Science and Technology Facilities Council

HSL_MC65

PACKAGE SPECIFICATION HSL

1 SUMMARY

HSL_MC65 is a suite of Fortran 95 procedures for constructing and manipulating sparse matrix objects in compressed sparse row format.

For a general sparse matrix, the compressed sparse row format consists of three arrays, PTR, COL and VAL. PTR holds the starting positions of the rows in the COL and VAL arrays. The indices of the entries of row I are held in COL(PTR(I):PTR(I+1)-1) and the corresponding values are held in VAL(PTR(I):PTR(I+1)-1). However, the user should not need to deal with these arrays individually; HSL_MC65 encapsulates them in a sparse matrix object of the derived type HSL_ZD11_TYPE. HSL_MC65 provides procedures that perform basic operations, such as sparse matrix summation and multiplication, on these sparse matrix objects. There is an option for omitting VAL, that is, for a pattern-only matrix.

ATTRIBUTES — Version: 2.3.0 (13 February 2014). **Types:** Real (single, double). **Uses:** HSL_ZD11. **Date:** November 2000. **Origin:** Y. F. Hu, Daresbury Laboratory. **Language:** Fortran 95 + TR 15581 (allocatable components).

2 HOW TO USE THE PACKAGE

2.1 General overview

There are 25 procedures available to the user. A detailed description of each procedure is found in Section 3 and an index is found in Section 4. A brief description of the facilities is given in the following paragraph.

Sections 3.1 and 3.2 contain procedures for constructing or destructing a sparse matrix object. Section 3.3 provides a procedure to expand or shrink the memory allocated for a sparse matrix object. Section 3.4 contains procedures to check whether a matrix has no entry values or whether it is symmetric. A number of procedures are provided in Section 3.5 for symmetrizing, cleaning, sorting, removing diagonal entries, moving diagonal entries to the row starts, and condensing a sparse matrix pattern. Section 3.6 provides procedures to access a row of a sparse matrix. Section 3.7 contains a procedure for matrix-matrix operations (summation, multiplications, comparison of two matrices). Section 3.8 has procedures for matrix-vector multiplications. Section 3.9 provides a procedure to make a new copy of an existing matrix. Section 3.10 contains a procedure for transposing a matrix. Section 3.11 has a procedure for converting a sparse matrix into coordinate format. Section 3.12 contains procedures for the input/output of a sparse matrix. Finally, error/warning messages can optionally be printed using the procedure in Section 3.13.

Throughout HSL_MC65, a matrix object is assumed to contain no duplicated entries. Duplicated entries can be treated (summed) either during the construction of a sparse matrix (Section 3.1) by setting the CHECKING flag or by using the matrix cleaning procedure in Section 3.5.

2.2 Calling sequences

Access to the package requires a USE statement to use the modules of HSL_MC65 and HSL_ZD11.

Single precision version

```
USE HSL_MC65_SINGLE
USE HSL_ZD11_SINGLE
```

Double precision version

```
USE HSL_MC65_DOUBLE
USE HSL_ZD11_DOUBLE
```

If it is required to use the single and the double precision versions at the same time, the derived type ZD11_type and any MC65 procedures invoked must be renamed in one of the use statements.

2.3 The derived data types

The main derived data type used is ZD11_TYPE. The following components are employed:

M is an INTEGER scalar that holds the number of rows.

N is an INTEGER scalar that holds the number of columns.

PTR is an INTEGER allocatable array of rank one that holds the starting position of each row in a compressed sparse row storage scheme.

COL is an INTEGER allocatable array of rank one that holds the column indices of the entries of the sparse matrix in a compressed sparse row storage scheme.

VAL is a REAL (double precision REAL in HSL_MC65_DOUBLE) allocatable array of rank one that holds the entry values of the sparse matrix in a compressed sparse row storage scheme. This array is not allocated if the sparse matrix is a pattern-only matrix.

TYPE is a CHARACTER (LEN=1) allocatable array of rank one used to indicate the properties of the sparse matrix. Two main types are employed. By default, a matrix is of type "general" and has numerical values for its entries. When the matrix is of type "pattern", it has no such values.

The other components are not used.

3 THE ARGUMENT LISTS

We use square brackets [ ] to indicate OPTIONAL arguments. In each call, optional arguments follow the argument INFO. Since we reserve the right to add additional optional arguments in future releases of the code, **we strongly recommend that all optional arguments be called by keyword, not by position.**

3.1 The construction of a sparse matrix object

Before using a sparse matrix object, it must first be constructed by calling the generic subroutine MC65_MATRIX_CONSTRUCT. This provides three methods of constructing the sparse matrix object. The subroutine MC65_MATRIX_DESTRUCT (Section 3.2) should be called when the sparse matrix object is no longer needed.

3.1.1 Constructing a sparse matrix object: storage allocation only

This method constructs a sparse matrix object by setting the type of the sparse matrix and allocating storage space for the entries. It does not define the content of the matrix.

```
CALL MC65_MATRIX_CONSTRUCT(MATRIX,M,NZ,INFO[,N,TYPE,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, it holds the sparse matrix object. By default, the sparse matrix is assumed to be of type "general" and allocatable arrays of size NZ are allocated for the column indices and entry values. When the optional argument TYPE is present, the matrix type is set to TYPE. If TYPE = "pattern", the matrix is assumed to be a pattern-only matrix, the allocatable array for the column indices is allocated to be of size NZ, and the allocatable array for the entry values is left unallocated.

M is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of rows.

NZ is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the size of the storage to be allocated for the entries of the matrix. This must be greater than or equal to the number of entries in the matrix. If NZ < 0, storage of size 0 is allocated.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully; INFO = MC65_ERR_MEMORY_ALLOC if memory allocation has failed; and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation has failed An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

N is an OPTIONAL INTEGER scalar of INTENT (IN). If present, it must be set by the user to hold the number of columns. If not present, the number of columns is M.

TYPE is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If present, it must be set by the user to hold the type of the matrix. If not present, the type of the matrix will be set to "general". If the matrix is a pattern-only matrix, the user must set it to "pattern".

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.1.2 Constructor from compressed sparse row format

This method constructs a sparse matrix MATRIX from arrays holding it in compressed sparse row format. Storage is allocated for MATRIX and the content of MATRIX is copied from the arrays.

```
CALL MC65_MATRIX_CONSTRUCT(MATRIX,M,N,PTR,COL,INFO[,VAL,TYPE, &
   CHECKING,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, it holds the sparse matrix object.

- The type of the matrix is as shown in the following table

	VAL is present	VAL is not present
TYPE is present	TYPE	"pattern"
TYPE is not present	"general"	"pattern"

- The number of entries presented by the user is NZ = PTR(M+1)-1. Storage of this size is allocated for MATRIX and data is copied from the arrays PTR(1:M+1) and COL(1:NZ). In addition, if MATRIX is not of type "pattern", data is copied from the array VAL(1:NZ).

M is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of rows.

N is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of columns.

PTR is an INTEGER array of rank one with INTENT (IN) and size M+1. It must be set by the user to hold the starting positions of the rows in a compressed sparse row storage format. More specifically, the column indices of row I must be in COL(PTR(I):PTR(I+1)-1). Furthermore, if MATRIX is not of type "pattern", the entry values of row I must be in VAL(PTR(I):PTR(I+1)-1).

COL is an INTEGER array of rank one with INTENT (IN) and size NZ = PTR(M+1)-1. It must be set by the user to hold the column indices.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.

- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_RANGE_PTR if PTR(1:M+1) is not monotonically increasing.
- INFO = MC65_WARN_RANGE_COL if any element of COL(1:NZ) is not within the range [1,N]. Any such entries are excluded from MATRIX.
- INFO = MC65_WARN_DUP_ENTRY if duplicated entries were found and treated (summed).
- INFO = MC65_WARN_ENTRIES if both duplicate entries and out-of-range entries were found and treated.

An error/warning message can be printed by calling subroutine MC65_PRINT_MESSAGE.

VAL is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) array of rank one with INTENT (IN) and size NZ = PTR(M+1)-1. If present, it must be set by the user to hold the values of the entries of the sparse matrix.

TYPE is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If TYPE and VAL are present, TYPE must be set by the user to hold the type of the matrix.

CHECKING is an OPTIONAL INTEGER scalar with INTENT (IN). When present and set by the user to a non-negative value, the matrix will be cleaned by treating (summing) duplicate entries and the input data will be checked for consistency. If a column index is out of range, the entry will be excluded and a warning will be recorded. When absent or present and set by the user to a negative value, no such checks are made.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.1.3 Constructor from coordinate format

This method constructs a sparse matrix from arrays holding it in coordinate format. Storage is allocated for MATRIX and the content of MATRIX is copied from the arrays.

```
CALL MC65_MATRIX_CONSTRUCT(MATRIX,M,N,NZ,IRN,JCN,INFO[,VAL, &
   TYPE,CHECKING,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, it holds the sparse matrix object. The type of the matrix is as shown in the following table

	VAL is present	VAL is not present
TYPE is present	TYPE	"pattern"
TYPE is not present	"general"	"pattern"

M is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of rows.

N is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of columns.

NZ is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the number of entries in the sparse matrix.

IRN is an INTEGER rank-one array of size NZ with INTENT (IN). It must be set by the user to hold the row indices of the matrix.

JCN is an INTEGER rank-one array of size NZ with INTENT (IN). It must be set by the user to hold the column indices of the matrix.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_WARN_RANGE_IRN if any element of IRN is not within the range [1,M]. Any such entries are excluded from MATRIX.
- INFO = MC65_WARN_RANGE_JCN if any element of JCN is not within the range [1,N]. Any such entries are excluded from MATRIX.
- INFO = MC65_WARN_RANGE_BOTH if both IRN and JCN contain elements that are out-of-range. Any such entries are excluded from MATRIX.
- INFO = MC65_WARN_DUP_ENTRY if duplicated entries were found and treated (summed).
- INFO = MC65_WARN_ENTRIES if both duplicate entries and out-of-range entries were found and treated.

An error/warning message can be printed by calling subroutine MC65_PRINT_MESSAGE.

VAL is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) rank-one array of size NZ with INTENT (IN). If present, it must be set by the user to hold the values of the entries of the sparse matrix.

TYPE is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If TYPE and VAL are present, TYPE must be set by the user to hold the type of the matrix.

CHECKING is an OPTIONAL INTEGER scalar with INTENT (IN). When present and set by the user to a non-negative value, the matrix will be cleaned by treating (summing) duplicate entries and the input data will be checked for consistency. If a column index is out of range, the entry will be excluded and a warning will be recorded. When absent or present and set by the user to a negative value, no such checks are made.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.2 Destructor

When a sparse matrix object is no longer required, MC65_MATRIX_DESTRUCT should be called. This releases the memory occupied by the sparse matrix object.

```
CALL MC65_MATRIX_DESTRUCT(MATRIX,INFO[,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the sparse matrix object to be destroyed.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation has failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.3 Storage management: reallocating storage

When it is necessary to change the allocated storage for an existing matrix object to cater for a changed number of matrix entries, the subroutine MC65_MATRIX_REALLOCATE should be used.

This subroutine allocates new storage of the amount as specified by NZ, copies the matrix entries into it, and releases the old storage. If the matrix is of the type "pattern", the reallocation is only performed for the allocatable array MATRIX%COL, otherwise the reallocation is performed for both MATRIX%COL and MATRIX%VAL. If NZ < 0, storage of size 0 is allocated. When NZ is smaller than the size of the allocatable array(s), only the first NZ elements of the array(s) are copied.

```
CALL MC65_MATRIX_REALLOCATE(MATRIX,NZ,INFO[,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the sparse matrix to be reallocated. On exit, it holds the matrix with changed storage.

NZ is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the size for the allocatable array of column indices. When the matrix is not of type "pattern", NZ is also the size for the allocatable array of entry values of the matrix. If NZ < 0, arrays of size 0 will be allocated.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine returns successfully; INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed; and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.4 Getting properties of a matrix

3.4.1 Checking whether a matrix has no entry values

The function MC65_MATRIX_IS_PATTERN checks whether a matrix has no entry values, in other words, whether it is of type "pattern".

```
PATTERN = MC65_MATRIX_IS_PATTERN(MATRIX)
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the sparse matrix.

PATTERN is a LOGICAL scaler. On exit, PATTERN=.TRUE. if the matrix is of type "pattern", otherwise PATTERN=.FALSE..

3.4.2 Testing for symmetry

The subroutine MC65_MATRIX_IS_SYMMETRIC checks whether a matrix is symmetric. It returns SYMMETRIC = .TRUE. if the matrix is symmetric, and SYMMETRIC = .FALSE. if not. When the matrix is of type "pattern", only structural symmetry is checked, otherwise the matrix is said to be symmetric if it is both structurally and numerically symmetric. Here a matrix A is said to be numerically symmetric if for each entry a_{ij}, the inequality $|a_{ij} - a_{ji}| \leq E$ holds, where E is a tolerance set by default to $E = 0.0$.

Two OPTIONAL arguments are supported. If the OPTIONAL argument PATTERN is present and is set to .TRUE., only structural symmetry is checked. If the optional argument TOL is present, the tolerance E is set to TOL.

```
CALL MC65_MATRIX_IS_SYMMETRIC(MATRIX,SYMMETRIC,INFO[,PATTERN,TOL,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the sparse matrix whose symmetry is to be checked.

SYMMETRIC is a LOGICAL scalar of INTENT (OUT). On exit,

- If the matrix is of type "pattern" or if PATTERN is present with value .TRUE., SYMMETRIC is .TRUE. if the matrix is structurally symmetric and .FALSE. otherwise.
- Otherwise, SYMMETRIC is .TRUE. if the matrix is both structurally and numerically symmetric, .FALSE. otherwise.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine returns successfully; INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed; and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

PATTERN is an OPTIONAL LOGICAL scalar of INTENT (IN). When it is present with the value .TRUE., only structural symmetry is checked. Otherwise, structural and numerical symmetry are checked.

TOL is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) scalar of INTENT (IN). If present, it must be set by the user to hold the tolerance for $|a_{ij} - a_{ji}|$.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.5 Changing the properties of a matrix

3.5.1 Making a matrix symmetric

The subroutine MC65_MATRIX_SYMMETRIZE makes a matrix symmetric by summing it with its transpose or replacing it with the graph of this sum (here the graph of a symmetric matrix is defined as the matrix with diagonal elements removed and with all entry values set to 1). The graph can be used for forming the connectivity graph of a unsymmetric matrix.

```
CALL MC65_MATRIX_SYMMETRIZE(MATRIX,INFO[,GRAPH,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the matrix to be made symmetric. On exit, it contains the symmetrized matrix.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine returns successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_SUM_DIM_MISMATCH if trying to symmetrize a non-square matrix.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

GRAPH is an OPTIONAL LOGICAL scalar of INTENT (IN). If present with the value .TRUE., the graph of the summation, excluding the diagonal, is returned; if the matrix is not of the type "pattern", the values of the matrix entries are all set to 1.0.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.5.2 Cleaning a matrix

The subroutine MC65_MATRIX_CLEAN checks a matrix for duplicate entries (that is, the same column index appearing more than once in a row), and merges these duplicate entries. When the matrix is not of type "pattern", the value of duplicate entries are summed. If the optional argument REALLOC is present, a reallocation may be carried out to reduce the storage to what is required for the "cleaned" matrix.

```
CALL MC65_MATRIX_CLEAN(MATRIX,INFO[,REALLOC,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT), On entry, it must be set by the user to hold the sparse matrix to be cleaned. On exit, it holds the cleaned matrix.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully;
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed; and
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_WARN_DUP_ENTRY if duplicated entries were found and treated (summed).

An error/warning message can be printed by calling subroutine MC65_PRINT_MESSAGE.

REALLOC is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) scalar of INTENT (IN). If present, it must be set by the user to control the reallocation of storage.

- if REALLOC < 0, no reallocation.
- if REALLOC = 0, reallocation is carried out.
- if REALLOC > 0, reallocation is carried out if and only if memory saving is greater than REALLOC*100%. For example, if REALLOC = 0.5, reallocation will be carried out if saving in storage is greater than 50%.

If REALLOC is not present, no reallocation is carried out.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.5.3 Sorting the column entries of a matrix

Subroutine MC65_MATRIX_SORT sorts a matrix so that the column indices in each row are in increasing order. The sorting is not carried out in place.

```
CALL MC65_MATRIX_SORT(MATRIX,INFO[,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT), On entry, it must be set by the user to hold the matrix to be sorted. On exit, the column indices in each row of this matrix are in increasing order.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully; INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed; and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.5.4 Removing the diagonal of a matrix

Subroutine MC65_MATRIX_REMOVE_DIAGONAL removes the diagonal of a matrix.

```
CALL MC65_MATRIX_REMOVE_DIAGONAL(MATRIX)
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the matrix whose diagonal is to be removed. On exit, the diagonal of the matrix will have been removed.

3.5.5 Move the diagonal entries to the starts of the rows

Subroutine MC65_MATRIX_DIAGONAL_FIRST modifies the data structure of a sparse matrix object MATRIX so that the diagonal entry in each row is moved to the first position within the row.

```
CALL MC65_MATRIX_DIAGONAL_FIRST(MATRIX,INFO)
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the matrix.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully. INFO = MC65_WARN_MV_DIAG_MISSING if some diagonal entries are missing. An error/warning message can be printed by calling subroutine MC65_PRINT_MESSAGE.

3.5.6 Matrix condensing

Subroutine MC65_MATRIX_CONDENSE forms super-columns in a sparse matrix of type "pattern" by removing columns that have no entries and merging other columns that have the same pattern into "super-columns".

There is an option to provide column weights and have them summed over each super-column. The user is free hold these column weights in a real or an integer array (and it would not be an error to use both).

There is also an option to remove any column with less than a specified number of entries.

```
CALL MC65_MATRIX_CONDENSE(MATRIX,INFO[,COL_WGT_INT,COL_WGT_REAL,&
   REALLOC, IREMOVE,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry, it must be set by the user to hold the patern-only matrix to be condensed. On exit, all columns with no entries will have been deleted. If the optional argument IREMOVE is present, any columns that have no more than IREMOVE entries will also have been deleted. Other columns that have the same pattern will have been merged. MATRIX%N will have been changed to the number of super-columns.

COL_WGT_INT is an OPTIONAL INTEGER rank-one array of INTENT (INOUT) and size MATRIX%N. If present, it must be set by the user on entry to hold column weights. On exit, the leading elements will hold the super-column weights of the condensed matrix.

COL_WGT_REAL is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) rank-one array of INTENT (INOUT) and size MATRIX%N. If present, it must be set by the user on entry to hold column weights. On exit, the leading elements will hold the super-column weights of the condensed matrix.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_CONDENSE_NOPAT if the matrix is not of type "pattern".
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

REALLOC is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) scalar of INTENT (IN). If present, it must be set by the user to control the reallocation of storage.

- if REALLOC < 0, no reallocation.
- if REALLOC = 0, reallocation is carried out
- if REALLOC > 0, reallocation is carried out if and only if memory saving is greater than REALLOC*100%. For example, if REALLOC = 0.5, reallocation will be carried out if saving in storage is greater than 50%

If REALLOC is not present, no reallocation is carried out.

IREMOVE is an OPTIONAL INTEGER scalar of INTENT (IN). When present, it must be set by the user. Any columns that have no more than IREMOVE entries are removed.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.6 Accessing part of a matrix

3.6.1 Accessing the column indices of a row

Subroutine MC65_MATRIX_GETROW(MATRIX,I,COL) associates an integer pointer array with the section of MATRIX%COL that holds the column indices of the I-th row of the matrix.

```
CALL MC65_MATRIX_GETROW(MATRIX,I,COL)
```

MATRIX is a scalar of the derived type ZD11_TYPE with the TARGET attribute and INTENT (IN). It must be set by the user to hold the sparse matrix whose row is to be accessed.

I is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the index of the row of MATRIX to be accessed.

COL is an INTEGER POINTER array of rank one that becomes associated with the vector of column indices of row I.

3.6.2 Accessing the entry values of a row

Subroutine MC65_MATRIX_GETROWVAL(MATRIX,I,VAL) associates a real pointer array with the section of MATRIX%VAL that holds the values of the entries of the I-th row of the matrix.

```
CALL MC65_MATRIX_GETROWVAL(MATRIX,I,VAL)
```

MATRIX is a scalar of the derived type ZD11_TYPE with with the TARGET attribute and INTENT (IN). It must be set by the user to hold the sparse matrix whose row is to be accessed. It must have entry values and must not be of type "pattern").

I is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the index of the row of MATRIX to be accessed.

VAL is a REAL (double precision REAL for HSL_MC65_DOUBLE) POINTER array of rank one that becomes associated with the vector of values of the row.

3.7 Matrix-matrix operations

3.7.1 Matrix summation

Subroutine MC65_MATRIX_SUM adds two matrices MATRIX1 and MATRIX2 together, and returns the result in RESULT_MATRIX. If the OPTIONAL argument GRAPH is present and is .TRUE., the RESULT_MATRIX is the sum of MATRIX1 and MATRIX2, but with diagonal removed and all other entries set to 1.0 (1.0D0 for HSL_MC65_DOUBLE). Otherwise if the OPTIONAL argument SCALING is present, the scaled summation MATRIX1+SCALING*MATRIX2 is calculated.

The storage is allocated inside this routine for the sparse matrix object RESULT_MATRIX, and when RESULT_MATRIX is no longer needed, this memory should be deallocated by calling MC65_MATRIX_DESTRUCT.

```
CALL MC65_MATRIX_SUM(MATRIX1,MATRIX2,RESULT_MATRIX,INFO[,&
   SCALING,GRAPH,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the first matrix to be summed. It must have the same shape as the second matrix.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the second matrix to be summed.

RESULT_MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, RESULT_MATRIX is one of the following

- If GRAPH is present and is .TRUE., RESULT_MATRIX is a matrix of type "general", and is set to the sum of MATRIX1 and MATRIX2, with the diagonal removed and with all entry values set to 1.0 (1.0D0 for HSL_MC65_DOUBLE).
- Otherwise, the type of RESULT_MATRIX is set to "pattern" if either MATRIX1 or MATRIX2 is of this type; it is set to the type of MATRIX1 if both MATRIX1 and MATRIX2 have the same type; otherwise, it is set to "general". If the OPTIONAL argument SCALING is present, RESULT_MATRIX = MATRIX1 + SCALING*MATRIX2; otherwise, RESULT_MATRIX = MATRIX1 + MATRIX2.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_SUM_DIM_MISMATCH if the shape of MATRIX1 does not match that of MATRIX2.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

SCALING is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) scalar of INTENT (IN). When present, it must be set by the user to hold the scaling factor to be applied to MATRIX2 before MATRIX2 is summed with MATRIX1.

GRAPH is an OPTIONAL LOGICAL scalar of INTENT (IN). When present, it must be set by the user. If GRAPH is .TRUE., RESULT_MATRIX is a matrix of type "general", and is set to the sum of MATRIX1 and MATRIX2, with the diagonal removed and all other entries set to 1.0 (1.0D0 for HSL_MC65_DOUBLE)

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.7.2 Matrix multiplication

Subroutine MC65_MATRIX_MULTIPLY multiplies two sparse matrices MATRIX1 and MATRIX2, and returns the result in RESULT_MATRIX. The storage is allocated inside this routine for the sparse matrix object RESULT_MATRIX, and when RESULT_MATRIX is no longer needed, this memory should be deallocated using MC65_MATRIX_DESTRUCT.

```
CALL MC65_MATRIX_MULTIPLY(MATRIX1,MATRIX2,RESULT_MATRIX,INFO[,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the first matrix to be multiplied. It must have as many columns as the second matrix has rows.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the second matrix to be multiplied.

RESULT_MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, RESULT_MATRIX = MATRIX1*MATRIX2. The type of RESULT_MATRIX is set to "pattern" if either MATRIX1 or MATRIX2 has the type "pattern"; it is set to the type of MATRIX1 if both MATRIX1 and MATRIX2 have the same type; otherwise, it is set to "general".

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_MATMUL_DIM_MISMATCH if the number of columns of MATRIX1 is not equal to the number of rows of MATRIX2.
- INFO = MC65_ERR_SMM_TOO_LARGE if number of entries in RESULT_MATRIX exceeds maximum default integer.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.7.3 Matrix multiplication for graph theory applications

Subroutine MC65_MATRIX_MULTIPLY_GRAPH calculates RESULT_MATRIX $= \overline{\texttt{MATRIX1}} * \overline{\texttt{MATRIX2}} - \texttt{DIAG}\left(\overline{\texttt{MATRIX1}} * \overline{\texttt{MATRIX2}}\right)$. Here $\overline{\texttt{A}}$ stands for the matrix with the same pattern as A and with entry values of 1.0, and DIAG(A) stands for the diagonal of A. This subroutine is primarily used to calculate the matrix representation of the row (column) connectivity graph of a matrix A, which is given by $\overline{\texttt{A}}\overline{\texttt{A}}^{\texttt{T}} - \texttt{DIAG}\left(\overline{\texttt{A}}\overline{\texttt{A}}^{\texttt{T}}\right)$ (or $\overline{\texttt{A}}^{\texttt{T}}\overline{\texttt{A}} - \texttt{DIAG}\left(\overline{\texttt{A}}^{\texttt{T}}\overline{\texttt{A}}\right)$).

When the optional array COL_WGT is present, RESULT_MATRIX = $\overline{\texttt{MATRIX1}}$*D*$\overline{\texttt{MATRIX2}}$, where D is the diagonal matrix specified in COL_WGT.

Storage is allocated inside this routine for the sparse matrix object RESULT_MATRIX, and when it is no longer needed, storage occupied by RESULT_MATRIX should be deallocated using MC65_MATRIX_DESTRUCT.

```
CALL MC65_MATRIX_MULTIPLY_GRAPH(MATRIX1,MATRIX2,RESULT_MATRIX,&
   INFO[,COL_WGT,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the first matrix to be multiplied. It must have as many columns as the second matrix has rows.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the second matrix to be multiplied.

RESULT_MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit, RESULT_MATRIX = $\overline{\texttt{MATRIX1}} * \overline{\texttt{MATRIX2}}$. The type of RESULT_MATRIX is set to "general".

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_MATMULG_DIM_MISMATCH if the number of columns of MATRIX1 is not equal to the number of rows of MATRIX2.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

COL_WGT is an OPTIONAL REAL rank-one array of INTENT (IN) and size MATRIX1%N. When present, it must be set by the user to hold the column weights.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.7.4 Matrix comparison

Subroutine MC65_MATRIX_IS_DIFFERENT tests whether two matrices are different. There is an option for testing only for difference in pattern.

```
CALL MC65_MATRIX_IS_DIFFERENT(MATRIX1,MATRIX2,DIFF,INFO[,PATTERN,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to contain the first matrix to be compared.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to contain the second matrix to be compared.

DIFF is a REAL (double precision REAL in HSL_MC65_DOUBLE) scalar. On exit,

- if the shapes do not match, DIFF = HUGE(0.0) (DIFF = HUGE(0.0D0) for HSL_MC65_DOUBLE);
- else if the patterns do not match, DIFF = HUGE(0.0) (DIFF = HUGE(0.0D0) for HSL_MC65_DOUBLE);
- else if either MATRIX1 or MATRIX2 is of type "pattern", DIFF = 0.0 (DIFF = 0.0D0 for HSL_MC65_DOUBLE);
- else if PATTERN is present with the value .TRUE., DIFF = 0.0 (DIFF = 0.0D0 for HSL_MC65_DOUBLE);
- else DIFF = SUM(ABS((MATRIX1%VAL(1:NZ) - MATRIX2%VAL(1:NZ))), where NZ is the number of entries in MATRIX1 or MATRIX2.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine call was successful; INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed; and INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

PATTERN is an OPTIONAL LOGICAL scalar of INTENT (IN). If present, it must be set by the user. If PATTERN is .TRUE., only the difference in pattern is to be tested.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.8 Matrix-vector multiplication

The subroutine MC65_MATRIX_MULTIPLY_VECTOR multiplies a sparse matrix MATRIX or its transpose by a vector X and returns the result in Y. There is an option for multiplying $\overline{\texttt{MATRIX}}$ or its transpose by an integer vector, where $\overline{\texttt{MATRIX}}$ stands for the matrix with the same pattern as MATRIX and with entry values of 1.

```
CALL MC65_MATRIX_MULTIPLY_VECTOR(MATRIX,X,Y,INFO[,TRANS])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the matrix.

X is a REAL (double precision REAL for HSL_MC65_DOUBLE) or INTEGER rank-one array of INTENT (IN). It must be set by the user to hold the vector. If TRANS is absent or present with the value .FALSE., X is of size MATRIX%N; otherwise, it is of size MATRIX%M.

Y is a rank-one array of the same type as X and of INTENT (OUT). If TRANS is absent or present with the value .FALSE., Y is of size MATRIX%M; otherwise, it is of size MATRIX%N. On exit, Y is defined as follows

- If X and Y are REAL arrays, and TRANS is absent or present with the value .FALSE., Y = MATRIX*X; otherwise, Y = $(\texttt{MATRIX})^{\texttt{T}}$*X.
- If X and Y are INTEGER arrays, and TRANS is absent or present with the value .FALSE., Y = $\overline{\texttt{MATRIX}}$*X; otherwise, Y =$(\overline{\texttt{MATRIX}})^{\texttt{T}}$*X. Here $\overline{\texttt{MATRIX}}$ stands for the matrix with the same pattern as MATRIX and with entry values of 1.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully. INFO = MC65_ERR_MATVEC_NOVALUE if MATRIX is of type "pattern" but X is not an INTEGER array. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

TRANS is an OPTIONAL LOGICAL scalar of INTENT (IN). If present with the value .TRUE., the multiplication is performed with the transpose of the matrix.

3.9 Copying a matrix

Subroutine MC65_MATRIX_COPY creates a new sparse matrix MATRIX2 that is a copy of an existing sparse matrix MATRIX1. Storage is allocated for MATRIX2 and this storage should be deallocated when MATRIX2 is no longer used by calling subroutine MC65_MATRIX_DESTRUCT.

```
CALL MC65_MATRIX_COPY(MATRIX1,MATRIX2,INFO[,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the sparse matrix to be copied from.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit it holds the sparse matrix to be copied to.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully. INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed. INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.10 Matrix transpose

Subroutine MC65_MATRIX_TRANSPOSE computes the transpose of a sparse matrix MATRIX1 or its pattern. Storage is allocated for MATRIX2 and this storage should be deallocated when MATRIX2 is no longer used by calling subroutine MC65_MATRIX_DESTRUCT.

```
CALL MC65_MATRIX_TRANSPOSE(MATRIX1,MATRIX2,INFO[,MERGE,PATTERN,STAT])
```

MATRIX1 is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the matrix whose transpose is to be computed.

MATRIX2 is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit,

- If the optional argument PATTERN is present and is .TRUE., MATRIX2 is of the type "pattern" and contains the pattern of $(\texttt{MATRIX1})^{\texttt{T}}$.
- Otherwise, MATRIX2 will be of the same type as MATRIX1 and $\texttt{MATRIX2} = (\texttt{MATRIX1})^{\texttt{T}}$.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successfully; INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed; and INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

MERGE is an OPTIONAL LOGICAL scalar of INTENT (IN). If present, it must be set by the user. If MERGE = .TRUE, duplicate entries of MATRIX2 will be merged by summing the entry values; otherwise, no merge is performed.

PATTERN is an OPTIONAL LOGICAL scalar of INTENT (IN). If present, it must be set by the user. If PATTERN = .TRUE., MATRIX2 will be of type "pattern" and hold the pattern of $(\texttt{MATRIX1})^{\texttt{T}}$.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.11 Copying to another format

Subroutine MC65_MATRIX_TO_COO copies a sparse matrix MATRIX to a coordinate formatted matrix, specified by (M, N, NZ, IRN, ICN, VAL), where M, N and NZ are the numbers of rows, columns and entries, respectively, and IRN, JCN and VAL are allocatable arrays holding the row, column indices and entry values, respectively. Storage space for the allocatable arrays IRN, JCN and VAL are allocated inside the subroutine, and should be deallocated by the user when they are no longer used.

```
CALL MC65_MATRIX_TO_COO(MATRIX,M,N,NZ,IRN,JCN,INFO[,VAL,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the matrix.

M is an INTEGER scalar of INTENT (OUT). On exit, it holds the number of rows of MATRIX.

N is an INTEGER scalar of INTENT (OUT). On exit, it holds the the number of columns of MATRIX.

NZ is an INTEGER scalar of INTENT (OUT). On exit, it holds the number of entries of MATRIX.

IRN is an INTEGER allocatable array of rank one. A target of size NZ is allocated inside the subroutine for IRN, and on exit it holds the row indices.

JCN is an INTEGER allocatable array of rank one. A target of size NZ is allocated inside the subroutine for JCN, and on exit it holds the column indices.

VAL is an OPTIONAL REAL (double precision REAL for HSL_MC65_DOUBLE) allocatable array of rank one. When present, and the matrix is not of type "pattern", a target of size NZ is allocated for VAL, and on exit it holds the entry values; if the matrix is of type "pattern", VAL will be allocated a target of size 0.

INFO is an INTEGER scalar of INTENT (OUT). On exit, INFO = 0 if the subroutine completed successful. INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed. An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.12 Input and output of sparse matrices

3.12.1 Writing a sparse matrix to a file

Subroutine MC65_MATRIX_WRITE writes a sparse matrix to a file. By default, the file is formatted and matrix is written in the coordinate format (also known as triplet format). There are options for writing the pattern as a formatted Gnuplot data file or as a formatted file with a record for the indices of each column. There is an option for permuting the matrix before writing it in coordinate or Gnuplot format. There is an option for writing the whole matrix to an unformatted file.

If the Gnuplot option is used, a Gnuplot command is placed in FILE_NAME.com. The user needs to start Gnuplot and type load "FILE_NAME.com" to view the matrix.

We use M, N and NZ to denote the numbers of rows, columns and entries, respectively. We use $\{($i$_l$,j$_l$, a(i$_l$,j$_l$)), l $= 1,2,\ldots,$NZ$\}$ to denote the entries. We use TYPE to denote the matrix type with trailing blanks removed.

```
CALL MC65_MATRIX_WRITE(MATRIX,FILE_NAME,INFO[,UNIT,FORM,ROW_ORD,&
   COL_ORD,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (IN). It must be set by the user to hold the sparse matrix.

FILE_NAME is a CHARACTER (LEN=*) scalar with INTENT (IN). It must be set by the user to hold the file name. The file is written in one of the following ways.

- **Pattern in coordinate format.** This is the default for a matrix of type "pattern". The file is formatted and of the following form

  M N NZ
  i$_1$ j$_1$
  i$_2$ j$_2$
  ...
  i$_{NZ}$ j$_{NZ}$

- **Whole matrix in coordinate format.** This is the default for a matrix not of type "pattern". The file is formatted and of the following form

  M N NZ
  i$_1$ j$_1$ a(i$_1$,j$_1$)
  i$_2$ j$_2$ a(i$_2$,j$_2$)
  ...
  i$_{NZ}$ j$_{NZ}$ a(i$_{NZ}$,j$_{NZ}$)

- **Unformatted.** The file is unformatted and of the following form

  LEN_TRIM(TYPE)
  TYPE
  M N NZ
  MATRIX%PTR(1:NZ)
  MATRIX%COL(1:NZ)
  MATRIX%VAL(1:NZ) (omitted for a pattern-only matrix)

- **Gnuplot.** The file is formatted and in Gnuplot form.
- **Hypergraph.** The file is formatted and of the following form

  N M
  row_indices_of_column_1
  row_indices_of_column_2
  row_indices_of_column_3
  ...
  row_indices_of_column_NZ

FORM is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If present, it must be set by the user to

- "ij" for the pattern in coordinate format,
- "coordinate" or "ija" for the whole matrix in coordinate format,
- "unformatted" for unformatted output,
- "gnuplot" for Gnuplot output, or
- "hypergraph" for hypergraph output.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine completed successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation failed.
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation failed.
- INFO = MC65_ERR_NO_VACANT_UNIT if no vacant unit has been found.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

ROW_ORD is an OPTIONAL INTEGER rank-one array of INTENT (IN) and size MATRIX%M. If present, it must be set by the user to hold the permutation to be applied to the rows when writing in coordinate or Gnuplot format. Row I is row ROW_ORD(I) in the reordered matrix.

COL_ORD is an OPTIONAL INTEGER rank-one array of INTENT (IN) and size MATRIX%N. If present, it must be set by the user to hold the permutation to be applied to the columns when writing in coordinate or Gnuplot format. Column J is COL_ORD(J) in the reordered matrix.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.12.2 Reading in a sparse matrix

Subroutine MC65_MATRIX_READ reads in a matrix from a file. Storage is allocated in creating the MATRIX object and should be released by calling MC65_MATRIX_DESTRUCT when MATRIX is no longer needed.

```
CALL MC65_MATRIX_READ(MATRIX,FILE_NAME,INFO[,FORM,STAT])
```

MATRIX is a scalar of the derived type ZD11_TYPE with INTENT (INOUT). On entry the user does not need to set it. On exit it holds the matrix read from a file.

FILE_NAME is a CHARACTER (LEN=*) scalar with INTENT (IN). It must be set by the user to hold the name of the file from which the matrix is to be read.

INFO is an INTEGER scalar of INTENT (OUT). On exit,

- INFO = 0 if the subroutine returns successfully.
- INFO = MC65_ERR_MEMORY_ALLOC if memory allocation has failed;
- INFO = MC65_ERR_MEMORY_DEALLOC if memory deallocation has failed;
- INFO = MC65_ERR_NO_VACANT_UNIT if no vacant unit has been found.
- INFO = MC65_ERR_READ_FILE_MISS if the file <FILE_NAME> does not exist.
- INFO = MC65_ERR_READ_OPEN if the file <FILE_NAME> cannot be opened.
- INFO = MC65_ERR_READ_WRONGFORM if FORM specifies an unsupported format.

An error message can be printed by calling subroutine MC65_PRINT_MESSAGE.

FORM is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If present, it must be set by the user to hold the format of the input file. Only FORM = "unformatted" is supported, which corresponds to the unformatted output of MC65_MATRIX_WRITE. If FORM is not present, the subroutine assumes that the input file is unformatted.

STAT is an OPTIONAL INTEGER scalar of INTENT (OUT). On exit, it holds the Fortran STAT parameter.

3.13 Printing error/warning messages

Subroutine MC65_PRINT_MESSAGE can be used to print an error or warning message related to a particular information tag INFO.

```
CALL MC65_PRINT_MESSAGE(INFO[,STREAM,CONTEXT])
```

INFO is an INTEGER scalar of INTENT (IN). It must be set by the user to hold the information tag generated when calling a subroutine or function of HSL_MC65.

STREAM is an OPTIONAL INTEGER scalar of INTENT (IN). If present, it must be set by the user to hold the unit number for the message. If this number is negative, printing is suppressed. If STREAM is not present, the message will be printed on unit 6.

CONTEXT is an OPTIONAL CHARACTER (LEN=*) scalar with INTENT (IN). If present, it must be set by the user to provide the context under which the error/warning occurs. It is printed ahead of the error or warning message.

4 GENERAL INFORMATION

List of routines:

MC65_MATRIX_CLEAN merges repeated entries of a sparse matrix (Section 3.5.2).

MC65_MATRIX_CONDENSE merges columns in a sparse matrix of the same pattern into super-columns (Section 3.5.6).

MC65_MATRIX_CONSTRUCT constructs a sparse matrix object (Section 3.1).

MC65_MATRIX_COPY copies one sparse matrix to another (Section 3.9).

MC65_MATRIX_DESTRUCT releases the memory occupied by the sparse matrix object (Section 3.2).

MC65_MATRIX_DIAGONAL_FIRST moves the diagonal entry in each row of a sparse matrix to the first position within the row (Section 3.5.5).

MC65_MATRIX_GETROW associates a pointer array with the vector of column indices of a row of a sparse matrix (Section 3.6.1).

MC65_MATRIX_GETROWVAL associates a pointer array with the vector of entry values of a row of a sparse matrix (Section 3.6.2).

MC65_MATRIX_IS_DIFFERENT tests whether two matrices are different (Section 3.7.4).

MC65_MATRIX_IS_PATTERN returns .TRUE. if the matrix is of type "pattern", and .FALSE. if otherwise (Section 3.4.1).

MC65_MATRIX_IS_SYMMETRIC checks whether a matrix is symmetric (Section 3.4.2).

MC65_MATRIX_MULTIPLY multiplies two sparse matrices (Section 3.7.2).

MC65_MATRIX_MULTIPLY_GRAPH multiplies two sparse matrices, but with their entry values assumed to be one (Section 3.7.3). It is used for finding the row connectivity graph of a sparse matrix.

MC65_MATRIX_MULTIPLY_VECTOR multiplies a sparse matrix by a vector (Section 3.8).

MC65_PRINT_MESSAGE writes an error/warning message corresponding to a particular INFO tag (Section 3.13).

MC65_MATRIX_READ reads in a sparse matrix from a file (Section 3.12.2).

MC65_MATRIX_REALLOCATE reallocates the storage space for an existing sparse matrix object (Section 3.3).

MC65_MATRIX_REMOVE_DIAGONAL removes the diagonal elements of a sparse matrix (Section 3.5.4).

MC65_MATRIX_SORT sorts the column indices of each row to be in increasing order (Section 3.5.3).

MC65_MATRIX_SUM performs a sum or a SAXPY-like operation on two sparse matrices (Section 3.7.1).

MC65_MATRIX_SYMMETRIZE symmetrizes a sparse matrix by summing it with its transpose (Section 3.5.1).

MC65_MATRIX_TO_COO converts a sparse matrix into the coordinate formatted data (Section 3.11)

MC65_MATRIX_TRANSPOSE transposes a sparse matrix (Section 3.10).

MC65_MATRIX_WRITE writes out a sparse matrix to a file in various formats (Section 3.12.1).

Use of common: None.

Workspace: Provided automatically by the module.

Other routines called directly: None.

Other modules used directly: HSL_ZD11

Input/output: No input; No output unless the subroutine MC65_PRINT_MESSAGE is called. When this subroutine is called, by default an error/warning message is printed to unit 6. If the user supplies a unit number, the message is printed to the user supplied unit. However if this unit number is negative, printing is suppressed.

Changes from Version 1.0.0: HSL_ZD11 used instead of HSL_ZD01.

5 METHOD

6 EXAMPLE OF USE

We illustrate the use of the package by carrying out a number of operations on a 5×5 sparse matrix A using the following code. In the code, we will form AA^T and write it out in the file "AAT". The use of sorting, cleaning and copying functionality is also demonstrated. The 5×5 matrix A is

$$A = \begin{pmatrix} 1 & 0 & 0 & 6 & 0 \\ 0 & 10.4 & 0 & 0 & 1 \\ 0 & 0 & .015 & 0 & 0 \\ 0 & 250.5 & 0 & -280 & 33.32 \\ 0 & 3 & 0 & 0 & 12 \end{pmatrix}$$

For reference,

$$AA^T = \begin{pmatrix} 37 & 0 & 0 & -1680. & 0 \\ 0 & 109.2 & 0 & 2639. & 43.20 \\ 0 & 0 & 0.000225 & 0 & 0 \\ -1680. & 2639. & 0 & 142300 & 1151. \\ 0 & 43.20 & 0 & 1151. & 153 \end{pmatrix}$$

Program

```
program main
  use hsl_zd11_double
  use HSL_MC65_double
  implicit none

  INTEGER, PARAMETER :: myreal = KIND(1.0D+0)
  INTEGER, PARAMETER :: myint = KIND(1)
  real (kind = myreal), dimension (:), allocatable :: val
  integer (kind = myint), dimension (:), allocatable :: irn,jcn
  type (zd11_type) :: a,b,c,d
  integer  (kind = myint) :: m,n,nz,info
  integer  (kind = myint) :: unit = 6
  logical :: symmetric

  m = 5
  n = 5
  nz = 10

  allocate(val(nz),irn(nz),jcn(nz))


  irn = (/ 1,  1,    2,   2,      3,     4,     4,     4,  5,   5/)
  jcn = (/ 1,  4,    2,   5,      3,     2,     4,     5,  2,   5/)
  val = (/1., 6., 10.4,  1.,  0.015, 250.5, -280., 33.32, 3., 12./)

  unit = 6

  write(unit,"(a)") " MC65 example start ------- "

  ! construct A using coordinate format
  call MC65_matrix_construct(a,m,n,nz,irn,jcn,info,val,type = "general",&
       checking = 1)
  call MC65_print_message(info,unit,"MC65_matrix_construct")
  if (info < 0) stop

  ! clean A
  call MC65_matrix_clean(a,info)
```

```
  call MC65_print_message(info,unit,"MC65_matrix_clean")
  if (info < 0) stop

  ! sort A
  call MC65_matrix_sort(a,info)
  call MC65_print_message(info,unit,"MC65_matrix_sort")
  if (info < 0) stop

  ! B = A^T
  call MC65_matrix_transpose(a,b,info,merge=.true.)
  call MC65_print_message(info,unit,"MC65_matrix_transpose")
  if (info < 0) stop

  ! C = A*A^T, check that C is symmetric
  call MC65_matrix_multiply(a,b,c,info)
  call MC65_print_message(info,unit,"MC65_matrix_multiply")
  if (info < 0) stop
  call MC65_matrix_is_symmetric(c,symmetric,info,tol = real(1.0D-8,myreal))
  call MC65_print_message(info,unit,"MC65_matrix_symmetric")
  if (.not.symmetric) then
     stop
  else
     write(unit,"(a)") " A*A^T is confirmed to be symmetric"
  end if

  ! copy C to D
  call MC65_matrix_copy(c,d,info)
  call MC65_print_message(info,unit,"MC65_matrix_copy")
  if (info < 0) stop

  ! write out AA^T
  write(unit,"(a)") " writing A*A^T in AAT"
  call MC65_matrix_write(d,"AAT",info)
  call MC65_print_message(info,unit,"MC65_matrix_write")
  if (info < 0) stop

  ! destroy A, B, C, D
  call MC65_matrix_destruct(a,info)
  if (info < 0) stop "deallocation error"
  call MC65_matrix_destruct(b,info)
  if (info < 0) stop "deallocation error"
  call MC65_matrix_destruct(c,info)
  if (info < 0) stop "deallocation error"
  call MC65_matrix_destruct(d,info)
  if (info < 0) stop "deallocation error"

  deallocate(val,irn,jcn)

end program main
```

This produces the following output

```
 MC65 example start -------
MC65_matrix_construct : successful completion
MC65_matrix_clean : successful completion
MC65_matrix_sort : successful completion
MC65_matrix_transpose : successful completion
MC65_matrix_multiply : successful completion
MC65_matrix_symmetric : successful completion
 A*A^T is confirmed to be symmetric
MC65_matrix_copy : successful completion
 writing A*A^T in AAT
MC65_matrix_write : successful completion
```

The output file "AAT" is

```
5         5        13
1         1   37.00
1         4  -1680.
2         2   109.2
2         4   2639.
2         5   43.20
3         3  0.2250E-03
4         2   2639.
4         4  0.1423E+06
4         5   1151.
4         1  -1680.
5         2   43.20
5         4   1151.
5         5   153.0
```