NAG Library Function Document

nag_step_regsn (g02eec)

1 Purpose

nag_step_regsn (g02eec) carries out one step of a forward selection procedure in order to enable the 'best' linear regression model to be found.

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_step_regsn (Nag_OrderType order, Integer *istep,
     Nag_IncludeMean mean, Integer n, Integer m, const double x[],
     Integer pdx, const char *var_names[], const Integer sx[], Integer maxip,
     const double y[], const double wt[], double fin, Nag_Boolean *addvar,
     const char *newvar[], double *chrss, double *f, const char *model[],
     Integer *nterm, double *rss, Integer *idf, Integer *ifr,
     const char *free_vars[], double exss[], double q[], Integer pdq,
     double p[], NagError *fail)
```

3 Description

One method of selecting a linear regression model from a given set of independent variables is by forward selection. The following procedure is used:

(i) Select the best fitting independent variable, i.e., the independent variable which gives the smallest residual sum of squares. If the F-test for this variable is greater than a chosen critical value, F_c, then include the variable in the model, else stop.

(ii) Find the independent variable that leads to the greatest reduction in the residual sum of squares when added to the current model.

(iii) If the F-test for this variable is greater than a chosen critical value, F_c, then include the variable in the model and go to (ii), otherwise stop.

At any step the variables not in the model are known as the free terms.

nag_step_regsn (g02eec) allows you to specify some independent variables that must be in the model, these are known as forced variables.

The computational procedure involves the use of QR decompositions, the R and the Q matrices being updated as each new variable is added to the model. In addition the matrix $Q^{\mathrm{T}}X_{\mathrm{free}}$, where X_{free} is the matrix of variables not included in the model, is updated.

nag_step_regsn (g02eec) computes one step of the forward selection procedure at a call. The results produced at each step may be printed or used as inputs to nag_regsn_mult_linear_upd_model (g02ddc), in order to compute the regression coefficients for the model fitted at that step. Repeated calls to nag_step_regsn (g02eec) should be made until $F < F_c$ is indicated.

4 References

Draper N R and Smith H (1985) *Applied Regression Analysis* (2nd Edition) Wiley

Weisberg S (1985) *Applied Linear Regression* Wiley

5 Arguments

Note: after the initial call to nag_step_regsn (g02eec) with **istep** $= 0$ all arguments except **fin** must not be changed by you between calls.

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **istep** – Integer * *Input/Output*

On entry: indicates which step in the forward selection process is to be carried out.

istep $= 0$
The process is initialized.

Constraint: **istep** ≥ 0.

On exit: is incremented by 1.

3: **mean** – Nag_IncludeMean *Input*

On entry: indicates if a mean term is to be included.

mean = Nag_MeanInclude
A mean term, intercept, will be included in the model.

mean = Nag_MeanZero
The model will pass through the origin, zero-point.

Constraint: **mean** = Nag_MeanInclude or Nag_MeanZero.

4: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: **n** ≥ 2.

5: **m** – Integer *Input*

On entry: m, the total number of independent variables in the dataset.

Constraint: **m** ≥ 1.

6: **x**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **x** must be at least

$\max(1, \mathbf{pdx} \times \mathbf{m})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdx})$ when **order** = Nag_RowMajor.

Where $\mathbf{X}(i, j)$ appears in this document, it refers to the array element

$\mathbf{x}[(j-1) \times \mathbf{pdx} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1]$ when **order** = Nag_RowMajor.

On entry: $\mathbf{X}(i, j)$ must contain the ith observation for the jth independent variable, for $i = 1, 2, \ldots, \mathbf{n}$ and $j = 1, 2, \ldots, \mathbf{m}$.

7: **pdx** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **x**.

Constraints:

if **order** = Nag_ColMajor, **pdx** $\geq$ **n**;
if **order** = Nag_RowMajor, **pdx** $\geq$ **m**.

8: **var_names**[**m**] – const char * *Input*

On entry: **var_names**$[i-1]$ must contain the name of the independent variable in row i of **x**, for $i = 1, 2, \ldots, \mathbf{m}$.

9: **sx**[**m**] – const Integer *Input*

On entry: indicates which independent variables could be considered for inclusion in the regression.

sx$[j-1] \geq 2$
The variable contained in the jth column of **x** is automatically included in the regression model, for $j = 1, 2, \ldots, \mathbf{m}$.

sx$[j-1] = 1$
The variable contained in the jth column of **x** is considered for inclusion in the regression model, for $j = 1, 2, \ldots, \mathbf{m}$.

sx$[j-1] = 0$
The variable in the jth column is not considered for inclusion in the model, for $j = 1, 2, \ldots, \mathbf{m}$.

Constraint: **sx**$[j-1] \geq 0$ and at least one value of **sx**$[j-1] = 1$, for $j = 1, 2, \ldots, \mathbf{m}$.

10: **maxip** – Integer *Input*

On entry: the maximum number of independent variables to be included in the model.

Constraints:

if **mean** = Nag_MeanInclude, **maxip** $\geq 1 +$ number of values of **sx** > 0;
if **mean** = Nag_MeanZero, **maxip** $\geq$ number of values of **sx** > 0.

11: **y**[**n**] – const double *Input*

On entry: the dependent variable.

12: **wt**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **wt** must be at least **n**.

On entry: W, **wt** must contain the weights to be used in the weighted regression.

If **wt**$[i-1] = 0.0$, then the ith observation is not included in the model, in which case the effective number of observations is the number of observations with nonzero weights.

If weights are not provided then **wt** must be set to the null pointer, i.e., `(double *)0`, and the effective number of observations is **n**.

Constraint: if **wt** is not **NULL**, **wt**$[i] \geq 0.0$, for $i = 0, 1, \ldots, \mathbf{n}-1$.

13: **fin** – double *Input*

On entry: the critical value of the F statistic for the term to be included in the model, F_c.

Suggested value: 2.0 is a commonly used value in exploratory modelling.

Constraint: **fin** ≥ 0.0.

14: **addvar** – Nag_Boolean * *Output*

On exit: indicates if a variable has been added to the model.

addvar = Nag_TRUE
A variable has been added to the model.

addvar = Nag_FALSE
No variable had an F value greater than F_c and none were added to the model.

15: **newvar**[**1**] – const char * *Output*

On exit: if **addvar** = Nag_TRUE, **newvar** contains the name of the variable added to the model.

16: **chrss** – double * *Output*

On exit: if **addvar** = Nag_TRUE, **chrss** contains the change in the residual sum of squares due to adding variable **newvar**.

17: **f** – double * *Output*

On exit: if **addvar** = Nag_TRUE, **f** contains the F statistic for the inclusion of the variable in **newvar**.

18: **model**[**maxip**] – const char * *Input/Output*

On entry: if **istep** = 0, **model** need not be set.

If **istep** $\neq 0$, **model** must contain the values returned by the previous call to nag_step_regsn (g02eec).

On exit: the names of the variables in the current model.

19: **nterm** – Integer * *Input/Output*

On entry: if **istep** = 0, **nterm** need not be set.

If **istep** $\neq 0$, **nterm** must contain the value returned by the previous call to nag_step_regsn (g02eec).

On exit: the number of independent variables in the current model, not including the mean, if any.

20: **rss** – double * *Input/Output*

On entry: if **istep** = 0, **rss** need not be set.

If **istep** $\neq 0$, **rss** must contain the value returned by the previous call to nag_step_regsn (g02eec).

On exit: the residual sums of squares for the current model.

21: **idf** – Integer * *Input/Output*

On entry: if **istep** = 0, **idf** need not be set.

If **istep** $\neq 0$, **idf** must contain the value returned by the previous call to nag_step_regsn (g02eec).

On exit: the degrees of freedom for the residual sum of squares for the current model.

22: **ifr** – Integer * *Input/Output*

On entry: if **istep** = 0, **ifr** need not be set.

If **istep** $\neq 0$, **ifr** must contain the value returned by the previous call to nag_step_regsn (g02eec).

On exit: the number of free independent variables, i.e., the number of variables not in the model that are still being considered for selection.

23: **free_vars**[**maxip**] – const char * *Input/Output*

On entry: if **istep** $= 0$, **free_vars** need not be set.

If **istep** $\neq 0$, **free_vars** must contain the values returned by the previous call to nag_step_regsn (g02eec).

On exit: the first **ifr** values of **free_vars** contain the names of the free variables.

24: **exss**[**maxip**] – double *Output*

On exit: the first **ifr** values of **exss** contain what would be the change in regression sum of squares if the free variables had been added to the model, i.e., the extra sum of squares for the free variables. **exss**$[i-1]$ contains what would be the change in regression sum of squares if the variable **free_vars**$[i-1]$ had been added to the model.

25: **q**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **q** must be at least

$\max(1, \mathbf{pdq} \times \mathbf{maxip} + 2)$ when **order** $=$ Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdq})$ when **order** $=$ Nag_RowMajor.

The (i,j)th element of the matrix Q is stored in

q$[(j-1) \times \mathbf{pdq} + i - 1]$ when **order** $=$ Nag_ColMajor;
q$[(i-1) \times \mathbf{pdq} + j - 1]$ when **order** $=$ Nag_RowMajor.

On entry: if **istep** $= 0$, **q** need not be set.

If **istep** $\neq 0$, **q** must contain the values returned by the previous call to nag_step_regsn (g02eec).

On exit: the results of the QR decomposition for the current model:

the first column of **q** contains $c = Q^{\mathrm{T}}y$ (or $Q^{\mathrm{T}}W^{\frac{1}{2}}y$ where W is the vector of weights if used);

the upper triangular part of columns 2 to $p+1$ contain the R matrix;

the strictly lower triangular part of columns 2 to $p+1$ contain details of the Q matrix;

the remaining $p+1$ to $p+$ **ifr** columns of contain $Q^{\mathrm{T}}X_{free}$ (or $Q^{\mathrm{T}}W^{\frac{1}{2}}X_{free}$),

where $p =$ **nterm**, or $p =$ **nterm** $+ 1$ if **mean** $=$ Nag_MeanInclude.

26: **pdq** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **q**.

Constraints:

if **order** $=$ Nag_ColMajor, **pdq** $\geq$ **n**;
if **order** $=$ Nag_RowMajor, **pdq** $\geq$ **maxip** $+ 2$.

27: **p**[**maxip** $+$ **1**] – double *Input/Output*

On entry: if **istep** $= 0$, **p** need not be set.

If **istep** $\neq 0$, **p** must contain the values returned by the previous call to nag_step_regsn (g02eec).

On exit: the first p elements of **p** contain details of the QR decomposition, where $p =$ **nterm**, or $p =$ **nterm** $+ 1$ if **mean** $=$ Nag_MeanInclude.

28: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_DENOM_ZERO

Denominator of **f** statistic is ≤ 0.0.

NE_FREE_VARS

There are no free variables in the regression.

NE_FULL_RANK

Forced variables not of full rank.

NE_INT

On entry, **istep** = ⟨*value*⟩.
Constraint: **istep** ≥ 0.

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 1.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 2.

On entry, **pdq** = ⟨*value*⟩.
Constraint: **pdq** > 0.

On entry, **pdx** = ⟨*value*⟩.
Constraint: **pdx** > 0.

NE_INT_2

On entry, **istep** and **nterm** are inconsistent: **istep** = ⟨*value*⟩ and **nterm** = ⟨*value*⟩.

On entry, **pdq** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdq** $\geq$ **n**.

On entry, **pdx** = ⟨*value*⟩ and **m** = ⟨*value*⟩.
Constraint: **pdx** $\geq$ **m**.

On entry, **pdx** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdx** $\geq$ **n**.

NE_INT_ARRAY

On entry, **maxip** is too small for number of terms given by **sx**: **maxip** = ⟨*value*⟩.

NE_INT_ARRAY_ELEM_CONS

On entry, **sx**[⟨*value*⟩] < 0.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_REAL

On entry, **fin** $= \langle value \rangle$.
Constraint: **fin** ≥ 0.0.

On entry, with nonzero **istep**, **rss** ≤ 0.0: **rss** $= \langle value \rangle$.

NE_REAL_ARRAY_ELEM_CONS

On entry, **wt**$[\langle value \rangle] < 0.0$.

NE_ZERO_DF

Degrees of freedom for error will equal 0 if new variable is added.

On entry, number of forced variables $\geq$ **n**, i.e., **idf** would be zero.

NE_ZERO_VARS

Maximum number of variables to be included is 0.

7 Accuracy

As nag_step_regsn (g02eec) uses a QR transformation the results will often be more accurate than traditional algorithms using methods based on the cross-products of the dependent and independent variables.

8 Parallelism and Performance

nag_step_regsn (g02eec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_step_regsn (g02eec) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

9 Further Comments

None.

10 Example

The data, from an oxygen uptake experiment, is given by Weisberg (1985). The names of the variables are as given in Weisberg (1985). The independent and dependent variables are read and nag_step_regsn (g02eec) is repeatedly called until **addvar** = Nag_FALSE. At each step the F statistic, the free variables and their extra sum of squares are printed; also, except for when **addvar** = Nag_FALSE, the new variable, the change in the residual sum of squares and the terms in the model are printed.

10.1 Program Text

```
/* nag_step_regsn (g02eec) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>

int main(void)
{
  /* Scalars */
  double          chrss, f, fin, rss;
  Integer         exit_status, i, idf, ifr, istep, j, m, maxip, n, nterm, pdq,
                  pdx;
  /* Arrays */
  char            nag_enum_arg[40];
  char            *newvar = 0;
  double          *exss = 0, *p = 0, *q = 0, *wt = 0, *x = 0, *y = 0;
  double          *wtptr = 0;
  Integer         *sx = 0;
  char            **free_vars = 0, **model = 0;
  const char      *vname[] = { "DAY", "BOD", "TKN", "TS", "TVS", "COD" };
  /* NAG Types */
  Nag_OrderType   order;
  Nag_IncludeMean mean;
  Nag_Boolean     addvar = Nag_FALSE, weight;
  NagError        fail;

#ifdef NAG_COLUMN_MAJOR
#define X(I, J) x[(J-1)*pdx + I - 1]
  order = Nag_ColMajor;
#else
#define X(I, J) x[(I-1)*pdx + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_step_regsn (g02eec) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"", &n, &m);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"", &n, &m);
#endif
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  mean = (Nag_IncludeMean) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, _countof(nag_enum_arg));
```

```
#else
  scanf(" %39s", nag_enum_arg);
#endif
  weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
  maxip = m;

  /* Allocate memory */
  if (!(exss = NAG_ALLOC(maxip, double)) ||
      !(p = NAG_ALLOC(maxip+1, double)) ||
      !(q = NAG_ALLOC(n * (maxip+2), double)) ||
      !(wt = NAG_ALLOC(n, double)) ||
      !(x = NAG_ALLOC(n * m, double)) ||
      !(y = NAG_ALLOC(n, double)) ||
      !(sx = NAG_ALLOC(m, Integer)) ||
      !(free_vars = NAG_ALLOC(maxip, char *)) ||
      !(model = NAG_ALLOC(maxip, char *))
      )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#ifdef NAG_COLUMN_MAJOR
  pdx = n;
  pdq = n;
#else
  pdx = m;
  pdq = maxip+2;
#endif

  if (weight)
    {
      for (i = 1; i <= n; ++i)
        {
#ifdef _WIN32
          for (j = 1; j <= m; ++j) scanf_s("%lf", &X(i, j));
#else
          for (j = 1; j <= m; ++j) scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
          scanf_s("%lf%lf%*[^\n]", &y[i - 1], &wt[i - 1]);
#else
          scanf("%lf%lf%*[^\n]", &y[i - 1], &wt[i - 1]);
#endif
          wtptr = wt;
        }
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
#ifdef _WIN32
          for (j = 1; j <= m; ++j) scanf_s("%lf", &X(i, j));
#else
          for (j = 1; j <= m; ++j) scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
          scanf_s("%lf%*[^\n] ", &y[i - 1]);
#else
          scanf("%lf%*[^\n] ", &y[i - 1]);
#endif
        }
    }

#ifdef _WIN32
  for (j = 0; j < m; ++j) scanf_s("%"NAG_IFMT"", &sx[j]);
#else
  for (j = 0; j < m; ++j) scanf("%"NAG_IFMT"", &sx[j]);
#endif
#ifdef _WIN32
```

```
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%lf%*[^\n]", &fin);
#else
  scanf("%lf%*[^\n]", &fin);
#endif
  printf("\n");

  istep = 0;
  for (i = 1; i <= m; ++i)
    {
      /* nag_step_regsn (g02eec).
       * Fits a linear regression model by forward selection
       */
      nag_step_regsn(order, &istep, mean, n, m, x, pdx, vname, sx, maxip, y,
                     wtptr, fin, &addvar, (const char **)&newvar, &chrss, &f,
                     (const char **)model, &nterm, &rss, &idf, &ifr,
                     (const char **)free_vars, exss, q, pdq, p, &fail);

      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_step_regsn (g02eec).\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      printf("Step %"NAG_IFMT"\n", istep);
      if (!addvar)
        {
          printf("No further variables added maximum F =%7.2f\n", f);
          printf("Free variables:        ");

          for (j = 1; j <= ifr; ++j)
            printf("%3.3s %s", free_vars[j-1], j%6 == 0 || j == ifr?"\n":" ");

          printf("\nChange in residual sums of squares for free variables:\n");

          printf("                  ");
          for (j = 1; j <= ifr; ++j)
            {
              printf("%9.4f", exss[j - 1]);
              printf("%s", j%6 == 0 || j == ifr?"\n":" ");
            }
          goto END;
        }
      else
        {
          printf("Added variable is %3s\n", newvar);
          printf("Change in residual sum of squares =%13.4e\n", chrss);
          printf("F Statistic = %7.2f\n\n", f);
          printf("Variables in model: ");

          for (j = 1; j <= nterm; ++j)
            printf("%3s %s", model[j-1], j%6 == 0 || j == nterm?"\n":" ");
          printf("Residual sum of squares = %13.4e\n", rss);
          printf("Degrees of freedom = %"NAG_IFMT"\n\n", idf);
          if (ifr == 0)
            {
              printf("No free variables remaining\n");
              goto END;
            }

          printf("%s%6s", "Free variables:  ", "");
          for (j = 1; j <= ifr; ++j)
            {
              printf("%3.3s ", free_vars[j-1]);
              printf(j%6 == 0 || j == ifr?"\n":" ");
            }
```

```
          printf("Change in residual sums of squares for free variables:\n");
          printf("                 ");

          for (j = 1; j <= ifr; ++j)
            printf("%9.4f%s", exss[j - 1], j%6 == 0 || j == ifr?"\n":" ");
          printf("\n");
        }
    }

 END:
  NAG_FREE(model);
  NAG_FREE(free_vars);
  NAG_FREE(exss);
  NAG_FREE(p);
  NAG_FREE(q);
  NAG_FREE(wt);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(sx);

  return exit_status;
}
```

10.2 Program Data

```
nag_step_regsn (g02eec) Example Program Data
  20 6 Nag_MeanInclude Nag_FALSE
   0. 1125.0 232.0 7160.0 85.9 8905.0  1.5563
   7.  920.0 268.0 8804.0 86.5 7388.0  0.8976
  15.  835.0 271.0 8108.0 85.2 5348.0  0.7482
  22. 1000.0 237.0 6370.0 83.8 8056.0  0.7160
  29. 1150.0 192.0 6441.0 82.1 6960.0  0.3010
  37.  990.0 202.0 5154.0 79.2 5690.0  0.3617
  44.  840.0 184.0 5896.0 81.2 6932.0  0.1139
  58.  650.0 200.0 5336.0 80.6 5400.0  0.1139
  65.  640.0 180.0 5041.0 78.4 3177.0 -0.2218
  72.  583.0 165.0 5012.0 79.3 4461.0 -0.1549
  80.  570.0 151.0 4825.0 78.7 3901.0  0.0000
  86.  570.0 171.0 4391.0 78.0 5002.0  0.0000
  93.  510.0 243.0 4320.0 72.3 4665.0 -0.0969
 100.  555.0 147.0 3709.0 74.9 4642.0 -0.2218
 107.  460.0 286.0 3969.0 74.4 4840.0 -0.3979
 122.  275.0 198.0 3558.0 72.5 4479.0 -0.1549
 129.  510.0 196.0 4361.0 57.7 4200.0 -0.2218
 151.  165.0 210.0 3301.0 71.8 3410.0 -0.3979
 171.  244.0 327.0 2964.0 72.5 3360.0 -0.5229
 220.   79.0 334.0 2777.0 71.9 2599.0 -0.0458
  0     1    1      1     1    2
 2.0
```

10.3 Program Results

```
nag_step_regsn (g02eec) Example Program Results

Step 1
Added variable is  TS
Change in residual sum of squares =   4.7126e-01
F Statistic =    7.38

Variables in model: COD   TS
Residual sum of squares =    1.0850e+00
Degrees of freedom = 17

Free variables:        TKN  BOD  TVS
Change in residual sums of squares for free variables:
                   0.1175    0.0600    0.2276

Step 2
No further variables added maximum F =   1.59
Free variables:        TKN  BOD  TVS

Change in residual sums of squares for free variables:
                   0.0979    0.0207    0.0217
```