Science and Technology Facilities Council

HSL_MC78

PACKAGE SPECIFICATION HSL

1 SUMMARY

Given an elimination order, HSL_MC78 performs common tasks required in the **analyse phase of a symmetric sparse direct solver**. Either the entire analyse may be performed or individual tasks. No checking is performed on the validity of user data, and failure to supply valid data will result in undefined behaviour.

Given the sparsity pattern of a sparse symmetric matrix A and permutation P, HSL_MC78 finds the pattern of the Cholesky factor L such that $PAP^{-1} = LL^T$. The pattern of A may be provided in either assembled or elemental format. The permutation P is referred to as the *elimination (pivot) order*.

Assembled matrices are specified by listing, for each column of A, the indices for rows of that column that are non-zero. *Elemental matrices* are formed as the sum $\sum_{k=1}^{m} A^{(k)}$ of element matrices, where only the rows and columns of $A^{(k)}$ that correspond to variables in the kth element are non-zero.

To reduce the amount of matrix data read during the analysis, supervariables of A may be identified. A *supervariable* is a set of columns of A that have the same sparsity pattern.

An *elimination tree* is built that describes the structure of the Cholesky factor in terms of data dependence between pivotal columns. This allows permutations of the elimination order that do not affect the number of entries in L to be identified and allows fast algorithms to be used in determining the exact structure of L.

A *supernode* is a set of columns that have the same pattern in the matrix L. This pattern is stored as a single row list for each supernode. The condensed version of the elimination tree consisting of supernodes is referred to as the *assembly tree*. To increase efficiency in a subsequent factorization phase, supernodes may be merged through a supernode amalgamation heuristic.

HSL_MC78 supports the use of 2×2 and larger block pivots. They must be input as consecutive pivots in the elimination order. The identification and use of supervariables of A is also optionally supported, allowing the matrix data to be compressed giving a consequent increase in performance for some problems. However, the simultaneous use of supervariables and block pivots is not currently supported.

ATTRIBUTES — Version: 1.6.1 (15 April 2023). **Interfaces:** C, Fortran. **Types:** Integer, Long integer. **Original date:** October 2010. **Origin:** J. D. Hogg, Rutherford Appleton Laboratory. **Language:** Fortran 2003 subset (F95 + TR15581).

2 HOW TO USE THE PACKAGE

Access to the package requires a USE statement
Default integer version

```
USE HSL_MC78_integer
```

Long integer version

```
USE HSL_MC78_long_integer
```

If it is required to use more than one module at the same time, the derived type mc78_control and the subroutines mc78_postorder, mc78_supernodes, mc78_stats and mc78_optimize_locality must be renamed in one of the USE statements.

This package has two possible usage modes. The first is provided by the subroutine

- mc78_analyse that may be called to perform the full analyse phase. It outputs information for a factorization phase.

The second mode exposes individual components of the analyse phase and may be used to perform partial analysis of a matrix. The following subroutines are available to the user:

- mc78_supervars finds supervariables (columns of A with the same sparsity pattern) and modifies the elimination order so that all variables of a supervariable are consecutive. This modification will not increase the number of entries in L.
- mc78_compress_by_svar takes an assembled matrix and its list of supervariables and creates a compressed version of the matrix.
- mc78_etree determines the elimination tree of an assembled matrix for a given elimination order.
- mc78_elt_equiv_etree finds supervariables of an elemental matrix A, produces an *equivalent matrix* that has the same non-zero pattern in its Cholesky factor as A, and determines the elimination tree of A. Only the lower triangle of the equivalent matrix is returned.
- mc78_postorder postorders an elimination tree.
- mc78_col_counts computes column counts for the Cholesky factor L, given an assembled matrix and its postordered elimination tree.
- mc78_supernodes identifies (relaxed) supernodes of the Cholesky factor L from the elimination tree and column counts.
- mc78_stats determines various statistics about the Cholesky factor L given either an elimination or assembly tree and the associated column counts.
- mc78_row_lists finds the row indices associated with each supernode of the Cholesky factor L. These may optionally be sorted.
- mc78_optimize_locality reorders variables within each supernode in a fashion that attempts to improve cache locality in a later factorization phase.

2.1 The derived data types

For some subroutines, the user must employ the derived type defined by the module to declare a scalar of type mc78_control. The following pseudocode illustrates this.

```
use hsl_mc78_double
...
type (mc78_control) :: control
```

The components of mc78_control are described in Section 2.3.7.

2.2 Argument lists and calling sequences

2.2.1 Optional arguments

We use square brackets [ ] to indicate OPTIONAL arguments, which are always at the end of the argument list. Since we reserve the right to modify the argument list and 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**.

2.2.2 Integer types

INTEGER denotes default integer and INTEGER(long) denotes INTEGER(kind=selected_int_kind(18)).
Package type denotes default integer if the integer version is being used, and INTEGER(long) if the long version is being used.

2.2.3 To analyse a matrix

To analyse a matrix, including determining an assembly tree, amalgamating supernodes and finding row lists, the user may call the following routine
For an assembled matrix:

```
call mc78_analyse(n, ptr, row,            perm,          nnodes, sptr, sparent, rptr, &
   rlist, control, info[, stat, nfact, nflops, piv_size])
```

For a matrix in elemental form:

```
call mc78_analyse(n, nelt, starts, vars, perm, eparent, nnodes, sptr, sparent, rptr, &
   rlist, control, info[, stat, nfact, nflops, piv_size])
```

n is a scalar INTENT(IN) argument of type INTEGER that holds the matrix order (in the element case, this is equal to the largest integer used to index a variable).

ptr is an array INTENT(IN) argument of package type and size n+1. For each column i of *A*, ptr(i) must specify the position of the first row index of that column in row(:), and ptr(n+1)-1 must specify the total number of entries.

row is an array INTENT(IN) argument of type INTEGER and size ptr(n+1)-1. The row indices associated with column i of *A* must be in row(ptr(i):ptr(i+1)-1). Entries in both the lower and upper triangles must be supplied.

nelt is a scalar INTENT(IN) argument of type INTEGER that holds the number of elements.

starts is an array INTENT(IN) argument of package type and size nelt+1. For each element, starts(elt) must specify the position of the first variable of element elt in vars(:), and starts(nelt+1)-1 must specify the total number of entries in all elements.

vars is an array INTENT(IN) argument of type INTEGER and size starts(nelt+1)-1. The variables of element elt must be in vars(starts(elt):starts(elt+1)-1).

perm is an array INTENT(INOUT) argument of type INTEGER and size n. On entry, must be set to hold a permutation that specifies the elimination order. Variable i is pivoted on in position perm(i). On exit, it specifies a potentially modified elimination order to which the output factor data corresponds. Any modification reflects relabelling of the assembly tree.

eparent is an array INTENT(OUT) argument of type INTEGER and size nelt. On exit, eparent(i) specifies the supernode corresponding to the least pivot of element *i*. If element i is empty, eparent(i) will have a value greater than nnodes.

nnodes is a scalar INTENT(OUT) argument of type INTEGER. On exit, it specifies the number of supernodes.

sptr is a rank-one ALLOCATABLE array argument of INTENT(OUT) and type INTEGER. On exit, it is allocated to have size nnodes+1 and specifies the variables belonging to each supernode. Supernode i contains pivotal variables sptr(i) through sptr(i+1)-1. If the matrix is rank deficient, not all variables will be part of a supernode: pivots sptr(nnodes+1):n correspond to unused variables (i.e. empty columns).

sparent is a rank-one ALLOCATABLE array argument of INTENT(OUT) and type INTEGER. On exit, it is allocated to have size nnodes and specifies the assembly tree. sparent(i) is the parent of supernode i in the assembly tree. If supernode i is a root then sparent(i) is set to nnodes+1.

rptr is a rank-one ALLOCATABLE array argument of INTENT(OUT) and type INTEGER(long). On exit it is allocated to have size nnodes+1 and specifies the position of first row index of each supernode in rlist.

rlist is a rank-one ALLOCATABLE array argument of INTENT(OUT) and type INTEGER. On exit it is allocated to have size rptr(nnodes+1)-1 and specifies the row lists for each supernode. The indices associated with supernode i are given by rlist(rptr(i):rptr(i+1)-1). The row list indices refer to the elimination order rather than the original (matrix) order. If control%sort=.true., entries within each list are in ascending order.

control is an INTENT(IN) argument of type mc78_control, whose components control the behaviour of the algorithm, as described in Section 2.3.7.

info is an INTENT(OUT) argument of type INTEGER. On exit it contains a return code. For normal completion this is zero, however other values indicate a warning or error return as described in Section 2.4.

stat is an optional INTEGER(OUT) argument of type INTEGER. If present, then on exit it contains the returned stat parameter of the last call made to allocate or deallocate.

nfact is an optional INTENT(OUT) argument of type INTEGER(long). If present, on exit it contains the number of entries present in the factors assuming no modifications are made to the elimination order.

nflops is an optional INTENT(OUT) argument of type INTEGER(long). If present, on exit it contains the number of floating point operations required to compute the Cholesky factorization assuming no modifications are made to the elimination order.

piv_size is an optional INTENT(INOUT) array argument of type INTEGER and size n. If present, on entry it specifies block pivots to be used and on exit it specifies the same information modified to match any changes to perm. The value piv_size(i) gives the number of pivots in the block pivot containing variable i of A. If a block pivot contains an unused variable, then that variable will be removed from the block pivot and placed at the end of the elimination order. Note that in the current version of HSL_MC78, the use of block pivots can significantly increase the time and memory required for analysis.

2.2.4 To identify supervariables of an assembled matrix

To identify supervariables of an assembled matrix, the user may call the routine

```
call mc78_supervars(n, ptr, row, perm, invp, nsvar, svar, st)
```

n is a scalar INTENT(INOUT) argument of type INTEGER that holds the matrix order. On exit it specifies the number of variables that are actually used (i.e. non-empty columns).

ptr is an array INTENT(IN) argument of package type and size n+1. For each column i of A, ptr(i) must specify the position of the first row index of that column in row(:), and ptr(n+1)-1 must specify the total number of entries.

row is an array INTENT(IN) argument of type INTEGER and size ptr(n+1)-1. The row indices associated with column i of A are given by row(ptr(i):ptr(i+1)-1). Entries in both the lower and upper triangles must be supplied.

`perm` and `invp` are rank-one array `INTENT(INOUT)` arguments of type `INTEGER` and size `n`. On entry they describe a permutation and its inverse such that if row `i` is the `j`-th pivot, then `perm(i)=j`, and `invp(j)=i`. On exit, the permutation is rearranged such that all variables in a given supervariable are consecutive (and take a position in the elimination order equivalent to the first variable of the supervariable). Any unused variables (i.e. empty columns) are permuted to the end of the elimination order.

`nsvar` is a scalar `INTENT(OUT)` argument of type `INTEGER`. On exit, it specifies the number of supervariables identified.

`svar` is an array `INTENT(OUT)` argument of type `INTEGER` and size `n`. On exit, the first `nsvar` entries specify the number of variables in each supervariable.

`st` is a scalar `INTENT(OUT)` argument of type `INTEGER`. On successful completion of the subroutine it will have value `0`. Otherwise, an error occurred when attempting to allocate workspace, and instead the stat parameter of the failed allocation is returned.

2.2.5 To compress an assembled matrix using supervariables

To compress an assembled matrix using previously identified supervariables, to obtain the symmetric matrix (in full storage) with columns and rows corresponding to supervariables in elimination order, the user may call the routine

```
call mc78_compress_by_svar(n, ptr, row, invp, nsvar, svar, &
   ptr2, lrow2, row2, info, st)
```

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

`ptr` is an array `INTENT(IN)` argument of package type and size `n+1`. For each column `i` of A, `ptr(i)` must specify the position of the first row index of that column in `row(:)`, and `ptr(i+1)-1` must specify the last.

`row` is an array `INTENT(IN)` argument of type `INTEGER` and size `ptr(n+1)-1`. The row indices associated with column `i` of A are given by `row(ptr(i):ptr(i+1)-1)`. Entries in both the lower and upper triangles must be supplied.

`invp` is a rank-one array `INTENT(IN)` arguments of type `INTEGER` and size `n`. It describes an inverse permutation such that if row `i` is the `j`-th pivot, then `invp(j)=i`.

`nsvar` is a scalar `INTENT(IN)` argument of type `INTEGER`. It must hold the number of supervariables.

`svar` is an array `INTENT(IN)` argument of type `INTEGER` and size `nsvar`. It must hold the number of variables in each supervariable.

`ptr2` is an array `INTENT(OUT)` argument of package type and size `nsvar+1`. On exit, for each column `i` of the compressed matrix, `ptr2(i)` specifies the first row index of that column in `row2(:)`, and `ptr2(i+1)-1` specifies the last.

`lrow2` is a scalar `INTENT(IN)` argument of package type that specifies the length of the array `row2`. It should be sufficiently large that `row2` can store the row indices of the compressed matrix. An upper limit on the required size is `ptr(n+1)-1`.

`row2` is an array `INTENT(OUT)` argument of type `INTEGER` and size `lrow2`. On exit, the row indices associated with column `i` of the compressed matrix are given by `row2(ptr2(i):ptr2(i+1)-1)`. Columns are specified in elimination order and entries in both the lower and upper triangles are stored. If the size of `row2` is insufficient, an error is returned.

info is an INTENT(OUT) argument to type INTEGER. On exit, it contains a return code. For normal completion this is zero, however other values indicate a warning or error return as described in Section 2.4.

st is a scalar INTENT(OUT) argument of type INTEGER. It contains the stat value of the last allocate call executed by the subroutine.

2.2.6 To determine the elimination tree of an assembled matrix

To determine the elimination tree of a matrix under a given elimination order, the user may call the routine

```
call mc78_etree(n, ptr, row, perm, invp, parent, st)
```

n is a scalar INTENT(IN) argument of type INTEGER. It specifies the number of rows and columns in the matrix.

ptr is an array INTENT(IN) argument of package type and size n+1. For each column i of A, ptr(i) must specify the position of the first row index of that column in row(:), and ptr(i+1)-1 must specify the last.

row is an array INTENT(IN) argument of type INTEGER and size ptr(n+1)-1. The row indices associated with column i of A are given by row(ptr(i):ptr(i+1)-1). Entries in both the lower and upper triangles must be supplied.

perm and invp are rank-one array INTENT(IN) arguments of type INTEGER and size n. They describe a permutation and its inverse such that if row i is the j-th pivot, then perm(i)=j, and invp(j)=i.

parent is an array INTENT(OUT) argument of type INTEGER and size n. On exit, parent(i) specifies the parent of node i in the elimination tree, or has the value n+1 if node i is a root.

st is a scalar INTENT(OUT) argument of type INTEGER. On successful completion of the subroutine it will have value 0. Otherwise an error occurred when attempting to allocate workspace, and instead the stat parameter of the failed allocation is returned.

2.3 To find supervariables, equivalent matrix and elimination tree of an element problem

Supervariables of an *elemental matrix* are determined and the lower triangle of an assembled symmetric matrix is returned. The Cholesky factor of this assembled matrix under natural ordering will have the same pattern as the elemental matrix under the supplied ordering, but it is compressed using supervariables to use less space. As the lower triangular form is unsuitable for determining the elimination tree, the elimination tree is determined for the user during this construction.

```
call mc78_elt_equiv_etree(n, nelt, starts, vars, perm, invp, nsvar, svar, &
   ptr, row, eparent, parent, st[, block_pivots])
```

n is a scalar INTENT(INOUT) argument of type INTEGER. It specifies the matrix order. On exit it specifies the number of variables that are actually used.

nelt is a scalar INTENT(IN) argument of type INTEGER. It specifies the number of elements.

starts is an array INTENT(IN) argument of package type and size nelt+1. For each element, starts(elt) must specify the position of the first variable of element elt in vars(:), and starts(elt+1)-1 must specify the last.

vars is an array INTENT(IN) argument of type INTEGER and size starts(nelt+1)-1. The variables of element elt are given by vars(starts(elt):starts(elt+1)-1).

`perm` and `invp` are rank-one array `INTENT(INOUT)` arguments of type `INTEGER` and size `n`. On entry they describe a permutation and its inverse such that if row `i` is the `j`-th pivot, then `perm(i)=j`, and `invp(j)=i`. On exit, the permutation is rearranged such that all variables in a given supervariable are consecutive (and take a position in the elimination order equivalent to the first variable of the supervariable). Any unused variables are permuted to the end of the elimination order.

`nsvar` is a scalar `INTENT(OUT)` argument of type `INTEGER`. On exit, it contains the number of supervariables in the compressed form.

`svar` is an array `INTENT(OUT)` argument of type `INTEGER` and size `n`. On exit, the first `nsvar` entries will have been set such that `svar(i)` gives the number of variables in supervariable `i`. The supervariables are numbered by the order they appear in the input permutation. Thus the first `svar(1)` entries of `invp` give the variables in supervariable 1, the next `svar(2)` entries the variables of supervariable 2 and so forth.

`ptr` is an array `INTENT(OUT)` argument of package type and size `n+1`. On output it specifies the column pointers of a lower triangular assembled matrix whose Cholesky factor has the same non-zero pattern as the elemental input matrix. The matrix is compressed through the use of supervariables. For each supercolumn `i` of the matrix, `ptr(i)` specifies the position of the first superrow index of supercolumn *i* in `row(:)`, and `ptr(i+1)-1` specifies the last.

`row` is an array `INTENT(OUT)` argument of type `INTEGER` and size `starts(nelt+1)-1`. On output it specifies the row indices of a lower triangular assembled matrix whose Cholesky factor has the same non-zero pattern as the elemental input matrix. The matrix is compressed through the use of supervariables. The superrow indices associated with supercolumn `i` of A are given by `row(ptr(i):ptr(i+1)-1)`. Entries in only the lower triangle and not on the diagonal are supplied.

`eparent` is an array `INTENT(OUT)` argument of type `INTEGER` and size `nelt`. On exit, `eparent(i)` specifies the variable corresponding to the least pivot of element *i*. If element `i` is empty, `eparent(i)` will have a value greater than `n`.

`parent` is an array `INTENT(OUT)` argument of type `INTEGER` and size `nsvar`. On exit, `parent(i)` specifies the parent of node `i` in the elimination tree, or has the value `nsvar+1` if node `i` is a root.

`st` is a scalar `INTENT(OUT)` argument of type `INTEGER`. On successful completion of the subroutine it will have value `0`. Otherwise an error occurred when attempting to allocate workspace, and instead the stat parameter of the failed allocation is returned.

`block_pivots` is an optional array `INTENT(INOUT)` argument of type `INTEGER` and size `n`. If present, then on entry it indicates block pivots that will be amalgamated into the same supernode. `block_pivots(i)` corresponds to pivot `i`, and takes one of the following values:

`0` Pivot `i` is neither the first nor the last pivot of a block pivot.

`1` Pivot `i` is the first pivot of a block pivot.

`2` Pivot `i` is the last pivot of a block pivot.

`3` Pivot `i` is a 1×1 pivot.

On exit, the array will be modified to reflect any changes to `perm`. If a block pivot contains unused variables, then they are removed and placed at the end of the elimination order.

2.3.1 To postorder an elimination tree

Given a matrix, elimination order and the corresponding elimination tree the user may call the following routine to modify the elimination order such that the elimination tree becomes postordered. That is to say, the numbering of each node and its descendants is such that for each node b, there exists a minimal descendant a, and the set of all descendants of b is the sequence $a, a+1, \dots b-1$.

```
call mc78_postorder(n, perm, invp, parent, st[, block_pivots])
```

If the user additionally wishes to ensure any pivots corresponding to unused variables (i.e. empty columns) are moved to the end of the elimination order he or she may use a call of the following form:

```
call mc78_postorder(n, realn, ptr, perm, invp, parent, st[, block_pivots])
```

`n` is a scalar `INTENT(IN)` argument of type `INTEGER`. It specifies the order of the matrix.

`realn` is a scalar `INTENT(OUT)` argument of type `INTEGER`. On exit it specifies the number of variables that are actually used.

`ptr` is an array `INTENT(IN)` argument of package type and size `n+1`. The number of entries in column `i` of A must be equal to `ptr(i+1)-ptr(i)`.

`perm` and `invp` are rank-one array `INTENT(INOUT)` arguments of type `INTEGER` and size `n`. On entry they describe a permutation and its inverse such that if row `i` is the `j`-th pivot, then `perm(i)=j`, and `invp(j)=i`. On exit they are modified such that the elimination tree specified by `parent` is postordered.

`parent` is an array `INTENT(INOUT)` argument of type `INTEGER` and size `n`. On entry, `parent(i)` specifies the parent of node `i` in the elimination tree, or has the value `n+1` if node `i` is a root. On exit, it is modified to match the new ordering given by `perm`.

`st` is as described in Section 2.2.6.

`block_pivots` is an optional `INTENT(INOUT)` array argument of type `INTEGER` and size `n`. If present it will be modified to match the new ordering given by `invp`.

2.3.2 To determine column counts of a Cholesky factor

To determine the number of entries in each column of the Cholesky factor L of a matrix A given a postordering and associated elimination tree, the user may call the routine

```
call mc78_col_counts(n, ptr, row, perm, invp, parent, cc, st[, wt])
```

`n`, `ptr` and `row` are as described in Section 2.2.6.

`perm` and `invp` are rank-one array `INTENT(IN)` arguments of type `INTEGER` and size `n`. They describe a permutation and its inverse such that if row `i` is the `j`-th pivot, then `perm(i)=j`, and `invp(j)=i`.

`parent` is an array `INTENT(IN)` argument of type `INTEGER` and size `n`. It must describe a postordered elimination tree corresponding to the matrix described by `n`, `ptr` and `row` under the elimination order given by `perm`. The entry `parent(i)` must specify the parent of node `i`, or have the value `n+1` if node `i` is a root.

`cc` is an array `INTENT(OUT)` argument of type `INTEGER` and size `n+1`. On exit, `cc(i)` gives the number of entries in column L of the Cholesky factor of PAP^{-1}.

st is as described in Section 2.2.6.

wt is an optional array INTENT(IN) argument of type INTEGER and size n. If present, then column and row i of the matrix is treated as if it were in fact wt(i) rows or columns with the same sparsity pattern. If this argument is present, the column counts output in cc correspond to the number of rows in the uncompressed matrix.

2.3.3 To identify supernodes

To identify (relaxed) supernodes of a Cholesky factor L given by an elimination tree and associated column counts, the user may call the routine

```
call mc78_supernodes(n, realn, parent, cc, sperm, nnodes, sptr, sparent, scc, invp, &
   control, info, st[, wt, block_pivots])
```

n is a scalar INTENT(IN) argument of type INTEGER. It specifies the order of the matrix.

realn is a scalar INTENT(IN) argument of type INTEGER. It specifies the number variables that are used in the description of A (empty columns need not be included in any supernode).

parent is an array INTENT(IN) argument of type INTEGER and size n. It describes a postordered elimination tree, such that parent(i) specifies the parent of node i, or has the value n+1 if node i is a root.

cc is an array INTENT(IN) argument of type INTEGER and size n. The number of entries in column i of L is cc(i).

sperm is an array INTENT(OUT) argument of type INTEGER and size n. On exit, it contains a map from the elimination order to a new order where supernodes are contiguous. If sperm(i)=j, then pivot i is now pivot j.

nnodes is a scalar INTENT(OUT) argument of type INTEGER. On exit, it contains the number of supernodes found.

sptr is a rank-one array INTENT(OUT) argument of type INTEGER and size n+1. On exit, the first nnodes+1 elements will be set such that supernode i will consist of pivots sptr(i) through sptr(i+1)-1.

sparent is a rank-one array INTENT(OUT) argument of type INTEGER and size n. On exit, the first nnodes will describe the assembly tree such that sparent(i) specifies the parent of supernode i, or nnodes+1 if supernode i is a root.

scc is a rank-one array INTENT(OUT) argument of type INTEGER and size n. On exit, the first nnodes entries will be set such that scc(i) gives the number of rows in supernode i of L.

invp is a rank-one array INTENT(IN) arguments of type INTEGER and size n. It describes an inverse permutation such that if row i is the j-th pivot, then invp(j)=i.

control is a scalar INTENT(IN) argument of type mc78_control. Its components control the supernode amalgamation heuristics used by the routine, as described in Section 2.3.7.

info is an INTENT(INOUT) argument to type INTEGER. On exit, it contains a return code. For normal completion this is the same value as on entry, however a negative value indicates an error return as described in Section 2.4.

st is a scalar INTENT(OUT) argument of type INTEGER. It contains the stat value of the last allocate call executed by the subroutine.

wt is an optional array INTENT(IN) argument of type INTEGER and size n. If present, then column i of the matrix described by parent and cc is treated as if it were in fact wt(i) columns with the same sparsity pattern for the purposes of supernode amalgamation heuristics.

block_pivots is an optional array INTENT(IN) argument of type INTEGER and size n. If present, it indicates block pivots that will be amalgamated into the same supernode. block_pivots(i) corresponds to pivot i, and takes one of the following values:

0 Pivot i is neither the first nor the last pivot of a block pivot.

1 Pivot i is the first pivot of a block pivot.

2 Pivot i is the last pivot of a block pivot.

3 Pivot i is a 1×1 pivot.

2.3.4 To determine statistics about a Cholesky factor

To determine the number of entries in the Cholesky factor L and number of floating-point operations required to compute L, given the supernode distribution and associated row counts, the user may call the routine

```
call mc78_stats(nnodes, sptr, scc[, nfact, nflops])
```

nnodes is a scalar INTENT(IN) argument of type INTEGER. It specifies the number of supernodes.

sptr is a rank-one array INTENT(IN) argument of type INTEGER and size nnodes+1. It must be set so that supernode i consists of pivots sptr(i) through sptr(i+1)-1.

scc is a rank-one array INTENT(IN) argument of type INTEGER and size nnodes. It must be set so that supernode i of L has scc(i) rows.

nfact is an optional scalar INTENT(OUT) argument of type INTEGER(long). If present, on exit it contains the number of entries in the Cholesky factor L with the given supernode pattern.

nflops is an optional scalar INTENT(OUT) argument of type INTEGER(long). If present, on exit it contains the number of floating-point operations required to calculate the Cholesky factor L with the given supernode pattern.

2.3.5 To determine the row indices of a supernodal Cholesky factor

To determine the row lists for each supernode of a Cholesky factor L of the matrix A, the user may call the subroutine

```
call mc78_row_lists(n, ptr, row, perm, invp, nnodes, sptr, sparent, scc, rptr, rlist, &
                    control, info, st)
call mc78_row_lists(nsvar, svar, n, ptr, row, perm, invp, nnodes, sptr, sparent, scc, &
                    rptr, rlist, control, info, st)
```

The first call should be used for non-supervariable representations, while the second should be used when the matrix has been compressed using supervariables.

nsvar is a scalar INTENT(IN) argument of type INTEGER. It specifies the number of supervariables used to compress the matrix.

svar is an array INTENT(IN) argument of type INTEGER and size nsvar. It should be set such that supervariable i contains svar(i) variables.

n, ptr and row are as described in Section 2.2.6.

perm and invp are rank-one array INTENT(IN) arguments of type INTEGER and size n. They describe a permutation and its inverse such that if row i is the j-th pivot, then perm(i)=j, and invp(j)=i.

`nnodes` is a scalar `INTENT(IN)` argument of type `INTEGER`. It specifies the number of supernodes.

`sptr` is a rank-one array `INTENT(IN)` argument of type `INTEGER` and size `nnodes+1`. It must be set so that supernode `i` consists of pivots `sptr(i)` through `sptr(i+1)-1`.

`sparent` is a rank-one array `INTENT(IN)` argument of type `INTEGER` and size `nnodes`. It describes the assembly tree, such that `sparent(i)` specifies the parent of supernode `i`, or has the value `nnodes+1` if supernode `i` is a root.

`scc` is a rank-one array `INTENT(IN)` argument of type `INTEGER` and size `nnodes`. It must be set so that supernode `i` of L has `scc(i)` rows.

`rptr` is a rank-one array `INTENT(OUT)` argument of type `INTEGER(long)` and size `nnodes+1`. On exit, it specifies the positions in `rlist` that contain the row lists for each supernode.

`rlist` is a rank-one array `INTENT(OUT)` argument of type `INTEGER` and size `sum(scc(1:nnodes))`. On exit, the row list for supernode `i` is given by the array section `rlist(row(i):row(i+1)-1)`.

`control` is a scalar `INTENT(IN)` argument of type `mc78_control`. Its components control printing of error information from the routine, as described in Section 2.3.7.

`info` is an `INTENT(INOUT)` argument to type `INTEGER`. On exit, it contains a return code. For normal completion this is the same value as on entry, however a negative value indicates an error return as described in Section 2.4.

`st` is as described in Section 2.3.3.

2.3.6 To optimize variable ordering for cache locality

To permute variables within supernodes to improve cache locality in sparse update operations in a subsequent factorization phase, the user may call the routine

```
call mc78_optimize_locality(n, realn, perm, invp, nnodes, sptr, sparent, rptr, rlist, &
                            st[, sort])
```

`n`, `nnodes`, `sptr`, and `sparent` are as described in Section 2.3.5.

`realn` is a scalar `INTENT(IN)` argument of type `INTEGER`. It specifies the number of variables that are actually used in A (i.e. the number of non-empty columns).

`perm` and `invp` are rank-one array `INTENT(INOUT)` arguments of type `INTEGER` and size `n`. On entry they describe a permutation and its inverse such that if row `i` is the `j`-th pivot, then `perm(i)=j`, and `invp(j)=i`. On exit, they have been modified to give a new ordering with the same supernodes but better cache locality in the factorization phase.

`rptr` is a rank-one array `INTENT(IN)` argument of type `INTEGER(long)` and size `nnodes+1`. It specifies the positions in `rlist` that contain the row lists for each supernode.

`rlist` is a rank-one array `INTENT(INOUT)` argument of type `INTEGER` and size `sum(scc(1:nnodes))`. On entry, the row list for supernode `i` is given by the array section `rlist(row(i):row(i+1)-1)`. On exit each row list has been updated to match the new permutation.

`st` is as described in Section 2.2.6.

`sort` is an optional scalar `INTENT(IN)` argument of type `LOGICAL`. If present with the value `.true.`, the entries of each row list are returned in ascending order. Otherwise if `sort` is not present, or is present and has the value `.false.`, the entries of each row list may be in any order.

2.3.7 The derived data type for holding control parameters

The derived data type mc78_control is used to hold controlling data. The components, which are automatically given default values upon instantiation, are:

nemin is a scalar of type INTEGER that controls the node amalgamation heuristic. A node and its parent are merged if both have fewer than nemin columns. It has default value 16.

lopt is a scalar of type LOGICAL. If lopt=.true., mc78_analyse will reorder variables within each supernode to attempt to maximize cache locality in the factorize phase. If, in the call to mc78_analyse, the optional argument piv_size is present, pivots within blocks may be separated, but will remain within the same supernode. The default value is .false..

sort is a scalar of type LOGICAL. If sort=.true., then on return from a call to mc78_analyse, the entries of each supernode row list are sorted into ascending order. Otherwise, these values are returned in arbitrary order. The default value is .false..

ssa_abort is a scalar of type LOGICAL and controls the action when an assembled matrix is found to be symbolically singular (a row or column contains no entries) during a call to mc78_analyse. If ssa_abort=.true. an error is raised as soon as rank deficiency is detected. If ssa_abort=.false. a warning is raised, but computation then proceeds as normal, but with empty columns moved to the end of the elimination order. The default value is .false..

svar is a scalar of type LOGICAL. If svar=.true., mc78_analyse identifies and exploits supervariables. Otherwise supervariables are not exploited. If, in the call to mc78_analyse, the optional argument piv_sizes is present, then supervariables are not exploited regardless of the value of this control. The default value is .false..

unit_error is a scalar of type INTEGER and controls the printing of error messages. If positive, then errors are printed on the Fortran unit unit_error. Otherwise error messages are suppressed. The default value is 6.

unit_warning is a scalar of type INTEGER and controls the printing of warning messages. If positive, then warnings are printed on the Fortran unit unit_warning. Otherwise warning messages are suppressed. The default value is 6.

2.4 Warning and error messages

A successful return is indicated by info having the value zero. A negative value is associated with an error, and a positive value is associated with a warning.
Possible negative values are:

-1 A memory allocation error has occurred.

-2 Matrix is symbolically singular, assembled and control%ssa_abort=.true..

-3 The array row has insufficient size to store the new matrix.

Possible positive values are:

+1 Matrix is symbolically singular, assembled and control%ssa_abort=.false..

+2 Both supervariables and block pivots have been requested. These options are not compatible so only block pivots were used.

+3 Both warnings +1 and +2.

3 GENERAL INFORMATION

Workspace: Provided automatically by the module.

Other routines called directly: None.

Input/output: Warnings and errors are printed on units `control%unit_warning` and `control%unit_error` respectively. If the supplied units are negative output is suppressed.

Restrictions: n≥0

Portability: Fortran 95, plus allocatable components of derived types.

4 METHOD

What follows is broad overview of the algorithms used in this package. A more detailed description, including numerical results, is included in the following report:

J.D. Hogg and J.A. Scott. *A modern analyse phase for sparse tree-based direct methods.* RAL-TR-2010-031.

4.1 Analyse Method

`mc78_analyse` calls other subroutines in order to determine most of its information.

In the assembled case, if supervariables are requested, they are identified and the matrix is compressed. In the element case, supervariables are determined and a compressed equivalent matrix is built. In both cases, the elimination tree is then determined. The assembled case calls `mc78_supervars`, `mc78_compress_by_svar` and `mc78_etree` to do this. The elemental case combines its operations for efficiency in a single call to `mc78_elt_equiv_etree`.

The elimination tree is postordered using `mc78_postorder`, allowing a call to `mc78_col_counts` to find the column counts. This gives sufficient information to perform supernode amalgamation with `mc78_supernodes`. All that then remains is to call `mc78_row_lists` to determine the supernodal sparsity pattern.

Finally, information is expanded from the compressed form if necessary and statistics are calculated through a call to `mc78_stats` if requested. If `control%lopt=.true.` then cache locality optimizations are performed by `mc78_optimize_locality`. If `control%sort=.true.`, row lists are sorted using a double transpose sort.

No checking of data validity is performed at any stage.

4.2 Identifying supervariables

The identification of supervariables is done with an algorithm based on that described in [1]. It has been modified to reduce the amount of data movement involved with supervariables that contain only a single variable.

4.3 Determining the elimination tree

The elimination tree is determined through the use of an algorithm due to Liu [2]. This needs only to access the entries in the upper half of the matrix (once in elimination order) and entries in the lower part are ignored.

4.4 The combined call `mc78_elt_equiv_etree`

In the element case, the processes of identifying supervariables and building the equivalent matrix are combined. In fact two equivalent matrices are built: one lower and one upper triangular, both preordered so that the elimination order is $1,2,3,4,\ldots n$. The upper triangular matrix is used to find the elimination tree only, and we exploit the fact that we do not need to access the permutation in this case. The lower triangular matrix is returned for use in further analysis.

4.5 Postordering the elimination tree

The elimination tree is postordered through a simple depth-first search. It is designed such that the relative ordering of the children of each node is preserved.

4.6 Finding column counts

Column counts are found in time proportional to the number of non-zeros in A using the algorithm of Gilbert, Ng and Peyton [3]. We have specialised their original algorithm to find only column counts (the original found row counts also). The algorithm has been modified to allow for optionally weighting the columns in order to support supervariables.

4.7 Supernode amalgamation

We loop over the nodes of the elimination tree in depth first order. If either of the following conditions hold in the current assembly tree, then a node is merged into its parent:

- Merging the parent and child will not introduce additional non-zeros in the factors.
- Both the parent and child have less than `nemin` columns each.

4.8 Determining statistics

The number of entries in L and the number of floating-point operations to perform a Cholesky factorization may be calculated given the supernode partition and column counts associated with each supernode. From this we can determine for the column count for each column of L. If we denote this count for column i as $cc(i)$, then the number of entries in the factors is given by

$$nfact = \sum_{i=0}^{n} cc(i),$$

and the number of floating point operations is given by

$$nflops = \sum_{i=0}^{n} cc(i)^2.$$

4.9 Finding the row lists

Given a supernode partition and associated assembly tree we can perform a straightforward symbolic factorization to identify the row lists. With the supernodal column counts available we can avoid expensive data reorganisations. Working in a depth-first order, the non-zero set of a supernode is obtained by merging the non-zero sets of its children in the assembly tree and the non-zero sets of its associated columns in the original matrix A.

4.10 Optimizing for cache locality

Given a particular supernode partition (that is assignment of variables to supernodes), there is still a freedom in ordering the variables within a supernode. The subroutine `mc78_optimize_locality` exploits this freedom to increase the cache locality in a future numerical factorization phase by maximizing the number of entries that are are used contiguously in sparse expansion operations.

The algorithm used to do this is as follows. A depth-first search order is established on the tree such that of the children at each node are ordered by the number of variables they pass to the parent, with the largest first. Loop over supernodes in this order, and examine variables present. If a variable is being encountered for the first time, order it in the first available position corresponding to its supernode.

4.11 References

[1] I.S. Duff and J.K. Reid. 1996. *Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric linear systems.* ACM TOMS 22, 2. pp227-257.

[2] Liu, J. W. 1986. *A compact row storage scheme for Cholesky factors using elimination trees.* ACM TOMS 12, 2. pp127–148.

[3] Gilbert, Ng, Peyton. 1994. *An efficient algorithm to compute row and column counts for sparse Cholesky factorization.* SIMAX 15, 4.

5 EXAMPLE OF USE

5.1 Assembled matrix example

To analyse an assembled matrix and obtain the size of the factors and number of floating point operations the user may use the following code

```
program hsl_mc78is
   use hsl_mc78_integer
   implicit none

   integer, parameter :: long = selected_int_kind(18)

   integer :: i, n, nnodes, info
   integer(long) :: nfact, nflops
   integer, dimension(:), allocatable :: ptr, row, perm, sptr, sparent, rlist
   integer(long), dimension(:), allocatable :: rptr
   type(mc78_control) :: control

   ! Read matrix from standard input
   read (*, "(i8)") n          ! Matrix dimension
   allocate(ptr(n+1), perm(n))
   read (*, "(5i8)") ptr      ! Column pointers
   allocate(row(ptr(n+1)-1))
   read (*, "(5i8)") row      ! Row indices

   ! Use identity as pivot order
   forall(i=1:n) perm(i) = i
```

```
   ! Perform analysis
   control%nemin = 1 ! Disable supernode amalgamation for such a small matrix
   call mc78_analyse(n, ptr, row, perm, nnodes, sptr, sparent, rptr, rlist, &
      control, info, nfact=nfact, nflops=nflops)
   if(info.lt.0) then
      write (*, "(a,i8)") "mc78_analyse returned with unexpected error ", info
      stop
   endif

   ! Print out results
   do i = 1, nnodes
      write(*,"(3(a,i4),a)") &
         "Node ", i, " (columns ", sptr(i), " to ", sptr(i+1)-1, ")"
      write(*,"(3x,a,i4)") "parent in assembly tree is ", sparent(i)
      write(*,"(3x,a)") "row list is:"
      write(*,"(3x,8i8)") rlist(rptr(i):rptr(i+1)-1)
   end do
   write(*, "()")
   write(*, "(a, i8)") "Total number of entries in factor = ", nfact
   write(*, "(a, i8)") "Total number of floating point operations = ", nflops
end program hsl_mc78is
```

For the matrix

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

the following input

```
       5
       1       3       6       9      11
      14
       1       3       2       3       5
       1       2       3       4       5
       2       4       5
```

is suitable and yields the following output

```
Node    1 (columns    1 to    1)
   parent in assembly tree is    3
   row list is:
          1       4
Node    2 (columns    2 to    2)
   parent in assembly tree is    3
   row list is:
          2       5
Node    3 (columns    3 to    5)
   parent in assembly tree is    4
```

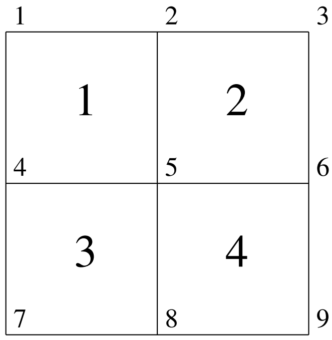
```
   row list is:
          3       4       5

Total number of entries in factor =       10
Total number of floating point operations =       22
```

5.2 Element case example

To analyse an elemental matrix with a block pivot $(4,5)$ the user may use the following code

```
program hsl_mc78is1
   use hsl_mc78_integer
   implicit none

   integer, parameter :: long = selected_int_kind(18)

   integer :: i, n, nelt, nnodes, info
   integer, dimension(:), allocatable :: starts, vars, perm, eparent, sptr, &
      sparent, rlist, piv_size
   integer(long), dimension(:), allocatable :: rptr
   type(mc78_control) :: control

   ! Read matrix from standard input
   read (*, "(2i8)") n,nelt   ! Matrix dimension and number of elements
   allocate(starts(nelt+1), perm(n), eparent(nelt), piv_size(n))
   read (*, "(5i8)") starts   ! Element pointers
   allocate(vars(starts(nelt+1)-1))
   read (*, "(5i8)") vars     ! Element variables

   ! Use identity as pivot order and define (4,5) as a block pivot
   forall(i=1:n) perm(i) = i
   piv_size(:) = 1
   piv_size(4:5) = 2

   ! Perform analysis
   control%nemin = 1 ! Disable supernode amalgamation for such a small matrix
   call mc78_analyse(n, nelt, starts, vars, perm, eparent, nnodes, sptr, &
      sparent, rptr, rlist, control, info, piv_size=piv_size)
   if(info.lt.0) then
      write (*, "(a,i8)") "mc78_analyse returned with unexpected error ", info
      stop
   endif

   ! Print out results
   do i = 1, nnodes
      write(*,"(3(a,i4),a)") &
         "Node ", i, " (columns ", sptr(i), " to ", sptr(i+1)-1, ")"
      write(*,"(3x,a,i4)") "parent in assembly tree is ", sparent(i)
      write(*,"(3x,a)") "row list is:"
      write(*,"(3x,8i8)") rlist(rptr(i):rptr(i+1)-1)
   end do
   do i = 1, nelt
      write(*,"(2(a,i4))") "Element ", i, " is a child of node ", eparent(i)
   end do
end program hsl_mc78is1
```

For the element problem in the following picture,



with variables at nodes 3, 6 and 9 fixed, the following input

```
       8       4
       1       5       7      11      13
       1       2       4       5       2
       5       4       5       7       8
       5       8
```

is suitable and yields the following output

```
Node    1 (columns    1 to    2)
   parent in assembly tree is    2
   row list is:
          1       2       3       4
Node    2 (columns    3 to    6)
   parent in assembly tree is    3
   row list is:
          3       4       5       6
Element    1 is a child of node    1
Element    2 is a child of node    1
Element    3 is a child of node    2
Element    4 is a child of node    2
```