

NAG Library Routine Document

H05AAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

Given a set of m features and a scoring mechanism for any subset of those features, H05AAF selects the best n subsets of size p using a reverse communication branch and bound algorithm.

2 Specification

```

SUBROUTINE H05AAF (IREVCM, MINCR, M, IP, NBEST, DROP, LZ, Z, LA, A,      &
                  BSCORE, BZ, MINCNT, GAMMA, ACC, ICOMM, LICOMM, RCOMM,  &
                  LRCOMM, IFAIL)
INTEGER           IREVCM, MINCR, M, IP, NBEST, DROP, LZ, Z(M-IP), LA,  &
                  A(max(NBEST,M)), BZ(M-IP,NBEST), MINCNT,              &
                  ICOMM(LICOMM), LICOMM, LRCOMM, IFAIL
REAL (KIND=nag_wp) BSCORE(max(NBEST,M)), GAMMA, ACC(2), RCOMM(LRCOMM)

```

3 Description

Given $\Omega = \{x_i : i \in \mathbb{Z}, 1 \leq i \leq m\}$, a set of m unique features and a scoring mechanism $f(S)$ defined for all $S \subseteq \Omega$ then H05AAF is designed to find $S_{o1} \subseteq \Omega, |S_{o1}| = p$, an optimal subset of size p . Here $|S_{o1}|$ denotes the cardinality of S_{o1} , the number of elements in the set.

The definition of the optimal subset depends on the properties of the scoring mechanism, if

$$f(S_i) \leq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \quad (1)$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\text{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

else if

$$f(S_i) \geq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \quad (2)$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\text{minimize}} f(S) \quad \text{subject to} \quad |S| = p.$$

If neither of these properties hold then H05AAF cannot be used.

As well as returning the optimal subset, S_{o1} , H05AAF can return the best n solutions of size p . If S_{oi} denotes the i th best subset, for $i = 1, 2, \dots, n-1$, then the $(i+1)$ th best subset is defined as the solution to either

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\text{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

or

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\text{minimize}} f(S) \quad \text{subject to} \quad |S| = p$$

depending on the properties of f .

The solutions are found using a branch and bound method, where each node of the tree is a subset of Ω . Assuming that (1) holds then a particular node, defined by subset S_i , can be trimmed from the tree if $f(S_i) < \hat{f}(S_{on})$ where $\hat{f}(S_{on})$ is the n th highest score we have observed so far for a subset of size p , i.

e., our current best guess of the score for the n th best subset. In addition, because of (1) we can also drop all nodes defined by any subset S_j where $S_j \subseteq S_i$, thus avoiding the need to enumerate the whole tree. Similar short cuts can be taken if (2) holds. A full description of this branch and bound algorithm can be found in Ridout (1988).

Rather than calculate the score at a given node of the tree H05AAF utilizes the fast branch and bound algorithm of Somol *et al.* (2004), and attempts to estimate the score where possible. For each feature, x_i , two values are stored, a count c_i and $\hat{\mu}_i$, an estimate of the contribution of that feature. An initial value of zero is used for both c_i and $\hat{\mu}_i$. At any stage of the algorithm where both $f(S)$ and $f(S - \{x_i\})$ have been calculated (as opposed to estimated), the estimated contribution of the feature x_i is updated to

$$\frac{c_i \hat{\mu}_i + [f(S) - f(S - \{x_i\})]}{c_i + 1}$$

and c_i is incremented by 1, therefore at each stage $\hat{\mu}_i$ is the mean contribution of x_i observed so far and c_i is the number of observations used to calculate that mean.

As long as $c_i \geq k$, for the user-supplied constant k , then rather than calculating $f(S - \{x_i\})$ this routine estimates it using $\hat{f}(S - \{x_i\}) = f(S) - \gamma \hat{\mu}_i$ or $\hat{f}(S) - \gamma \hat{\mu}_i$ if $f(S)$ has been estimated, where γ is a user-supplied scaling factor. An estimated score is never used to trim a node or returned as the optimal score.

Setting $k = 0$ in this routine will cause the algorithm to always calculate the scores, returning to the branch and bound algorithm of Ridout (1988). In most cases it is preferable to use the fast branch and bound algorithm, by setting $k > 0$, unless the score function is iterative in nature, i.e., $f(S)$ must have been calculated before $f(S - \{x_i\})$ can be calculated.

4 References

- Narendra P M and Fukunaga K (1977) A branch and bound algorithm for feature subset selection *IEEE Transactions on Computers* **9** 917–922
- Ridout M S (1988) Algorithm AS 233: An improved branch and bound algorithm for feature subset selection *Journal of the Royal Statistics Society, Series C (Applied Statistics) (Volume 37)* **1** 139–147
- Somol P, Pudil P and Kittler J (2004) Fast branch and bound algorithms for optimal feature selection *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume 26)* **7** 900–912

5 Arguments

Note: this routine uses **reverse communication**. Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **IREVCM**. Between intermediate exits and re-entries, **all arguments other than BSCORE must remain unchanged**.

1: IREVCM – INTEGER Input/Output

On initial entry: must be set to 0.

On intermediate exit: IREVCM = 1 and before re-entry the scores associated with LA subsets must be calculated and returned in BSCORE.

The LA subsets are constructed as follows:

DROP = 1

The j th subset is constructed by *dropping* the features specified in the first LZ elements of Z and the single feature given in A(j) from the full set of features, Ω . The subset will therefore contain $M - LZ - 1$ features.

DROP = 0

The j th subset is constructed by *adding* the features specified in the first LZ elements of Z and the single feature specified in A(j) to the empty set, \emptyset . The subset will therefore contain $LZ + 1$ features.

In both cases the individual features are referenced by the integers 1 to M with 1 indicating the first feature, 2 the second, etc., for some arbitrary ordering of the features. The same ordering must be used in all calls to H05AAF.

If LA = 0, the score for a single subset should be returned. This subset is constructed by adding or removing only those features specified in the first LZ elements of Z.

If LZ = 0, this subset will either be Ω or \emptyset .

The score associated with the j th subset must be returned in BSCORE(j).

On intermediate re-entry: IREVCM must remain unchanged.

On final exit: IREVCM = 0, and the algorithm has terminated.

Constraint: IREVCM = 0 or 1.

2: MINCR – INTEGER *Input*

On entry: flag indicating whether the scoring function f is increasing or decreasing.

MINCR = 1

$f(S_i) \leq f(S_j)$, i.e., the subsets with the largest score will be selected.

MINCR = 0

$f(S_i) \geq f(S_j)$, i.e., the subsets with the smallest score will be selected.

For all $S_j \subseteq \Omega$ and $S_i \subseteq S_j$.

Constraint: MINCR = 0 or 1.

3: M – INTEGER *Input*

On entry: m , the number of features in the full feature set.

Constraint: $M \geq 2$.

4: IP – INTEGER *Input*

On entry: p , the number of features in the subset of interest.

Constraint: $1 \leq IP \leq M$.

5: NBEST – INTEGER *Input*

On entry: n , the maximum number of best subsets required. The actual number of subsets returned is given by LA on final exit. If on final exit LA \neq NBEST then IFAIL = 53 is returned.

Constraint: NBEST ≥ 1 .

6: DROP – INTEGER *Input/Output*

On initial entry: DROP need not be set.

On intermediate exit: flag indicating whether the intermediate subsets should be constructed by dropping features from the full set (DROP = 1) or adding features to the empty set (DROP = 0). See IREVCM for details.

On intermediate re-entry: DROP must remain unchanged.

On final exit: DROP is undefined.

7: LZ – INTEGER *Input/Output*

On initial entry: LZ need not be set.

On intermediate exit: the number of features stored in Z.

On intermediate re-entry: LZ must remain unchanged.

On final exit: LZ is undefined.

- 8: $Z(M - IP)$ – INTEGER array *Input/Output*

On initial entry: Z need not be set.

On intermediate exit: $Z(i)$, for $i = 1, 2, \dots, LZ$, contains the list of features which, along with those specified in A, define the subsets whose score is required. See IREVCM for additional details.

On intermediate re-entry: Z must remain unchanged.

On final exit: Z is undefined.

- 9: LA – INTEGER *Input/Output*

On initial entry: LA need not be set.

On intermediate exit: if $LA > 0$, the number of subsets for which a score must be returned.

If $LA = 0$, the score for a single subset should be returned. See IREVCM for additional details.

On intermediate re-entry: LA must remain unchanged.

On final exit: the number of best subsets returned.

- 10: $A(\max(\text{NBEST}, M))$ – INTEGER array *Input/Output*

On initial entry: A need not be set.

On intermediate exit: $A(j)$, for $j = 1, 2, \dots, LA$, contains the list of features which, along with those specified in Z, define the subsets whose score is required. See IREVCM for additional details.

On intermediate re-entry: A must remain unchanged.

On final exit: A is undefined.

- 11: $\text{BSCORE}(\max(\text{NBEST}, M))$ – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: BSCORE need not be set.

On intermediate exit: BSCORE is undefined.

On intermediate re-entry: $\text{BSCORE}(j)$ must hold the score for the j th subset as described in IREVCM.

On final exit: holds the score for the LA best subsets returned in BZ.

- 12: $\text{BZ}(M - IP, \text{NBEST})$ – INTEGER array *Input/Output*

On initial entry: BZ need not be set.

On intermediate exit: BZ is used for storage between calls to H05AAF.

On intermediate re-entry: BZ must remain unchanged.

On final exit: the j th best subset is constructed by dropping the features specified in $\text{BZ}(i, j)$, for $i = 1, 2, \dots, M - IP$ and $j = 1, 2, \dots, LA$, from the set of all features, Ω . The score for the j th best subset is given in $\text{BSCORE}(j)$.

- 13: MINCNT – INTEGER *Input*

On entry: k , the minimum number of times the effect of each feature, x_i , must have been observed before $f(S - \{x_i\})$ is estimated from $f(S)$ as opposed to being calculated directly.

If $k = 0$ then $f(S - \{x_i\})$ is never estimated. If $\text{MINCNT} < 0$ then k is set to 1.

- 14: GAMMA – REAL (KIND=nag_wp) *Input*
On entry: γ , the scaling factor used when estimating scores. If $\text{GAMMA} < 0$ then $\gamma = 1$ is used.
- 15: ACC(2) – REAL (KIND=nag_wp) array *Input*
On entry: a measure of the accuracy of the scoring function, f .
 Letting $a_i = \epsilon_1 |f(S_i)| + \epsilon_2$, then when confirming whether the scoring function is strictly increasing or decreasing (as described in MINCR), or when assessing whether a node defined by subset S_i can be trimmed, then any values in the range $f(S_i) \pm a_i$ are treated as being numerically equivalent.
 If $0 \leq \text{ACC}(1) \leq 1$ then $\epsilon_1 = \text{ACC}(1)$, otherwise $\epsilon_1 = 0$.
 If $\text{ACC}(2) \geq 0$ then $\epsilon_2 = \text{ACC}(2)$, otherwise $\epsilon_2 = 0$.
 In most situations setting both ϵ_1 and ϵ_2 to zero should be sufficient. Using a nonzero value, when one is not required, can significantly increase the number of subsets that need to be evaluated.
- 16: ICOMM(LICOMM) – INTEGER array *Communication Array*
On initial entry: ICOMM need not be set.
On intermediate exit: ICOMM is used for storage between calls to H05AAF.
On intermediate re-entry: ICOMM must remain unchanged.
On final exit: ICOMM is not defined. The first two elements, ICOMM(1) and ICOMM(2) contain the minimum required value for LICOMM and LRCOMM respectively.
- 17: LICOMM – INTEGER *Input*
On entry: the length of the array ICOMM. If LICOMM is too small and $\text{LICOMM} \geq 2$ then IFAIL = 172 is returned and the minimum value for LICOMM and LRCOMM are given by ICOMM(1) and ICOMM(2) respectively.
Constraints:
 if MINCNT = 0
 , $\text{LICOMM} \geq 2 \times \max(\text{NBEST}, M) + M(M + 2) + (M + 1) \times \max(M - \text{IP}, 1) + 27$;
 o t h e r w i s e
 $\text{LICOMM} \geq 2 \times \max(\text{NBEST}, M) + M(M + 3) + (2M + 1) \times \max(M - \text{IP}, 1) + 25$.
- 18: RCOMM(LRCOMM) – REAL (KIND=nag_wp) array *Communication Array*
On initial entry: RCOMM need not be set.
On intermediate exit: RCOMM is used for storage between calls to H05AAF.
On intermediate re-entry: RCOMM must remain unchanged.
On final exit: RCOMM is not defined.
- 19: LRCOMM – INTEGER *Input*
On entry: the length of the array RCOMM. If LRCOMM is too small and $\text{LICOMM} \geq 2$ then IFAIL = 172 is returned and the minimum value for LICOMM and LRCOMM are given by ICOMM(1) and ICOMM(2) respectively.
Constraints:
 if MINCNT = 0, $\text{LRCOMM} \geq 9 + \text{NBEST} + M \times \max(M - \text{IP}, 1)$;
 otherwise $\text{LRCOMM} \geq 8 + M + \text{NBEST} + M \times \max(M - \text{IP}, 1)$.

20: IFAIL – INTEGER

Input/Output

On initial entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if $\text{IFAIL} \neq 0$ on exit, the recommended value is -1 . **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On final exit: $\text{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\text{IFAIL} = 0$ or -1 , explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\text{IFAIL} = 11$

On entry, $\text{IREVCM} = \langle \text{value} \rangle$.
Constraint: $\text{IREVCM} = 0$ or 1.

$\text{IFAIL} = 21$

On entry, $\text{MINCR} = \langle \text{value} \rangle$.
Constraint: $\text{MINCR} = 0$ or 1.

$\text{IFAIL} = 22$

MINCR has changed between calls.
On intermediate entry, $\text{MINCR} = \langle \text{value} \rangle$.
On initial entry, $\text{MINCR} = \langle \text{value} \rangle$.

$\text{IFAIL} = 31$

On entry, $M = \langle \text{value} \rangle$.
Constraint: $M \geq 2$.

$\text{IFAIL} = 32$

M has changed between calls.
On intermediate entry, $M = \langle \text{value} \rangle$.
On initial entry, $M = \langle \text{value} \rangle$.

$\text{IFAIL} = 41$

On entry, $\text{IP} = \langle \text{value} \rangle$ and $M = \langle \text{value} \rangle$.
Constraint: $1 \leq \text{IP} \leq M$.

$\text{IFAIL} = 42$

IP has changed between calls.
On intermediate entry, $\text{IP} = \langle \text{value} \rangle$.
On initial entry, $\text{IP} = \langle \text{value} \rangle$.

$\text{IFAIL} = 51$

On entry, $\text{NBEST} = \langle \text{value} \rangle$.
Constraint: $\text{NBEST} \geq 1$.

IFAIL = 52

NBEST has changed between calls.
On intermediate entry, $NBEST = \langle value \rangle$.
On initial entry, $NBEST = \langle value \rangle$.

IFAIL = 53

On entry, $NBEST = \langle value \rangle$.
But only $\langle value \rangle$ best subsets could be calculated.

IFAIL = 61

DROP has changed between calls.
On intermediate entry, $DROP = \langle value \rangle$.
On initial entry, $DROP = \langle value \rangle$.

IFAIL = 71

LZ has changed between calls.
On entry, $LZ = \langle value \rangle$.
On previous exit, $LZ = \langle value \rangle$.

IFAIL = 91

LA has changed between calls.
On entry, $LA = \langle value \rangle$.
On previous exit, $LA = \langle value \rangle$.

IFAIL = 111

$BSCORE(\langle value \rangle) = \langle value \rangle$, which is inconsistent with the score for the parent node. Score for the parent node is $\langle value \rangle$.

IFAIL = 131

MINCNT has changed between calls.
On intermediate entry, $MINCNT = \langle value \rangle$.
On initial entry, $MINCNT = \langle value \rangle$.

IFAIL = 141

GAMMA has changed between calls.
On intermediate entry, $GAMMA = \langle value \rangle$.
On initial entry, $GAMMA = \langle value \rangle$.

IFAIL = 151

$ACC(1)$ has changed between calls.
On intermediate entry, $ACC(1) = \langle value \rangle$.
On initial entry, $ACC(1) = \langle value \rangle$.

IFAIL = 152

$ACC(2)$ has changed between calls.
On intermediate entry, $ACC(2) = \langle value \rangle$.
On initial entry, $ACC(2) = \langle value \rangle$.

IFAIL = 161

ICOMM has been corrupted between calls.

IFAIL = 171

On entry, LICOMM = $\langle value \rangle$, LRCOMM = $\langle value \rangle$.
 Constraint: LICOMM $\geq \langle value \rangle$, LRCOMM $\geq \langle value \rangle$.
 ICOMM is too small to return the required array sizes.

IFAIL = 172

On entry, LICOMM = $\langle value \rangle$, LRCOMM = $\langle value \rangle$.
 Constraint: LICOMM $\geq \langle value \rangle$, LRCOMM $\geq \langle value \rangle$.
 The minimum required values for LICOMM and LRCOMM are returned in ICOMM(1) and ICOMM(2) respectively.

IFAIL = 181

RCOMM has been corrupted between calls.

IFAIL = -99

An unexpected error has been triggered by this routine. Please contact NAG.
 See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL = -399

Your licence key may have expired or may not have been installed correctly.
 See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL = -999

Dynamic memory allocation failed.
 See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The subsets returned by H05AAF are guaranteed to be optimal up to the accuracy of your calculated scores.

8 Parallelism and Performance

H05AAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The maximum number of unique subsets of size p from a set of m features is $N = \frac{m!}{(m-p)!p!}$. The efficiency of the branch and bound algorithm implemented in H05AAF comes from evaluating subsets at internal nodes of the tree, that is subsets with more than p features, and where possible trimming branches of the tree based on the scores at these internal nodes as described in Narendra and Fukunaga (1977). Because of this it is possible, in some circumstances, for more than N subsets to be evaluated. This will tend to happen when most of the features have a similar effect on the subset score.

If multiple optimal subsets exist with the same score, and NBEST is too small to return them all, then the choice of which of these optimal subsets is returned is arbitrary.

10 Example

This example finds the three linear regression models, with five variables, that have the smallest residual sums of squares when fitted to a supplied dataset. The data used in this example was simulated.

10.1 Program Text

```
!   H05AAF Example Program Text

!   Mark 26 Release. NAG Copyright 2016.
!   Module h05aafe_mod

!   .. Use Statements ..
!   Use nag_library, Only: nag_wp
!   .. Implicit None Statement ..
!   Implicit None
!   .. Accessibility Statements ..
!   Private
!   Public                                :: calc_subset_score, free_subset_score
!   .. Parameters ..
!   Integer, Parameter, Public           :: nin = 5, nout = 6
!   .. Derived Type Definitions ..
!   Type, Public                         :: calc_subset_data
!   Integer                               :: n, ldq, ldx
!   Real (Kind=nag_wp)                   :: tol
!   Real (Kind=nag_wp), Allocatable      :: x(:, :), y(:, :), b(:, :), cov(:, :), h(:, :), &
!                                       p(:, :), q(:, :), res(:, :), se(:, :), wk(:, :), &
!                                       wt(:, :))
!   Integer, Allocatable                 :: isx(:)
!   Character (1)                        :: mean, weight
!   End Type calc_subset_data
!   Contains
!   Subroutine calc_subset_score(m, drop, lz, z, la, a, bscore, cs)
!   Calculate the score associated with a particular set of feature
!   subsets.

!   M, DROP, LZ, Z, LA, A and BSCORE are all as described in the documentation
!   of H05AAF.

!   CS - Variable of type CALC_SUBSET_DATA that holds any additional data
!   required by this routine

!   This particular example finds the set, of a given size, of explanatory
!   variables that best fit a response variable when a linear regression
!   model is used. Therefore the feature set is the set of all the
!   explanatory variables and the best set of features is defined as set
!   of explanatory variables that gives the smallest residual sums of
!   squares.
!   See the documentation for G02DAF for details on linear regression
!   models.

!   .. Use Statements ..
!   Use nag_library, Only: g02daf
!   .. Scalar Arguments ..
!   Type (calc_subset_data), Intent (Inout) :: cs
!   Integer, Intent (In)                   :: drop, la, lz, m
!   .. Array Arguments ..
!   Real (Kind=nag_wp), Intent (Out) :: bscore(:)
!   Integer, Intent (In)               :: a(:, :), z(:, :)
```

```

!      Allocate various arrays and read in the data if this is the first time
!      this routine has been called
!      If (.Not. allocated(cs%x)) Then
!          Read in the number of observations for the data used in the linear
!          regression skipping any headings or blank lines
!          Do
!              Read (nin,*,Iostat=ifail) line
!              If (ifail/=0) Then
!                  Exit
!              End If
!              Read (line,*,Iostat=ifail) cs%n
!              If (ifail==0) Then
!                  Exit
!              End If
!          End Do

!      Read in the control parameters for the linear regression
!      Read (nin,*) cs%tol

!      Read in the data
!      cs%ldx = cs%n
!      Allocate (cs%x(cs%ldx,m),cs%y(cs%n))
!      Read (nin,*)(cs%x(i,1:m),cs%y(i),i=1,cs%n)

!      No intercept term and no weights
!      cs%mean = 'Z'
!      cs%weight = 'U'

!      Allocate memory required by the regression routine
!      cs%ldq = cs%n
!      Allocate (cs%b(m),cs%cov((m*m+m)/2),cs%h(cs%n),cs%p(m*(m+
!          2)),cs%q(cs%ldq,m+1),cs%res(cs%n),cs%se(m),cs%wk(m*m+5*(m-
!          1)),cs%wt(1),cs%isx(m))
!      End If

!      Set up the initial feature set.
!      If DROP = 0, this is the Null set (i.e. no features).
!      If DROP = 1 then this is the full set (i.e. all features)
!      cs%isx(1:m) = drop

!      Add (if DROP = 0) or remove (if DROP = 1) the all the features
!      specified in Z
!      inv_drop = abs(drop-1)
!      Do i = 1, lz
!          cs%isx(z(i)) = inv_drop
!      End Do

!      Do i = 1, max(la,1)
!          If (la>0) Then
!              If (i>1) Then
!                  Reset the feature altered at the last iteration
!                  cs%isx(a(i-1)) = drop
!              End If

!              Add or drop the I'th feature in A
!              cs%isx(a(i)) = inv_drop
!          End If

!          ip = count(cs%isx(1:m)==1)

!      Fit the regression model
!      ifail = 0
!      Call g02daf(cs%mean,cs%weight,cs%n,cs%x,cs%ldx,m,cs%isx,ip,cs%y,
!          cs%wt,rss,idf,cs%b,cs%se,cs%cov,cs%res,cs%h,cs%q,cs%ldq,svd,irank,
!          cs%p,cs%tol,cs%wk,ifail)

!      Return the score (the residual sums of squares)
!      bscore(i) = rss
!      End Do
End Subroutine calc_subset_score
Subroutine free_subset_score(cs)

```

```

!      .. Scalar Arguments ..
      Type (calc_subset_data), Intent (Inout) :: cs
!      .. Intrinsic Procedures ..
      Intrinsic                                :: allocated
!      .. Executable Statements ..
      If (allocated(cs%x)) Then
        Deallocate (cs%x)
      End If
      If (allocated(cs%y)) Then
        Deallocate (cs%y)
      End If
      If (allocated(cs%b)) Then
        Deallocate (cs%b)
      End If
      If (allocated(cs%cov)) Then
        Deallocate (cs%cov)
      End If
      If (allocated(cs%h)) Then
        Deallocate (cs%h)
      End If
      If (allocated(cs%p)) Then
        Deallocate (cs%p)
      End If
      If (allocated(cs%q)) Then
        Deallocate (cs%q)
      End If
      If (allocated(cs%res)) Then
        Deallocate (cs%res)
      End If
      If (allocated(cs%se)) Then
        Deallocate (cs%se)
      End If
      If (allocated(cs%wk)) Then
        Deallocate (cs%wk)
      End If
      If (allocated(cs%wt)) Then
        Deallocate (cs%wt)
      End If
      If (allocated(cs%isx)) Then
        Deallocate (cs%isx)
      End If
      End Subroutine free_subset_score
End Module h05aafe_mod

Program h05aafe

!      .. Use Statements ..
      Use nag_library, Only: h05aaf, nag_wp
      Use h05aafe_mod, Only: calc_subset_data, calc_subset_score,      &
                           free_subset_score, nin, nout
!      .. Implicit None Statement ..
      Implicit None
!      .. Local Scalars ..
      Type (calc_subset_data)                :: cs
      Real (Kind=nag_wp)                     :: gamma
      Integer                                :: cnt, drop, i, ifail, ip, irevcm, j, &
                           la, licomm, lrcomm, lz, m, mincnt,      &
                           mincr, mip, nbest
!      .. Local Arrays ..
      Real (Kind=nag_wp)                     :: acc(2)
      Real (Kind=nag_wp), Allocatable        :: bscore(:), rcomm(:)
      Integer, Allocatable                   :: a(:), bz(:, :), ibz(:), icomm(:),      &
                           z(:)
      Logical, Allocatable                   :: mask(:)
!      .. Intrinsic Procedures ..
      Intrinsic                                :: max, pack
!      .. Executable Statements ..
      Write (nout,*) 'H05AAF Example Program Results'
      Write (nout,*)

```

```

!      Skip headings in data file
      Read (nin,*)
      Read (nin,*)

!      Read in the problem size
      Read (nin,*) m, ip, nbest

!      Read in the control parameters for the subset selection
      Read (nin,*) mincr, mincnt, gamma, acc(1:2)

!      Allocate memory required by the subset selection routine
      mip = m - ip
      Allocate (z(mip),a(max(nbest,m)),bz(mip,nbest),bscore(max(nbest,m)))

!      Allocate dummy communication arrays, as we will query the required size
      licomm = 2
      lrcomm = 0
      Allocate (icomm(licomm),rcomm(lrcomm))

!      Query the required length for the communication arrays
      irevcm = 0
      ifail = 1
      Call h05aaf(irevcm,mincr,m,ip,nbest,drop,lz,z,la,a,bscore,bz,mincnt,      &
        gamma,acc,icomm,licomm,rcomm,lrcomm,ifail)

!      Extract the required sizes from the communication array
      licomm = icomm(1)
      lrcomm = icomm(2)

!      Reallocate communication arrays to the correct size
      Deallocate (icomm,rcomm)
      Allocate (icomm(licomm),rcomm(lrcomm))

!      Initialize reverse communication control flag
      irevcm = 0

!      Call the reverse communication best subset routine in a loop.
!      The loop should terminate when IRECVCM = 0 on exit
      cnt = 0
rev_lp: Do
      ifail = -1
      Call h05aaf(irevcm,mincr,m,ip,nbest,drop,lz,z,la,a,bscore,bz,mincnt,      &
        gamma,acc,icomm,licomm,rcomm,lrcomm,ifail)
      If (irevcm==0) Then
        If (ifail==0 .Or. ifail==53) Then
!          No error, or a warning was issued
          Exit rev_lp
        Else
!          An error occurred
          Go To 100
        End If
      End If

!      Calculate and return the score for the required models and keep track
!      of the number of subsets evaluated
      cnt = cnt + max(1,la)
      Call calc_subset_score(m,drop,lz,z,la,a,bscore,cs)

End Do rev_lp

      Call free_subset_score(cs)

!      Titles
      Write (nout,99999) '      Score      Feature Subset'
      Write (nout,99999) '      -----      -'

!      Display the best subsets and corresponding scores. H05AAF returns a list
!      of features excluded from the best subsets, so this is inverted to give
!      the set of features included in each subset
      Allocate (ibz(m),mask(m))
      ibz(1:m) = (/ (i,i=1,m) /)

```

```

Do i = 1, la
  mask(1:m) = .True.
  Do j = 1, mip
    mask(bz(j,i)) = .False.
  End Do
  Write (nout,99998) bscore(i), pack(ibz,mask)
End Do

Write (nout,*)
If (ifail==53) Then
  Write (nout,99997) nbest,
    ' subsets of the requested size do not exist, only ', la,
    ' are displayed.'
End If
Write (nout,99996) cnt, ' subsets evaluated in total'

100 Continue

99999 Format (1X,A)
99998 Format (1X,E12.5,100(1X,I5))
99997 Format (1X,I0,A,I0,A)
99996 Format (1X,I0,A)

End Program h05aafe

```

10.2 Program Data

H05AAF Example Program Data

Data required by H05AAF

```

14 5 3 :: M,IP,NBEST
0 -1 -1.0 -1.0 -1.0 :: MINCR, MINCNT, GAMMA, ACC(1:2)

```

Data required by the scoring function

```

40 :: N
1e-6 :: TOL

```

-1.59	0.19	0.40	0.43	-0.40	0.79	0.06	
0.33	1.60	0.58	-1.12	1.23	1.07	-0.07	-2.44
-0.25	0.61	-0.36	1.16	0.61	-2.05	-0.02	
-0.04	0.80	-0.73	-0.63	-0.75	-0.73	1.43	-2.97
-2.28	0.46	-0.65	0.33	0.16	-0.21	-1.61	
-0.54	0.48	0.37	-0.95	-2.14	0.48	2.02	7.42
-0.52	1.05	0.64	0.02	-1.12	0.23	0.06	
-1.26	1.40	-0.98	2.47	0.49	-0.02	-0.05	3.00
-0.84	1.86	0.10	0.73	-1.41	0.98	0.20	
-0.89	1.84	2.56	0.60	-0.12	0.71	0.23	8.83
1.12	-0.51	-0.58	0.09	-1.14	2.11	-0.11	
-0.34	-1.04	-0.43	-0.01	-0.38	1.80	0.05	0.03
0.06	0.85	-2.09	0.22	-1.35	-0.36	1.20	
0.41	0.80	-0.28	0.18	0.27	0.92	0.63	2.57
-0.48	-1.02	0.08	-0.06	0.13	-1.18	2.30	
0.03	0.45	0.62	-1.97	0.97	0.93	-0.18	8.31
0.08	-0.31	0.43	-0.38	0.01	1.30	0.66	
0.65	-0.59	0.76	0.04	0.17	-0.76	-0.90	4.55
0.66	1.14	0.40	2.37	1.10	0.17	-0.38	
1.15	-1.00	-0.13	-0.69	-0.62	-0.18	0.00	-23.10
-1.08	-0.21	-1.13	-0.79	-0.76	-1.58	0.38	
-0.03	1.26	-0.51	-0.75	0.86	0.29	0.68	3.38
-0.74	-1.59	-0.58	-1.09	1.18	-1.70	-1.02	
0.36	1.05	1.30	-0.98	-1.36	-1.28	-1.32	-0.13
0.40	-1.58	-1.30	-0.10	-1.34	0.65	-0.56	
0.39	-0.73	-0.32	2.19	-0.49	0.69	0.18	5.47
0.75	-3.09	-0.61	-1.89	0.15	0.77	-0.49	
-0.63	1.20	-0.04	1.02	0.31	0.81	-0.45	13.97
-0.65	1.57	-1.50	-1.45	0.21	0.06	0.24	
2.24	-1.34	0.30	1.39	-0.38	-0.71	0.48	20.94
1.36	1.40	-1.40	-0.90	0.36	-0.21	-0.97	
0.36	-0.26	0.08	0.06	-1.49	0.43	-1.61	-12.87
-1.01	1.50	-0.61	-0.25	-1.01	-0.43	1.90	
-1.33	-0.96	-0.02	0.51	-1.38	-0.78	1.82	28.26
1.34	1.02	3.50	0.10	0.50	0.04	0.61	

-0.57	-2.69	-0.64	-0.34	-0.21	-1.97	-0.19	6.89
0.29	0.67	-0.38	-0.63	-0.24	1.21	-0.09	
0.90	-2.20	1.72	0.29	0.66	0.19	-0.57	5.37
0.67	-0.56	-0.41	1.22	-0.30	0.77	0.82	
0.36	1.18	1.87	-1.48	0.52	1.35	0.13	-1.50
-0.40	-1.10	-0.83	0.71	1.99	-0.24	1.30	
-0.34	-0.70	0.28	0.16	0.27	0.37	-1.79	-23.00
-0.78	0.60	-0.45	-0.26	-0.23	0.89	0.87	
1.01	1.20	0.28	0.79	2.76	0.35	1.31	14.09
-1.29	0.62	-0.59	1.52	0.62	0.21	1.31	
1.09	-0.36	-0.34	-0.03	-0.59	-1.70	-0.03	-11.05
0.40	-1.45	-0.98	2.10	-1.09	-0.53	-0.38	
-1.36	0.13	0.70	-1.51	0.08	-0.62	-0.64	-32.04
0.43	-0.86	0.70	-1.07	-0.76	0.72	-0.14	
-1.58	0.00	0.58	-0.21	1.30	2.02	1.52	23.36
-0.48	0.01	1.30	0.58	-0.54	1.09	0.91	
2.90	1.32	-1.20	-0.59	-0.51	0.20	-1.74	-5.58
-1.32	-1.41	-0.58	-1.29	1.61	-0.35	-0.72	
-1.92	-1.09	0.56	-0.87	-0.71	1.25	0.10	2.48
1.43	0.69	1.34	-0.32	2.84	-1.43	-0.47	
-0.01	0.83	-0.72	-0.78	0.50	-1.22	0.54	-5.30
0.82	0.46	0.15	-0.57	0.93	1.33	-0.23	
-1.07	0.76	0.25	-1.96	0.39	0.24	-0.26	-7.77
-0.91	0.23	-0.19	1.58	-0.27	0.33	-0.60	
-1.39	-0.30	-0.81	-0.95	0.88	-0.09	-0.35	-34.25
0.65	-1.14	1.18	-1.06	-0.68	-0.22	0.21	
0.94	1.08	0.81	-0.33	0.42	-0.90	0.49	26.78
-0.36	-0.50	-0.02	-0.04	0.77	0.62	-1.35	
-0.64	1.20	1.22	0.18	-1.39	-0.81	-0.99	-11.85
-1.82	1.06	0.28	0.14	0.62	-0.80	-1.08	
-2.15	1.37	1.57	-1.48	-0.79	0.28	-0.20	-8.62
1.54	0.50	0.13	-0.68	0.26	-1.13	0.62	
-0.43	0.39	1.14	0.15	1.03	0.46	0.40	12.35
-1.61	-0.61	0.93	-0.37	0.44	-1.45	0.58	
-1.77	0.72	-2.05	-0.03	-1.24	-1.40	-0.06	-1.54
-0.48	0.67	0.04	0.27	-0.84	-0.06	-3.67	
0.09	1.66	-0.30	1.67	1.08	0.00	0.43	-16.59
-1.65	-1.16	-1.17	1.12	0.11	-0.15	0.48	
-1.72	1.08	-0.94	0.49	-0.56	0.95	1.09	-8.69
-0.85	-0.02	1.18	-1.16	0.49	1.56	-0.60	
0.32	0.72	-1.20	2.52	1.78	0.16	-0.01	7.82
-0.60	-0.73	-1.23	1.50	0.40	-0.20	-0.65	
0.68	1.09	0.40	-1.50	-2.10	0.21	-0.18	-18.56
-0.66	-0.01	-0.01	0.85	-2.04	1.17	-0.56	
1.72	-0.18	1.14	-0.96	-0.92	-0.28	1.58	17.21 :: X,Y

10.3 Program Results

H05AAF Example Program Results

Score	Feature Subset				
-----	-----				
0.10475E+04	4	7	8	10	14
0.10599E+04	4	5	7	8	14
0.10702E+04	4	5	7	10	14

45 subsets evaluated in total
