Science and Technology Facilities Council

HSL_MC34

PACKAGE SPECIFICATION HSL

1 SUMMARY

This subroutine **generates the expanded structure for a matrix A with a symmetric sparsity pattern** given the structure for the lower triangular part. Diagonal entries need not be present.

ATTRIBUTES — Version: 1.1.0. **Types:** Integer, Real (single, double), Complex (single, double). **Date:** Original date: October 2009. **Origin:** J.A. Scott, Rutherford Appleton Laboratory. **Language:** Fortran 95.

2 HOW TO USE THE PACKAGE

2.1 Calling sequences

Access to the package requires a `USE` statement.

Integer version
```
USE HSL_MC34_integer
```
Single precision version
```
USE HSL_MC34_single
```
Double precision version
```
USE HSL_MC34_double
```
Complex version
```
USE HSL_MC34_complex
```
Double complex version
```
USE HSL_MC34_double_complex
```

2.2 Argument lists and calling sequences

2.2.1 Optional arguments

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

2.2.2 Package type

We use the term **package type** to mean default integer if the integer version is being used, default real if the single precision version is being used, double precision real for the double precision version, default complex for the complex version, and double precision complex for the double complex version.

2.2.3 To expand the matrix

```
CALL MC34_expand(n,row,ptr,iw[,a,sym_type])
```

`n` is a scalar `INTENT(IN)` of type `INTEGER` that holds the order of A.

row is a INTENT(INOUT) rank-one array of type INTEGER and size at least the number of entries in the expanded matrix (twice the value of ptr(n+1)-1 on entry is sufficient). row(1:ptr(n+1)-1) must be set by the user to hold the row indices of the lower triangular part of A. The entries of a single column must be contiguous. The entries of column j must precede those of column j+1 (j = 1, 2,..., n-1), and there must be no wasted space between columns. Row indices within a column may be in any order. On exit, it will have the same meaning but will be changed to hold the row indices of the entries in the expanded structure. Diagonal entries need not be present. The new row indices added in the upper triangular part will be in order for each column and will precede the row indices for the lower triangular part which will remain in the input order.

ptr is a INTENT(INOUT) rank-one array of type INTEGER and size n+1 that must be set by the user so that ptr(j) is the position in row of the first entry in column j (j = 1, 2,..., n) and ptr(n+1) must be set to one more than the total number of entries. On exit, ptr(j) will have the same meaning but will be changed to point to the position of the first entry of column j in the expanded structure. The new value of ptr(n+1) will be one greater than the number of entries in the expanded structure.

iw is a rank-one array of type INTEGER and size n that is used as workspace.

a is an optional INTENT(INOUT) rank-one array of package type and size at least the number of entries in the expanded matrix (twice the value of ptr(n+1)-1 on entry is sufficient). If present, a(1:ptr(n+1)-1) must be set by the user so that a(k) holds the value of the entry in row(k), (k = 1, 2,..., ptr(n+1)-1). On exit, a will hold the values of the entries in the expanded structure corresponding to the output values of row.

sym_type is an optional scalar INTENT(IN) of type INTEGER. If present, it may take the following values, with the following effects:

0 the returned matrix is symmetric ($A = A^T$).

1 the returned matrix is skew symmetric ($A = -A^T$).

2 the returned matrix is Hermitian ($A = A^\dagger$).

If sym_type is not present, or is present with any other value, the returned matrix is symmetric.

3 GENERAL INFORMATION

Workspace: iw is used as workspace.

4 METHOD

The total number of entries in each column in the expanded form is first calculated by scanning the input matrix once. The entries of the input matrix are then placed at the ends of the columns in the expanded form. Since the same arrays are reused, this copying starts with the last column. A pointer is maintained pointing to the first filled position in each column. A final scan through the moved parts of the columns is used to copy each entry a_{ij}, $i > j$ immediately in the correct position of its counterpart a_{ji}.

5 EXAMPLE OF USE

The following code may be used to expand the structure of a 4×4 sparse matrix with 5 entries in the lower triangular part.

```
program hsl_mc34ds
   use hsl_mc34_double

   integer :: iw(4),row(7),ptr(5)
   integer :: i,n,nz

   read (5,'(15i4)') n
   read (5,'(15i4)') (ptr(i),i=1,n+1)
   nz = ptr(n+1) - 1
   read (5,'(15i4)') (row(i),i=1,nz)

   call mc34_expand(n,row,ptr,iw)

   write (6,'(a,i3)') ' n = ',n
   write (6,'(a/a,15i4)') ' ptr .... ', ' ', (ptr(i),i=1,n+1)
   nz = ptr(n+1) - 1
   write (6,'(a/a,15i4)') ' row .... ', ' ',(row(i),i=1,nz)

end program hsl_mc34ds
```

With the input file:

```
4
1   3   4   5   6
1   3   4   3   4
```

the following output is produced:

```
n =   4
ptr ....
   1   3   4   6   8
row ....
   1   3   4   1   3   2   4
```