen by the Nested Dissection method; this requires the user to have installed the external package METIS.
- 5 Ordering 4 will be used unless it is unavailable in which case ordering 2 will be chosen.

The default value is 5, and any inappropriate choice will be reset to this default.

PIVOTING is an INTEGER(ip_) scalar that is used to control numerical pivoting by SILS_FACTORIZE. It must have one of the following values:

- 1 Numerical pivoting will be performed, with relative pivot tolerance given by the component U.
- 2 No pivoting will be performed and an error exit will occur immediately a sign change is detected among the pivots. This is suitable for cases when **A** is thought to be definite and is likely to decrease the factorization time while still providing a stable decomposition.

3 No pivoting will be performed and an error exit will occur if a zero pivot is detected. This is likely to decrease the factorization time, but may be unstable if there is a sign change among the pivots.

4 No pivoting will be performed but the matrix will be altered if a non-positive pivot is encountered.

The default value is 1.

U is a REAL(rp_) scalar that is used by SILS_FACTORIZE when the component PIVOTING has the value 1 to hold the relative pivot tolerance. The default value is 0.01. For problems requiring greater than average numerical care a higher value than the default would be advisable. Values greater than 0.5 are treated as 0.5 and less than 0.0 as 0.0.

TOLERANCE is a REAL(rp_) scalar that is used by SILS_FACTORIZE. Any entry of modulus less than or equal to TOLERANCE is treated as zero. The default value is 0.0

FACTORBLOCKING is an INTEGER(ip_) scalar used by SILS_FACTORIZE to determine the block size used for the Level 3 BLAS. The default value is 16.

SOLVEBLOCKING is an INTEGER(ip_) scalar used by SILS_SOLVE to determine the block size used for the Level 2 and Level 3 BLAS. The default value is 16.

CONVERGENCE is a REAL(rp_) scalar that is used by SILS_SOLVE. This is used to used to monitor convergence iterative refinement. If the norm of the scaled residuals does not decrease by a factor of at least CONVERGENCE, convergence is deemed to be too slow and the solution phase is terminated. The default value is 0.5.

2.2.3 The derived data type for holding informational parameters from SILS_ANALYSE

The derived data type SILS_AINFO is used to hold parameters that give information about the progress of the SILS_ANALYSE subroutine. The components of SILS_AINFO are:

FLAG is an INTEGER(ip_) scalar. The value zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.1.

MORE is an INTEGER(ip_) scalar that provides further information in the case of an error, see Section 2.4.1.

OOR is an INTEGER(ip_) scalar that is set to the number of entries with one or both indices out of range.

DUP is an INTEGER(ip_) scalar that is set to the number of duplicate off-diagonal entries.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

NSTEPS is an INTEGER(ip_) scalar that is set to the number of nodes in the assembly tree (number of major steps in the factorization).

MAXFRT is an INTEGER(ip_) scalar that holds the largest front size.

OPSA is a REAL(rp_) scalar that is set to the number of floating-point additions required by the assembly of frontal matrices if no pivoting is performed. Numerical pivoting may increase the number of operations.

OPSE is a REAL(rp_) scalar that is set to the number of floating-point operations required by the factorization if no pivoting is performed. Numerical pivoting may increase the number of operations.

NRLTOT and NIRTOT are INTEGER(ip_) scalars that give the total amount of real and integer words respectively required for a successful factorization without the need for data compression, provided no numerical pivoting is performed.

NRLNEC and NIRNEC are INTEGER(ip_) scalars that give the total amount of real and integer words required respectively for successful factorization allowing data compression, provided no numerical pivoting is performed.

NRLADU and NIRADU are INTEGER(ip_) scalars that give the number of real and integer words required respectively to hold the matrix factors if no numerical pivoting is performed.

NCMPA is an INTEGER(ip_) scalar that holds the number of compresses of the internal data structure performed by SILS_ANALYSE.

2.2.4 The derived data type for holding informational parameters from SILS_FACTORIZE

The derived data type SILS_FINFO is used to hold parameters that give information about the progress of the SILS_FACTORIZE subroutine. The components of SILS_FINFO are:

FLAG is an INTEGER(ip_) scalar. The value of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.2.

MORE is an INTEGER(ip_) scalar that provides further information in the case of an error, see Section 2.4.2.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

MAXFRT is an INTEGER(ip_) scalar that holds the largest front size.

OPSA is a REAL(rp_) scalar that is set to the number of floating-point additions performed during assembly.

OPSE is a REAL(rp_) scalar that is set to the number of floating-point operations performed during factorization.

OPSB is a REAL(rp_) scalar that is set to the number of additional floating-point operations performed during factorization because of use of the BLAS.

NRLTOT and NIRTOT are INTEGER(ip_) scalars that give the total amount of REAL(rp_) and INTEGER(ip_) words respectively required for a successful factorization without the need for data compression, provided the same pivots are used.

NRLNEC and NIRNEC are INTEGER(ip_) scalars that give the amount of REAL(rp_) and INTEGER(ip_) words required respectively for successful factorization allowing data compression, provided the same pivots are used.

NEBDU is an INTEGER(ip_) scalar that gives the total number of entries in the factorization.

NRLBDU and NIRBDU are INTEGER(ip_) scalars that give the amount of real and integer words used respectively to hold the factorization.

NCMPBR and NCMPBI are INTEGER(ip_) scalars that hold the number of compresses of the real and integer data structure respectively required by the factorization.

NTWO is an INTEGER(ip_) scalar that holds the number of 2 by 2 pivots used during the factorization.

NEIG is an INTEGER(ip_) scalar that holds the number of negative eigenvalues of $\mathbf{A}$.

RANK is an INTEGER(ip_) scalar that holds the rank of the original factorization.

DELAY is an INTEGER(ip_) scalar that holds the number of pivots passed up the tree because of numerical pivoting considerations.

SIGNC is an INTEGER(ip_) scalar that holds the number of sign changes of pivot when SILS_CONTROL%PIVOTING is set to 3.

MODSTEP is an INTEGER(ip_) scalar that holds the pivot step at which matrix modification is first performed when SILS_CONTROL%PIVOTING is set to 4.

MAXCHANGE is a REAL(rp_) scalar that is set to the value of the largest change made to a pivot when SILS_AINFO%MODSTEP is positive.

2.2.5 The derived data type for holding informational parameters from SILS_SOLVE

The derived data type SILS_SINFO is used to hold parameters that give information about the progress of the SILS_FACTORIZE subroutine. The components of SILS_SINFO are:

FLAG is an INTEGER(ip_) scalar. The value of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.3.

STAT is an INTEGER(ip_) scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

2.2.6 The derived data type for holding factors of a matrix

The derived data type SILS_FACTORS is used to hold the factors and related data for a matrix. All components are private.

2.3 Argument lists and calling sequences

We use square brackets [ ] to indicate OPTIONAL arguments.

2.3.1 The initialization subroutine

The initialization subroutine must be called for each structure used to hold the factors. It may also be called for a structure used to control the subroutines. Each argument is optional. A call with no arguments has no effect.

```
CALL SILS_INITIALIZE( [FACTORS][, CONTROL] )
```

FACTORS is optional, scalar, of INTENT(OUT) and of type SILS_FACTORS. On exit, its allocatable array components will have targets of length zero. Without such initialization, these components are undefined and other calls are likely to fail.

CONTROL is optional, scalar, of INTENT(OUT) and of type SILS_CONTROL. On exit, its components will have been given the default values specified in Section 2.2.2.

2.3.2 The sparsity pattern analysis subroutine

The sparsity pattern of **A** may be analysed as follows:

```
CALL SILS_ANALYSE( MATRIX, FACTORS, CONTROL, AINFO[, PERM] )
```

MATRIX is scalar, of INTENT(IN) and of type SMT_TYPE. The user must set the components N, NE, ROW, and COL, and they are not altered by the subroutine. **Restrictions:** MATRIX%N $\geq$ 1 and MATRIX%NE $\geq$ 0.

FACTORS is scalar, of INTENT(INOUT) and of type SILS_FACTORS. It must have been initialized by a call to SILS_INITIALIZE or have been used for a previous calculation. In the latter case, the previous data will be lost but the allocatable arrays will not be reallocated unless they are found to be too small.

CONTROL is scalar, of INTENT(IN) and of type SILS_CONTROL. Its components control the action, as explained in Section 2.2.2.

AINFO is scalar, of INTENT(OUT) and of type SILS_AINFO. Its components provide information about the execution, as explained in Section 2.2.3.

PERM is an optional rank-one INTEGER(ip_) array of INTENT(IN) and length N. If present, PERM (i), $i = 1,\ldots,n$, should be set to the position of variable i in the pivotal sequence.

2.3.3 The numerical factorization subroutine

Once it has been analysed, the matrix **A** may be factorized as follows:

```
CALL SILS_FACTORIZE( MATRIX, FACTORS, CONTROL, FINFO )
```

MATRIX is scalar, of INTENT(IN) and of type SMT_TYPE. The components N and NE must be unaltered since the call to SILS_ANALYSE. The user must set the component VAL to hold the real values of the entries. None of the components are altered by the subroutine.

FACTORS is scalar, of INTENT(INOUT) and of type SILS_FACTORS. It must be unaltered since the call to SILS_ANALYSE or a subsequent call to SILS_FACTORIZE.

CONTROL is scalar, of INTENT(IN) and of type SILS_CONTROL. Its components control the action, as explained in Section 2.2.2.

FINFO is scalar, of INTENT(OUT) and of type SILS_FINFO. Its components provide information about the execution, as explained in Section 2.2.4.

2.3.4 The solution subroutine

Given the factorization, a set of equations may be solved as follows:

```
CALL SILS_SOLVE( MATRIX, FACTORS, X, CONTROL, SINFO[, RHS] )
```

MATRIX is scalar, of INTENT(IN) and of type SMT_TYPE. It must be unaltered since the call to SILS_FACTORIZE and is not altered by the subroutine.

FACTORS is scalar, of INTENT(IN) and of type SILS_FACTORS. It must be unaltered since the call to SILS_FACTORIZE and is not altered by the subroutine.

X is an assumed-shape array with 1 or 2 dimensions, of INTENT(INOUT) and of type REAL(rp_). If RHS is absent, X must be set by the user to the vector **b** or the matrix **B** and on return it holds the solution **x** or **X**. If RHS is present, X must be set by the user to an approximate solution and on return it holds an improved solution, obtained by one cycle of iterative refinement without any use of arithmetic with additional precision.

CONTROL is scalar, of INTENT(IN) and of type SILS_CONTROL. Its components control the action, as explained in Section 2.2.2.

SINFO is scalar, of INTENT(OUT) and of type SILS_SINFO. Its components provide information about the execution, as explained in Section 2.2.5.

RHS is an assumed-shape array of the same shape as X, optional, of intent(in), and of type REAL(rp_). If present, it must be set by the user to the vector **b** or the matrix **B**.

2.3.5 The termination subroutine

All previously allocated arrays are reallocated to have targets of length zero as follows:

```
CALL SILS_FINALIZE( FACTORS, CONTROL, INFO )
```

FACTORS is scalar, of INTENT(INOUT) and of type SILS_FACTORS. On exit, its allocatable array components will have been reallocated to have targets of length zero. Without such finalization, the storage occupied is unavailable for other purposes. In particular, this is very wasteful if the structure goes out of scope on return from a procedure.

CONTROL is scalar, of INTENT(IN) and of type SILS_CONTROL. Its components control the action, as explained in Section 2.2.2.

INFO is scalar, of INTENT(OUT) and of type INTEGER(ip_). On return, the value 0 indicates success. Any other value is the STAT value of an ALLOCATE or DEALLOCATE statement that has failed.

2.4 Warning and error messages

2.4.1 When analysing the sparsity pattern

A successful return from SILS_ANALYSE is indicated by AINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. Possible negative values are:

-1 Value of MATRIX%N out of range. MATRIX%N $<$ 1. AINFO%MORE is set to value of MATRIX%N.

-2 Value of MATRIX%NE out of range. MATRIX%NE $<$ 0. AINFO%MORE is set to value of MATRIX%NE.

-3 Failure of an allocate or deallocate statement. AINFO%STAT is set to the STAT value.

-9 The array PERM does not hold a permutation. AINFO%MORE holds first component at which error was detected.

A positive flag value is associated with a warning message which will be output on unit AINFO%MP. Possible positive values are:

1 Index (in MATRIX%ROW or MATRIX%COL) out of range. Action taken by subroutine is to ignore any such entries and continue. AINFO%OOR is set to the number of such entries. Details of the first ten are printed on unit CONTROL%MP.

2 Duplicate indices. Action taken by subroutine is to keep the duplicates and then to sum corresponding reals when SILS_FACTORIZE is called. AINFO%DUP is set to the number of faulty entries. Details of the first ten are printed on unit CONTROL%MP.

3 Both out-of-range indices and duplicates exist.

2.4.2 When factorizing the matrix

A successful return from SILS_FACTORIZE is indicated by FINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, no factorization will have been calculated. Possible negative values are:

-1 Value of MATRIX%N differs from the SILS_ANALYSE value. FINFO%MORE holds value of MATRIX%N.

-2 Value of MATRIX%NE out of range. MATRIX%NE $<$ 0.
FINFO%MORE holds value of MATRIX%NE.

-3 Failure of an allocate or deallocate statement. FINFO%STAT is set to the STAT value.

-5 Zero pivot detected (CONTROL%PIVOTING has the value 2 or 3). FINFO%MORE is set to the pivot step at which this was detected.

-6 A change of sign of pivots has been detected (CONTROL%PIVOTING has the value 2). FINFO%MORE is set to the pivot step at which this was detected.

-7 The real array that holds data for the factors needs to be bigger than CONTROL%MAXLA.

-8 The integer array that holds data for the factors needs to be bigger than CONTROL%MAXLIW.

A positive flag value is associated with a warning message which will be output on unit CONTROL%MP. In this case, a factorization will have been calculated.

4 Matrix is rank deficient. In this case, FINFO%RANK will be set to the rank of the original factorization, but the factorization is altered by changing all the zero pivots to one. This will enable the subsequent solution of consistent sets of equations.

5 Pivots have different signs when CONTROL%PIVOTING has the value 3. FINFO%NEIG is set to the number of negative eigenvalues. Details of the first ten are printed on unit CONTROL%MP. FINFO%MORE is set to the number of sign changes.

2.4.3 When solving a linear system

A successful return from SILS_SOLVE is indicated by FINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, no solution will have been found. Possible negative values are:

-3 Failure of an allocate or deallocate statement. FINFO%STAT is set to the STAT value.

2.5 Further features

In this section, we describe features for enquiring about and manipulating the parts of the factorization constructed. These features will not be needed by a user who wants simply to solve systems of equations with matrix $\mathbf{A}$.

The algorithm produces an $\mathbf{LDL}^T$ factorization of a permutation of $\mathbf{A}$, where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{D}$ is a block diagonal matrix with blocks of order 1 and 2. It is convenient to write this factorization in the form

$$\mathbf{A} = (\mathbf{PLP}^T)(\mathbf{PDP}^T)(\mathbf{PL}^T\mathbf{P}^T),$$

where $\mathbf{P}$ is a permutation matrix. The following subroutines are provided:

SILS_ENQUIRE returns $\mathbf{P}$ or $\mathbf{D}$ or both.

SILS_ALTER_D alters $\mathbf{D}$. Note that this means that we no longer have a factorization of the given matrix $\mathbf{A}$.

SILS_PART_SOLVE solves one of the systems of equations $\mathbf{PLP}^T\mathbf{x} = \mathbf{b}$, $\mathbf{PDP}^T\mathbf{x} = \mathbf{b}$, or $\mathbf{PL}^T\mathbf{P}^T\mathbf{x} = \mathbf{b}$, for one or more right-hand sides.

2.5.1 To return P or D or both

```
CALL SILS_ENQUIRE( FACTORS[, PERM][, PIVOTS][, D][, PERTURBATION] )
```

FACTORS is scalar, of INTENT(IN) and of type SILS_FACTORS. It must be unaltered since the call to SILS_FACTORIZE or a subsequent call to SILS_ALTER_D.

PERM is an optional rank-one INTEGER(ip_) array of INTENT(OUT) and length N. If present, PERM will be set to the pivot permutation selected by SILS_ANALYSE.

PIVOTS is an optional rank-one INTEGER(ip_) array of INTENT(OUT) and length N. If present, the index of pivot i will be placed in PIVOTS (i), $i = 1,\ldots,n$, with its sign negative if it is the index of a 2 x 2 block.

D is an optional rank-two REAL(rp_) array of INTENT(OUT) and shape (2, N). If present, the diagonal entries of $\mathbf{D}^{-1}$ will be placed in D(1, i), $i = 1,\ldots n$ and the off-diagonal entries of $\mathbf{D}^{-1}$ will be placed in D(2, i), $i = 1,\ldots n-1$.

PERTURBATION is an optional rank-one REAL(rp_) array of INTENT(OUT) and length N. If present, PERTURBATION will be set to a vector of diagonal perturbations chosen by SILS_FACTORIZE. This array can only be nonzero if SILS_FACTORIZE was last called with CONTROL%PIVOTING = 4.

2.5.2 To alter D

```
CALL SILS_ALTER_D( FACTORS, D, INFO )
```

FACTORS is scalar, of INTENT(INOUT) and of type SILS_FACTORS. It must be unaltered since the call to SILS_FACTORIZE or a subsequent call to SILS_ALTER_D.

D is an array of shape (2, n) of INTENT(INOUT) and of type REAL(rp_). The diagonal entries of $\mathbf{D}^{-1}$ will be altered to D(1, i), $i = 1,\ldots n$ and the off-diagonal entries of $\mathbf{D}^{-1}$ will be altered to D(2, i), $i = 1,\ldots n-1$.

INFO is scalar, of INTENT(OUT) and of type INTEGER(ip_). On return, the value 0 indicates success and the value $i > 0$ indicates that D(2, i) is nonzero, but is not part of a block of order 2 of $\mathbf{D}$.

2.5.3 To perform a partial solution

```
CALL SILS_PART_SOLVE( FACTORS, CONTROL, PART, X, INFO )
```

FACTORS is scalar, of INTENT(IN) and of type SILS_FACTORS. It must be unaltered since the call to SILS_FACTORIZE or a subsequent call to SILS_ALTER_D.

CONTROL is scalar, of INTENT(IN) and of type SILS_CONTROL. Its components control the action, as explained in Section 2.2.2.

PART is scalar, of INTENT(IN) and of type CHARACTER. It must have one of the values

- L for solving $\mathbf{P}\mathbf{L}\mathbf{P}^T\mathbf{x} = \mathbf{b}$ or $\mathbf{P}\mathbf{L}\mathbf{P}^T\mathbf{X} = \mathbf{B}$,
- D for solving $\mathbf{P}\mathbf{D}\mathbf{P}^T\mathbf{x} = \mathbf{b}$ or $\mathbf{P}\mathbf{D}\mathbf{P}^T\mathbf{X} = \mathbf{B}$, or
- U for solving $\mathbf{P}\mathbf{L}^T\mathbf{P}^T\mathbf{x} = \mathbf{b}$ or $\mathbf{P}\mathbf{L}^T\mathbf{P}^T\mathbf{X} = \mathbf{B}$.

X is an assumed-shape array with 1 or 2 dimensions, of intent(inout), and of type REAL(rp_). It must be set by the user to the vector $\mathbf{b}$ or the matrix $\mathbf{B}$ and on return it holds the solution $\mathbf{x}$ or $\mathbf{X}$.

INFO is scalar, of INTENT(OUT) and of type INTEGER(ip_). On return, the value 0 indicates success. Any other value is the STAT value of an ALLOCATE or DEALLOCATE statement that has failed.

3 GENERAL INFORMATION

Use of common: None.

Workspace: Provided automatically by the module.

Other routines called directly: MA27A/AD, MA27B/BD, MA27C/CD.

Other modules used directly: GALAHAD_SMT_single/double.

Input/output: Error, warning and diagnostic messages only. Error messages on unit CONTROL%LP and warning and diagnostic messages on unit CONTROL%WP and CONTROL%MP, respectively. These have default value 6, and printing of these messages is suppressed if the relevant unit number is set negative. These messages are also suppressed if SILS_CONTROL%LDIAG is less than 1.

Restrictions: MATRIX%N ≥ 1, MATRIX%NE ≥ 0.

Portability: ISO Fortran 95 + TR 15581 or Fortran 2003. The package is thread-safe.

4 METHOD

A version of sparse Gaussian elimination is used.

The SILS_ANALYSE entry chooses pivots from the diagonal using the minimum degree criterion employing a generalized element model of the elimination thus avoiding the need to store the filled-in pattern explicitly. The elimination is represented as an assembly tree with the order of elimination determined by a depth-first search of the tree.

The SILS_FACTORIZE entry factorizes the matrix by using the assembly and elimination ordering generated by SILS_ANALYSE. At each stage in the multifrontal approach, pivoting and elimination are performed on full submatrices and, when diagonal 1×1 pivots would be numerically unstable, 2×2 diagonal blocks are used. The operations on the full submatrices are performed using the Level 3 BLAS. SILS_FACTORIZE can thus be used to factor indefinite systems and will perform well on machines with caches or levels of memory hierarchy.

The SILS_SOLVE entry uses the factors from SILS_FACTORIZE to solve systems of equations either by loading the appropriate parts of the vectors into an array of the current front-size and using full matrix code employing the Level 2 and Level 3 BLAS or by indirect addressing at each stage, whichever performs better.

Reference: A fuller account of this method is given by Duff and Reid (AERE-R.10533, 1982) and Duff and Reid, ACM Trans. Math. Software **9** (1983), 302-325.

5 EXAMPLE OF USE

We illustrate the use of the package on the solution of the single set of equations

$$\begin{pmatrix} 2 & 3 & & & \\ 3 & & 4 & & 6 \\ & 4 & 1 & 5 & \\ & & 5 & & \\ & 6 & & & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 8 \\ 45 \\ 31 \\ 15 \\ 17 \end{pmatrix}$$

(Note that this example does not illustrate all the facilities). Then we may use the following code:

Then, choosing the solver SILS, we may use the following code:

```
   PROGRAM SILS_EXAMPLE  !  GALAHAD 3.3 - 05/05/2021 AT 16:30 GMT.
   USE GALAHAD_SMT_double
   USE GALAHAD_SILS_double
   IMPLICIT NONE
   INTEGER, PARAMETER :: wp = KIND( 1.0D+0 )
   TYPE( SMT_type ) :: matrix
   TYPE( SILS_control ) :: control
   TYPE( SILS_ainfo ) :: ainfo
   TYPE( SILS_finfo ) :: finfo
   TYPE( SILS_sinfo ) :: sinfo
   TYPE( SILS_factors ) :: factors
   INTEGER, PARAMETER :: n = 5
   INTEGER, PARAMETER :: ne = 7
   REAL ( KIND = wp ) :: B( n ), X( n )
   INTEGER :: i
! Read matrix order and number of entries
   matrix%n = n
   matrix%ne = ne
! Allocate and set matrix
   ALLOCATE( matrix%val( ne ), matrix%row( ne ), matrix%col( ne ) )
   matrix%row( : ne ) = (/ 1, 1, 2, 2, 3, 3, 5 /)
   matrix%col( : ne ) = (/ 1, 2, 3, 5, 3, 4, 5 /)
   matrix%val( : ne ) = (/ 2.0_wp, 3.0_wp, 4.0_wp, 6.0_wp, 1.0_wp,             &
                           5.0_wp, 1.0_wp /)
   CALL SMT_put( matrix%type, 'COORDINATE', i )     ! Specify co-ordinate
! Set right-hand side
   B( : n ) = (/ 8.0_wp, 45.0_wp, 31.0_wp, 15.0_wp, 17.0_wp /)
! Initialize the structures
   CALL SILS_INITIALIZE( factors, control )
! Analyse
   CALL SILS_ANALYSE( matrix, factors, control, ainfo )
   IF ( ainfo%FLAG < 0 ) THEN
    WRITE(6,"( ' Failure of SILS_ANALYSE with AINFO%FLAG=',I0)" ) ainfo%FLAG
    STOP
   END IF
! Factorize
   CALL SILS_FACTORIZE( matrix, factors, control, finfo )
   IF ( finfo%FLAG < 0 ) THEN
     WRITE(6,"(' Failure of SILS_FACTORIZE with FINFO%FLAG=', I0)") finfo%FLAG
     STOP
   END IF
! Solve without refinement
   X = B
   CALL SILS_SOLVE( matrix, factors, X, control, sinfo )
   IF ( sinfo%FLAG == 0 ) WRITE(6,                                           &
     "(' Solution without refinement is',/,(3F20.16))") X
! Perform one refinement
   CALL SILS_SOLVE( matrix, factors, X, control, sinfo, B )
   IF( sinfo%FLAG == 0 ) WRITE(6,                                            &
     "(' Solution after one refinement is',/,(3F20.16))") X
! Clean up
   DEALLOCATE( matrix%type, matrix%val, matrix%row, matrix%col )
   STOP
END PROGRAM SILS_EXAMPLE
```

This produces the following output:

```
Solution without refinement is
 1.0000000000000000  2.0000000000000000  3.0000000000000000
 4.0000000000000000  5.0000000000000000
Solution after one refinement is
 1.0000000000000000  2.0000000000000000  3.0000000000000000
 4.0000000000000000  5.0000000000000000
```