



GALAHAD

SMT

USER DOCUMENTATION

GALAHAD Optimization Library version 3.0

1 SUMMARY

This package defines a derived type capable of **supporting a variety of sparse matrix storage schemes**. Its principal use is to allow exchange of data between GALAHAD subprograms and other codes. The derived type is structurally equivalent to the type `ZD11_type` available from the HSL package `ZD11`.

ATTRIBUTES — Versions: `GALAHAD_SMT_single`, `GALAHAD_SMT_double`. **Uses:** None. **Date:** March 1998. **Origin:** N. I. M. Gould and J. K. Reid, Rutherford Appleton Laboratory. **Language:** Fortran 95 + TR 15581 or Fortran 2003.

2 HOW TO USE THE PACKAGE

Access to the package requires a `USE` statement such as

Single precision version

```
USE GALAHAD_SMT_single
```

Double precision version

```
USE GALAHAD_SMT_double
```

If it is required to use both modules at the same time, the derived type `SMT_type` (Section 2.1), the subroutine `SMT_put`, and the function `SMT_get` (Section 2.2) must be renamed on one of the `USE` statements.

2.1 The derived data type

A single derived data type, `SMT_type`, is accessible from the package. It is intended that, for any particular application, only those components which are needed will be set. The components are:

`id` is an allocatable array of rank one and type default `CHARACTER` that may be used to hold the name of the matrix.

`type` is an allocatable array of rank one and type default `CHARACTER` that may be used to hold a key which indicates the type (or kind) of the matrix in question.

`m` is a scalar component of type default `INTEGER` that may be used to hold the number of rows in the matrix.

`n` is a scalar component of type default `INTEGER` that may be used to hold the number of columns in the matrix.

`ne` is a scalar component of type default `INTEGER` that may be used to hold the number of entries in the matrix.

`row` is an allocatable array of rank one and type default `INTEGER` that may be used to hold the row indices of the entries of the matrix.

`col` is an allocatable array of rank one and type default `INTEGER` that may be used to hold the column indices of the entries of the matrix.

`val` is an allocatable array of rank one and type default `REAL` (double precision in `GALAHAD_SMT_double`) that may be used to hold the numerical values of the entries of the matrix.

`ptr` is an allocatable array of rank one and type default `INTEGER` that may be used to hold the starting positions of each row in a row-wise storage scheme, or the starting positions of each column in a column-wise storage scheme.

All use is subject to licence. See <http://galahad.rl.ac.uk/galahad-www/cou.html>.
For any commercial application, a separate license must be signed.

2.2 Argument lists and calling sequences

To assist use of the character arrays in the components %id and %type, the module provides two procedures:

1. The subroutine SMT_put allocates a character array and sets its components from a character variable.
2. The function SMT_get obtains the elements of a character array as a character variable.

We use square brackets [] to indicate OPTIONAL arguments.

2.2.1 Allocate a character array and set its components

To allocate a character array and set its components from a character variable,

```
CALL SMT_put( array, string, stat )
```

array is a rank one allocatable array of type default CHARACTER. If string is present, array is allocated with size LEN_TRIM(string) and its elements are given the values string(i:i), i = 1, 2, ...; otherwise, array is allocated to be of size zero.

string is an OPTIONAL, INTENT(IN) argument of type CHARACTER with any character length.

stat is an INTENT(OUT) argument of type default INTEGER. An ALLOCATE statement with this as its STAT= variable is executed and a successful allocation will be indicated by the value zero.

2.2.2 Obtain the elements of a character array as a character variable

To obtain the elements of a character array as a character variable,

```
string = SMT_get( array )
```

array is an INTENT(IN) array of rank one and type default CHARACTER. It is not altered.

The result is scalar and of type CHARACTER(LEN=SIZE(array)). SMT_get(i:i) is given the value array(i), i = 1, 2, ..., SIZE(array).

3 GENERAL INFORMATION

Other modules used directly: None.

Input/output: None.

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

4 EXAMPLE OF USE

Suppose that we wish to store the symmetric matrix

$$\begin{pmatrix} 1.0 & & \\ & 1.0 & \\ & & 1.0 \end{pmatrix},$$

whose name is “Sparse”, using a coordinate sparse matrix storage format. Then the following code is appropriate:

All use is subject to licence. See <http://galahad.rl.ac.uk/galahad-www/cou.html>.
For any commercial application, a separate license must be signed.

```

PROGRAM GALAHAD_SMT_example
USE GALAHAD_SMT_double
INTEGER :: i
TYPE ( SMT_type ) :: A
A%n = 3 ; A%ne = 2
ALLOCATE( A%row( A%ne ), A%col( A%ne ), A%val( A%ne ) )
CALL SMT_put( A%id, 'Sparse' )      ! Put name into A%id
CALL SMT_put( A%type )             ! Allocate space for A%type
A%row( 1 ) = 1 ; A%col( 1 ) = 1 ; A%val( 1 ) = 1.0
A%row( 2 ) = 2 ; A%col( 2 ) = 3 ; A%val( 2 ) = 1.0
WRITE( 6, "( 3A, I2, //, A )" ) ' Matrix ', SMT_get( A%id ), &
    ' dimension', A%n, ' row col value '
DO i = 1, A%ne
    WRITE( 6, "( I3, 1X, I3, ES9.1 )" ) A%row( i ), A%col( i ), A%val( i )
END DO
DEALLOCATE( A%id, A%row, A%col, A%val )
END PROGRAM GALAHAD_SMT_example

```

This produces the following output:

```

Matrix Sparse dimension 3

row col value
 1  1  1.0E+00
 2  3  1.0E+00

```

For examples of how the derived data type `packagename_problem_type` may be used in conjunction with the GALAHAD linear equation solver, see the specification sheet for the package `GALAHAD_SILS`.

All use is subject to licence. See <http://galahad.rl.ac.uk/galahad-www/cou.html>.
For any commercial application, a separate license must be signed.