Science and Technology Facilities Council

HSL_KB22

PACKAGE SPECIFICATION HSL

1 SUMMARY

HSL_KB22 is a suite of Fortran 95 procedures for successively arranging a set of real numbers, $\{a_1, a_2, ..., a_n\}$, in increasing order using the Heapsort method of J. W. J. Williams. At the k-th stage of the method, the k-th smallest member of the set is found (where 'smallest' means 'most negative' if negative numbers are present). The method is particularly appropriate if it is not known in advance how many smallest members of the set will be required as the Heapsort method is able to calculate the $k+1$-st smallest member of the set efficiently once it has determined the first k smallest members. The method is guaranteed to sort all n numbers in $O(n \log n)$ operations. If a complete sort is required, the Quicksort algorithm, KB05, may be preferred.

ATTRIBUTES — Version: 1.0.0. **Types:** Real (single, double), Integer (default, long). **Remark:** Supersedes HSL_KB12. **Date:** September 2007. **Origin:** N. I. M. Gould, Rutherford Appleton Laboratory, and Ph. L. Toint, University of Namur, Belgium. **Language:** Fortran 95.

2 HOW TO USE THE PACKAGE

Access to the package requires a USE statement such as

Single precision version

```
USE HSL_KB22_single
```

Double precision version

```
USE HSL_KB22_double
```

Default integer version

```
USE HSL_KB22_integer
```

Long integer version

```
USE HSL_KB22_long_integer
```

2.1 Argument lists and calling sequences

The method works by rearranging the elements into a so-called "heap", removing the smallest element from the heap and then forming another heap with the remaining elements. To form the initial heap, the user must first make a call to KB22_build_heap. The smallest element in the heap may be recorded and removed, and the remaining elements reordered to form a heap with one fewer element by calling KB22_get_smallest. Further calls to KB22_get_smallest may be used to record further remaining smallest elements.

In the following the **package type** denotes

Default REAL in HSL_KB22_single

DOUBLE PRECISION in HSL_KB22_double

Default INTEGER in HSL_KB22_integer

INTEGER(kind = selected_int_kind(18)) in HSL_KB22_long_integer

We use square brackets [ ] to indicate OPTIONAL arguments.

2.1.1 Forming the initial heap

The initial heap is constructed as follows:

```
CALL KB22_build_heap( n, A, inform [, INDA ] )
```

n is a scalar INTENT(IN) argument of type default INTEGER, that must be set by the user to n, the number of entries of A that are to be (partially) sorted. **Restriction:** n > 0.

A is a rank-one INTENT(INOUT) array argument of dimension at least n and of the desired package type, that must be set by the user on input so that its first n elements contain the values $a_1, a_2, ..., a_n$. On successful return, the elements of A will have been permuted so that they form a heap.

inform is a scalar INTENT(OUT) argument of type default INTEGER. A successful call to KB22_build_heap is indicated when inform has the value 0 on exit. For other return values of inform, see Section 2.2.

INDA is an optional rank-one INTENT(INOUT) array argument of dimension at least n and type default INTEGER. If INDA is present, exactly the same permutation is applied to the elements of INDA as to the elements of A. For example, the permutation will be provided if INDA(i) is set to i, for $i = 1, \ldots, n$ on entry.

2.1.2 Finding the smallest entry in the current heap

To find the smallest entry in a given heap, to place this entry at the end of the list of entries in the heap and to form a new heap with the remaining entries:

```
CALL KB22_get_smallest ( m, A, inform [, INDA ] )
```

m is a scalar INTENT(IN) argument of type default INTEGER, that must be set by the user to m, the number of entries of A that lie on the heap on entry. **Restriction:** m > 0.

A is a rank-one INTENT(INOUT) array argument of dimension at least m and of the desired package type, that must be set by the user on input so that its first m elements contain the values $a_1, a_2, ..., a_m$, ordered so that the entries lie on a heap. In practice, this normally means that they have been placed on a heap by a previous call to KB22_build_heap or KB22_get_smallest. On output, the smallest component of the first m elements of A will have been moved to position A(m) and the remaining elements will now occupy locations $1, 2, ...$ m-1 of A and will again form a heap.

inform is a scalar INTENT(OUT) argument of type default INTEGER. A successful call to KB22_get_smallest is indicated when inform has the value 0 on exit. For other return values of inform, see Section 2.2.

INDA is an optional rank-one INTENT(INOUT) array argument of dimension at least m and type default INTEGER. If INDA is present, exactly the same permutation is applied to the elements of INDA as to the elements of A.

2.1.3 Finding the k smallest components of a set of n elements

To find the k smallest components of a set, $\{a_1, a_2, ..., a_n\}$, of n elements, the user should firstly call KB22_build_heap with n $= n$ and a_1 to a_n stored in A(1) to A(n). This places the components of A on a heap. This should then be followed by k calls of KB22_get_smallest, with m $= n - i + 1$ for $i = 1, ..., k$. The required k smallest values, in increasing order, will now occupy positions $n - i + 1$ of A for $i = 1, ..., k$.

2.2 Warning and error messages

A positive value of inform on exit from KB22_solve or KB22_terminate indicates that an input error has occurred. The other arguments will not have been altered. The only possible value is:

1. One of the restrictions n > 0 (KB22_build_heap) or m > 0 (KB22_get_smallest) has been violated. The heap will not have been formed.

3 GENERAL INFORMATION

Use of common: None.

Workspace: None.

Other routines called directly: None.

Other modules used directly: None.

Input/output: None.

Restrictions: `n` > 0 (`KB22_build_heap`) and `m` > 0 (`KB22_get_smallest`).

Portability: ISO Fortran 95.

4 METHOD

The Heapsort method is due to J. W. J. Williams (Algorithm 232, Communications of the ACM **7** (1964), 347-348). Subroutine `KB22_build_heap` is a partial amalgamation of Williams' Algol procedures *setheap* and *inheap* while `KB22_get_smallest` is based upon his procedures *outheap* and *swopheap*.

The elements of the set $\{a_1, a_2, ..., a_n\}$ are first allocated to the nodes of a heap. A heap is a binary tree in which the element at each parent node has a numerical value as small as or smaller than the elements at its two children. The smallest value is thus placed at the root of the tree. This value is now removed from the heap and a subset of the remaining elements interchanged until a new, smaller, heap is constructed. The smallest value of the new heap is now at the root and may be removed as described above. The elements of the initial set may thus be arranged in order of increasing size, the i-th largest element of the array being found in the i-th sweep of the method. The method is guaranteed to sort all n numbers in $O(n \log n)$ operations.

5 EXAMPLE OF USE

As a simple example, suppose we wish to find the 12 smallest elements of the set

$$\{a_1, a_2, ..., a_{20}\} = \{-5, -7, 2, 9, 0, -3, 3, 5, -2, -6, 8, 7, -1, -8, 10, -4, 6, -9, 1, 4\}.$$

Then we may use the following code

```
PROGRAM HSL_KB22_EXAMPLE
USE HSL_KB22_double                        ! double precision version
IMPLICIT NONE
INTEGER, PARAMETER :: wp = KIND( 1.0D+0 ) ! set precision
INTEGER, PARAMETER :: n = 20
INTEGER :: i, m, inform
INTEGER :: INDA( n )
REAL ( KIND = wp ) :: A( n )
A = (/ -5.0, -7.0, 2.0, 9.0, 0.0, -3.0, 3.0, 5.0, -2.0, -6.0,                &
       8.0, 7.0, -1.0, -8.0, 10.0, -4.0, 6.0, -9.0, 1.0, 4.0 /) ! values
INDA = (/ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,                     &
          15, 16, 17, 18, 19, 20 /) ! indices
CALL KB22_build_heap( n, A, inform, INDA ) !  Build the heap
```

```
DO i = 1, 12
  m = n - i + 1
  CALL KB22_get_smallest( m, A, inform, INDA ) !  Reorder the variables
  WRITE( 6, "( ' The ', I2, '-th(-st) smallest value, a(', I2, ') is ',       &
 &       F5.1 ) " ) i, INDA( m ), A( m )
END DO
STOP
END PROGRAM HSL_KB22_EXAMPLE
```

This produces the following output:

```
The  1-th(-st) smallest value, a(18) is  -9.0
The  2-th(-st) smallest value, a(14) is  -8.0
The  3-th(-st) smallest value, a( 2) is  -7.0
The  4-th(-st) smallest value, a(10) is  -6.0
The  5-th(-st) smallest value, a( 1) is  -5.0
The  6-th(-st) smallest value, a(16) is  -4.0
The  7-th(-st) smallest value, a( 6) is  -3.0
The  8-th(-st) smallest value, a( 9) is  -2.0
The  9-th(-st) smallest value, a(13) is  -1.0
The 10-th(-st) smallest value, a( 5) is   0.0
The 11-th(-st) smallest value, a(19) is   1.0
The 12-th(-st) smallest value, a( 3) is   2.0
```