NAG Library Routine Document

g22ycf

Note: please be advised that this routine is classed as 'experimental' and its interface may be developed further in the future. Please see Section 3.1.1 in How to Use the NAG Library and its Documentation for further information.

1 Purpose

g22ycf generates a design matrix from a data matrix and model description.

2 Specification

```
Subroutine g22ycf (hform, hddesc, dat, lddat, sddat, hxdesc, x, ldx,       &
                   sdx, mx, ifail)
Integer, Intent (In)                :: lddat, sddat, ldx, sdx
Integer, Intent (Inout)             :: ifail
Integer, Intent (Out)               :: mx
Real (Kind=nag_wp), Intent (In)     :: dat(lddat,sddat)
Real (Kind=nag_wp), Intent (Inout)  :: x(ldx,sdx)
Type (c_ptr), Intent (In)           :: hform, hddesc
Type (c_ptr), Intent (Inout)        :: hxdesc
```

3 Description

g22ycf generates a design matrix from a data matrix and a model description. Design matrices encapsulate the observed values of the independent variables and the required model in a form that can be used by many of the model fitting routines available in the NAG Library, for example those in Chapter G02.

3.1 Notation

Let D denote a data matrix with n observations on m_d independent variables, denoted by V_j, for $j = 1, 2, \dots, m_d$. If V_j is a categorical variable, let L_j denote the number of levels associated with it. If V_j is a binary, ordinal or continuous variable, let $L_j = 1$.

Let V_{ji} denote the ith value of V_j.

Let $\mathcal{M}$ denote a model made up of one or more terms, denoted by T_i. Each term consists of either a main effect or an interaction and hence can be described using one or more variable names V_j and the interaction operator '.'. The operator '+' is used to denote the addition of a term to the model. Therefore, $\mathcal{M} = T_1 + T_2 + T_3 = V_1 + V_2 + V_1.V_2$ denotes a model with three terms, the first two terms being the main effects for variables V_1 and V_2 and the last term the interaction between them. For simplicity we reorder the terms of the model by the number of variables in them, so main effects come first, then two-way interactions, then three-way interactions etc. By default it is assumed that the model $\mathcal{M}$ contains a mean effect (or intercept term), if the mean effect is excluded, this will be denoted by '-1', so $\mathcal{M} = T_1$ is a model with one term and a mean effect and $\mathcal{M} = T_1 - 1$ is the same model with the mean effect dropped.

g22ycf generates an n by m_x design matrix, X, from D and $\mathcal{M}$.

3.2 Dummy Variables

When constructing a design matrix, we cannot work directly with categorical variables. Categorical variables must first be recoded into dummy variables. A categorical variable V_j requires L_j dummy variables. Let $\mathcal{D}^j$ denote an $n \times L_j$ matrix of dummy variables for V_j defined as

$$\mathcal{D}^j_{li} = \begin{cases} 1; & \text{if } V_{ji} = l, \\ 0; & \text{otherwise} \end{cases}$$

where $\mathcal{D}^j_l$ is the lth column of $\mathcal{D}^j$ and $\mathcal{D}^j_{li}$ is the ith element of $\mathcal{D}^j_l$.

For a binary, ordinal or continuous variable, $\mathcal{D}^j_{1i} = V_{ji}$.

3.3 Full Design Matrix

Given a model, $\mathcal{M}$, and the matrices of dummy variables constructing the full design matrix X_F is trivial. Each term is processed in order and

1. If term i is a main effect, that is $T_i = V_j$ for some j, $\mathcal{D}^j$ is copied into X_F.
2. If term i is a two-way interaction, that is $T_i = V_j.V_k$, for some $j \neq k$, then
   (i) Loop over $l_j = 1, 2, \ldots L_j$.
   (ii) Loop over $l_k = 1, 2, \ldots L_k$.
   (iii) Add a column to X_F corresponding to the element-wise product of $\mathcal{D}^j_{l_j}$ and $\mathcal{D}^k_{l_k}$.
3. Higher interaction terms are handled in a similar manner as the two-way interactions by adding columns constructed from multiplying all combinations of the columns of the corresponding $\mathcal{D}$s that correspond to the variables involved. In all cases, the variables towards the right hand side of a term are iterated over the quickest.

3.4 Contrasts

Using the full design matrix X_F in an analysis can result in an overparameterized model. This is due to X_F often not being of full rank as the sum of all the dummy variables for a particular variable is a vector of ones. This source of overparameterization can be alleviated by using a design matrix X where (some) dummy variables are replaced by contrasts. For a categorical variable V_j the contrasts are a set of $L_j - 1$ functionally independent linear combinations of the dummy variables.

Whilst the choice of contrasts used in term T_i will affect the individual model coefficients (parameters), it has no effect on the overall contribution of T_i.

For a given variable V_j, the contrasts can be represented by an L_j by $L_j - 1$ matrix, C_j. The rows of C_j correspond to a particular value of V_j and the columns correspond to the values to use in the design matrix.

Six types of contrast are available in **g22ycf**; two types of treatment contrasts, two types of sum contrasts, Helmert contrasts and polynomial contrasts. Unless specified otherwise, the contrasts used by **g22ycf** are treatment contrasts relative to the first level. See the description of the optional parameter **Contrast** in **g22yaf** for ways of changing the contrasts used.

3.4.1 Treatment Contrasts

Treatment contrasts are taken relative to either the first or last level of the variable. For example, if $L_j = 4$,

$$C_j = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

would be the contrast matrix for V_j using treatment contrasts relative to the first level. The contrast matrix obtained when using treatment contrasts relative to the last level is similar, but the row of zeros appears at the bottom and all other rows are shifted up one.

Strictly speaking, the term *contrast* implies that each row in the contrast matrix sums to zero. That is not the case for treatment contrasts, however they are included as this coding is commonly used in practice.

3.4.2 Sum Contrasts

Sum contrasts are similar to treatment contrasts and again can be taken relative to the first or last level of the variable. Unlike treatment contrasts, sum contrasts effectively constrain the coefficients related to the variable to sum to zero. For example, if $L_j = 4$,

$$C_j = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{pmatrix}$$

would be the contrast matrix for V_j using treatment contrasts relative to the last level. The contrast matrix obtained when using treatment contrasts relative to the first level is similar, but the row of -1s appears at the top and all other rows are shifted down one.

3.4.3 Helmert Contrasts

With Helmert contrasts level l of the variable is compared with the average effect of all previous levels. For example, if $L_j = 4$,

$$C_j = \begin{pmatrix} -1 & -1 & -1 \\ 1 & -1 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & 3 \end{pmatrix}$$

would be the contrast matrix for V_j using Helmert contrasts.

3.4.4 Polynomial Contrasts

With polynomial contrasts the entries in the columns of C_j correspond in linear, quadratic, cubic, quartic, etc. terms to a hypothetical underlying numeric variable that takes equally spaced values at each level. For example, if $L_j = 4$,

$$C_j = \begin{pmatrix} -0.67 & 0.50 & -0.22 \\ -0.22 & -0.50 & 0.67 \\ 0.22 & -0.50 & -0.67 \\ 0.67 & 0.50 & 0.22 \end{pmatrix}$$

would be the contrast matrix for V_j using polynomial contrasts.

3.4.5 When Contrasts Can Be Used

Depending on the specifics of the model, $\mathcal{M}$, it may not be possible to always replace the L_j dummy variables with $L_j - 1$ contrasts for all variables in all terms and retain the same model. A simple example of this is a data matrix, D, with four observations and two variables which have two and three levels respectively. This data matrix might look something like:

$$D = \begin{pmatrix} 1 & 1 \\ 2 & 3 \\ 1 & 2 \\ 2 & 2 \end{pmatrix}$$

For the sake of argument, assume that our model contains the main effect for each variable, but does not contain a mean effect (or intercept term). So using the notation established earlier, $\mathcal{M} = V_1 + V_2 - 1$. The full design matrix, X_F, for this data matrix and model would be

$$X_F = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

However, X_F is not of full rank (and hence $\mathcal{M}$ is overparameterized) because the sum of the first two columns is a vector of ones as is the sum of the last three columns.

In order to alleviate this we might try constructing X_C where the dummy variables have been replaced by contrasts. Assuming treatment contrasts, relative to the first level, we would have

$$X_C = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

However, using X_C makes an implicit assumption that the expected value of the dependent variable (the quantity being modelled) is zero when $V_1 = 1$ and $V_2 = 1$. This assumption was not made when we used X_F and hence the two design matrices are not equivalent. One solution would be to use dummy variables for V_1 and contrasts for V_2, which would result in a design matrix, X of

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Using X would give an equivalent model to using X_F.

The algorithm used by **g22ycf** to decide which variables, in which terms, can be coded as contrasts and which need to be coded as dummy variables is described below.

Suppose V_j is any variable that appears in term T_i, let $T_{i(j)}$ denote the term obtained by dropping V_j from T_i. For example, if $T_3 = V_1.V_2.V_3$, $T_{3(2)} = V_1.V_3$. In this context, the empty term is taken to be the mean effect (or intercept term). We say that $T_{i(j)}$ appears in $\mathcal{M}$ if there exists a term T_k, $k < i$, that contains all of the variables appearing in $T_{i(j)}$. In most cases $T_k = T_{i(j)}$, but this is not required. Note, as stated earlier, the terms in $\mathcal{M}$ are ordered by the number of variables in them.

A variable, V_j in term T_i is coded by contrasts if $T_{i(j)}$ appears in $\mathcal{M}$ and by dummy variables otherwise. It is therefore possible for variable V_j to be coded by contrasts in some terms and dummy variables in others within the same X.

The above rule assumes the presence of a mean effect. If no such effect is present in the model, the main effect of the first categorical variable is coded by dummy variables to compensate. If no main effects appear in the model, the warning **ifail** $= 14$ is returned.

A longer description and informal proof that the resulting X is a suitable design matrix for the model of interest can be found in chapter two of Chambers and Hastie (1992).

3.5 Mean Effect

The mean effect (or intercept term) is included in a design matrix by adding a column of ones as the first column of X. However, many model fitting routines in the NAG Library handle the mean effect as a special case and do not require it to be explicitly added to the design matrix. Therefore, by default, **g22ycf** does not explicitly add the mean effect to the design matrix. This behaviour can be changed via the optional parameter **Explicit Mean** in **g22yaf**.

4 References

Chambers J M and Hastie T J (1992) *Statistical Models in S* Wadsworth and Brooks/Cole Computer Science Series

5 Arguments

1: **hform** – Type (c_ptr) *Input*

On entry: a G22 handle to the internal data structure containing a description of the model $\mathcal{M}$ as returned in **hform** by **g22yaf**.

2: **hddesc** – Type (c_ptr) *Input*

On entry: a G22 handle to the internal data structure containing a description of the data matrix, D as returned in **hddesc** by **g22ybf**.

3: **dat**(**lddat**, **sddat**) – Real (Kind=nag_wp) array *Input*

On entry: the data matrix, D. By default D_{ij}, the ith value for the jth variable, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m_d$, should be supplied in **dat**(i, j).

If the optional parameter **Storage Order**, described in **g22ybf**, is set to VAROBS, D_{ij} should be supplied in **dat**(j, i).

4: **lddat** – Integer *Input*

On entry: the first dimension of the array **dat** as declared in the (sub)program from which **g22ycf** is called.

Constraints:

if the optional parameter **Storage Order**, described in **g22ybf**, is set to VAROBS, **lddat** $\geq m_d$;
otherwise **lddat** $\geq n$.

5: **sddat** – Integer *Input*

On entry: the second dimension of the array **dat** as declared in the (sub)program from which **g22ycf** is called.

Constraints:

if the optional parameter **Storage Order**, described in **g22ybf**, is set to VAROBS, **sddat** $\geq n$;
otherwise **sddat** $\geq m_d$.

6: **hxdesc** – Type (c_ptr) *Input/Output*

On entry: must be set to **c_null_ptr**.

As an alternative an existing G22 handle may be supplied in which case this routine will destroy the supplied G22 handle as if **g22zaf** had been called.

On exit: holds a G22 handle to the internal data structure containing a description of the design matrix, X. You **must not** change the G22 handle other than through the routines in Chapter G22.

7: **x**(**ldx**, **sdx**) – Real (Kind=nag_wp) array *Output*

On exit: the design matrix, X. By default X_{ij}, the ith value for the jth column, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m_x$, is returned in **x**(i, j)

If the optional parameter **Storage Order**, described in **g22yaf**, is set to VAROBS, X_{ij} is returned in **x**(j, i).

If **ldx** or **sdx** are too small to hold **x**, the number of columns required to hold the design matrix is returned in **mx**.

Under some conditions it is possible to use the data matrix in place of the design matrix. Specifically, if D has no categorical variables, $\mathcal{M}$ has only main effects and either has no mean effect or the mean effect does not need to be explicitly added to the design matrix. If **ldx** or **sdx** are too small under such circumstances, **ifail** $= 71$ is returned and **hxdesc** is set up in such a way as to allow **dat** to be used as the design matrix.

8: **ldx** – Integer *Input*

On entry: the first dimension of the array **x** as declared in the (sub)program from which **g22ycf** is called.

Constraints:

if the optional parameter **Storage Order**, described in **g22yaf**, is set to VAROBS, **ldx** $\geq m_x$;
otherwise **ldx** $\geq n$.

9: **sdx** – Integer *Input*

On entry: the second dimension of the array **x** as declared in the (sub)program from which **g22ycf** is called.

Constraints:

if the optional parameter **Storage Order**, described in **g22yaf**, is set to VAROBS, **sdx** $\geq n$;
otherwise **sdx** $\geq m_x$.

10: **mx** – Integer *Output*

On exit: the minimum number of columns required to hold the design matrix.

In most cases **mx** $= m_x$. The one exception is when **ifail** $= 71$, that is the size of **x** was too small but the data matrix given in **dat** can be used as the design matrix. In this case **mx** holds the number of columns that would be required if only the relevant parts of **dat** were copied into a new array.

11: **ifail** – Integer *Input/Output*

On 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, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of ifail on exit.**

On exit: **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 **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:

ifail $= 11$

hform has not been initialized or is corrupt.

ifail $= 12$

hform is not a G22 handle as generated by **g22yaf**.

ifail $= 13$

A variable name used when creating **hform** is not present in **hddesc**.
Variable name: $\langle$*value*$\rangle$.

ifail $= 14$

The model contains categorical variables, but no intercept or main effects terms have been requested.
Please check the design matrix returned matches the model you require.

ifail $= 21$

hddesc has not been initialized or is corrupt.

ifail $= 22$

hddesc is not a G22 handle as generated by **g22ybf**.

ifail $= 31$

On entry, column j of the data matrix, D, is not consistent with information supplied in **hddesc**, $j = \langle$*value*$\rangle$.

ifail $= 41$

On entry, $n = \langle$*value*$\rangle$ and **lddat** $= \langle$*value*$\rangle$.
Constraint: **lddat** $\geq n$.

ifail $= 42$

On entry, $m_d = \langle$*value*$\rangle$ and **lddat** $= \langle$*value*$\rangle$.
Constraint: **lddat** $\geq m_d$.

ifail $= 51$

On entry, $m_d = \langle$*value*$\rangle$ and **sddat** $= \langle$*value*$\rangle$.
Constraint: **sddat** $\geq m_d$.

ifail $= 52$

On entry, $n = \langle$*value*$\rangle$ and **sddat** $= \langle$*value*$\rangle$.
Constraint: **sddat** $\geq n$.

ifail $= 61$

On entry, **hxdesc** is not **c_null_ptr** or a recognised G22 handle.

ifail $= 71$

On entry, the size of **x** is too small to hold the design matrix. **dat** can be used instead.

ifail $= 81$

On entry, $n = \langle$*value*$\rangle$ and **ldx** $= \langle$*value*$\rangle$.
Constraint: **ldx** $\geq n$.

ifail $= 82$

On entry, $m_x = \langle$*value*$\rangle$ and **ldx** $= \langle$*value*$\rangle$.
Constraint: **ldx** $\geq m_x$.

ifail $= 91$

On entry, $m_x = \langle$*value*$\rangle$ and **sdx** $= \langle$*value*$\rangle$.
Constraint: **sdx** $\geq m_x$.

ifail $= 92$

On entry, $n = \langle value \rangle$ and **sdx** $= \langle value \rangle$.
Constraint: **sdx** $\geq n$.

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

Not applicable.

8 Parallelism and Performance

g22ycf is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

g22ycf 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 routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

g22ydf can be used to obtain labels for the columns of the design matrix X.

Many of the analysis routines that require a design matrix to be supplied allow submodels to be defined through the use of a vector of ones or zeros indicating whether a column of X should be included or excluded from the analyses (see for example **isx** in **g02daf** or **g02gaf**). This allows nested models to be fit without having to reconstruct the design matrix for each analysis. **g22ydf** offers a mechanism for constructing these vectors using submodels specified using **g22yaf**.

10 Example

This example creates and outputs two design matrices for a simple linear regression model. The first design matrix uses sum contrasts for all variables and the second uses a combination of polynomial and Helmert contrasts. Column labels are generated using **g22ydf**.

See also the examples for **g22yaf**, **g22ybf** and **g22ydf**.

10.1 Program Text

```
!   G22YCF Example Program Text
!   Mark 26.1 Release. NAG Copyright 2017.

    Module g22ycfe_mod
!     G22YCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: print_x, read_line
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6

    Contains
      Subroutine read_line(ierr,v1,v2)
!       Read in a line from NIN, remove any comments and optionally
!       split out the first word

!       .. Scalar Arguments ..
        Integer, Intent (Out)          :: ierr
        Character (*), Intent (Out)    :: v1
        Character (*), Intent (Out), Optional :: v2
!       .. Local Scalars ..
        Integer                        :: pend
!       .. Intrinsic Procedures ..
        Intrinsic                      :: adjustl, index, present
!       .. Executable Statements ..
        Continue

        Read (nin,'(A200)',Iostat=ierr) v1
        If (ierr==0) Then
          pend = index(v1,'::')
          If (pend/=0) Then
            v1 = v1(1:pend-1)
          End If
          v1 = adjustl(v1)

          If (present(v2)) Then
!           split the first word from the line
            pend = index(v1,' ')
            If (pend/=0) Then
              v2 = adjustl(v1(pend:))
              v1 = adjustl(v1(1:pend))
            Else
              v2 = ''
            End If
          End If
        End If

        Return
      End Subroutine read_line
      Subroutine print_x(intcpt,plab,nobs,mx,x,text)
!       Print the transpose of the first 10 rows of the design matrix

!       .. Parameters ..
        Integer, Parameter             :: max_rows = 10
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: mx, nobs
        Character (*), Intent (In)     :: intcpt, text
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x(:,:)
        Character (*), Intent (In)     :: plab(:)
!       .. Local Scalars ..
        Integer                        :: i, pnobs, si
!       .. Intrinsic Procedures ..
```

```
        Intrinsic                      :: min, repeat, trim
!       .. Executable Statements ..
        Continue

!       PLAB holds the labels for the model parameters, so includes a label
!       for the mean effect, if one is present. As the mean effect is not
!       being explicitly included in the design matrix, we may need to skip
!       the first element of PLAB (which will always be the label for the
!       mean effect if one is present)
        If (intcpt=='M') Then
          si = 1
        Else
          si = 0
        End If

!       Printing the first MAX_ROWS rows of the design matrix
        pnobs = min(max_rows,nobs)

!       Display the design matrix
        Write (nout,99998) 'Transpose of First ', pnobs, ' Rows of the ',      &
          text, ' Design Matrix (X)'
        Write (nout,99997) 'Column Name', (i,i=1,pnobs)
        Write (nout,99996) repeat('-',15+pnobs*5)
        Do i = 1, mx
          Write (nout,99999) plab(i+si), x(1:pnobs,i)
        End Do
        Write (nout,*) 'Intercept flag = ', trim(intcpt)

        Return
99999   Format (1X,A15,100(1X,F4.1))
99998   Format (1X,A,I0,A,A,A)
99997   Format (1X,A,3X,100(3X,I2))
99996   Format (1X,A)
      End Subroutine print_x
    End Module g22ycfe_mod

    Program g22ycfe

!     .. Use Statements ..
      Use g22ycfe_mod, Only: nin, nout, print_x, read_line
      Use, Intrinsic                   :: iso_c_binding, Only: c_null_ptr,     &
                                          c_ptr
      Use nag_library, Only: g22yaf, g22ybf, g22ycf, g22ydf, g22zaf, g22zmf,   &
                             nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Type (c_ptr)                     :: hddesc, hform, hxdesc
      Integer                          :: i, ierr, ifail, ip, lddat, ldx,      &
                                          lisx, lplab, lvinfo, lvnames, mx,    &
                                          nobs, nvar, sddat, sdx
      Character (200)                  :: formula, intcpt, line, tcontrast,    &
                                          tvname
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: dat(:,:), x(:,:)
      Real (Kind=nag_wp)               :: tx(0,0)
      Integer, Allocatable             :: isx(:), levels(:), vinfo(:)
      Character (50), Allocatable      :: plab(:), vnames(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: trim
!     .. Executable Statements ..
      Write (nout,*) 'G22YCF Example Program Results'
      Write (nout,*)

      hform = c_null_ptr
      hddesc = c_null_ptr
      hxdesc = c_null_ptr

!     Skip heading in data file
      Read (nin,*)
```

```
!     Read in the formula for the first specification of the model,
!     remove comments and parse it
      Call read_line(ierr,formula)
      ifail = 0
      Call g22yaf(hform,formula,ifail)

!     Read in the contrast to use for all parameters, remove comments and
!     set the contrast optional parameter
      Call read_line(ierr,tcontrast)
      line = 'Contrast=' // trim(tcontrast)
      ifail = 0
      Call g22zmf(hform,line,ifail)

!     Read in size of the data matrix and number of variable labels supplied
      Read (nin,*) nobs, nvar, lvnames

!     Read in number of levels and names for the variables
      Allocate (levels(nvar),vnames(lvnames))
      Read (nin,*) levels(1:nvar)
      If (lvnames>0) Then
        Read (nin,*) vnames(1:lvnames)
      End If

!     Create a description of the data matrix
      ifail = 0
      Call g22ybf(hddesc,nobs,nvar,levels,lvnames,vnames,ifail)

!     Read in the data matrix and response variable
      lddat = nobs
      sddat = nvar
      Allocate (dat(lddat,sddat))
      Read (nin,*)(dat(i,1:nvar),i=1,nobs)

!     Calculate the size of the design matrix
      ldx = 0
      sdx = 0
      ifail = 1
      Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,tx,ldx,sdx,mx,ifail)
      If (ifail/=91) Then
!       redisplay any error messages, other than IFAIL = 91
        ifail = 0
        Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,tx,ldx,sdx,mx,ifail)
      End If

!     Generate the design matrix, X
      ldx = nobs
      sdx = mx
      Allocate (x(ldx,sdx))
      ifail = 0
      Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,x,ldx,sdx,mx,ifail)

!     Generate labels for the columns of X
      lplab = mx + 1
      lvinfo = 0
      lisx = 0
      Allocate (isx(lisx),vinfo(lvinfo),plab(lplab))
      ifail = 0
      Call g22ydf(hform,hxdesc,intcpt,ip,lisx,isx,lplab,plab,lvinfo,vinfo,     &
        ifail)

!     Display the design matrix
      Call print_x(intcpt,plab,nobs,mx,x,'First')

c_lp: Do
!       Read in the name of the variable whose contrasts need to be changed,
!       the value to change them to, remove comments and set the contrast
!       optional argument for the specified variable
        Call read_line(ierr,tvname,tcontrast)
        If (ierr/=0) Then
          Exit c_lp
        End If
```

```
          line = 'Contrast:' // trim(tvname) // '=' // trim(tcontrast)
          ifail = 0
          Call g22zmf(hform,line,ifail)
        End Do c_lp

!     Regenerate the design matrix using the new contrasts
!     (the size of X should be the same as previously)
      ifail = 0
      Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,x,ldx,sdx,mx,ifail)

!     Generate labels for the columns of X
      ifail = 0
      Call g22ydf(hform,hxdesc,intcpt,ip,lisx,isx,lplab,plab,lvinfo,vinfo,     &
        ifail)

!     Display the design matrix
      Write (nout,*)
      Call print_x(intcpt,plab,nobs,mx,x,'Second')

!     Clean-up the G22 handles
      ifail = 0
      Call g22zaf(hform,ifail)
      Call g22zaf(hddesc,ifail)
      Call g22zaf(hxdesc,ifail)

      Deallocate (dat,x)
      Deallocate (isx,levels,vinfo)
      Deallocate (plab,vnames)

    End Program g22ycfe
```

10.2 Program Data

```
G22YCF Example Program Data
F1*F2*Con - F1.F2.Con                   :: FORMULA
Sum First                               :: contrast to use
25 3 3                                  :: NOBS,NVAR,LVNAMES
3 3 1                                   :: LEVELS
F1 F2 Con                               :: VNAMES
3 1 -2.4
3 3  0.2
1 3 -1.4
2 1 -5.4
3 3  0.2
3 2  1.4
1 2  6.8
1 2  6.7
1 1  5.3
2 3 -1.3
3 2 -3.6
3 2 -0.7
1 1  5.7
3 3  2.3
1 2  3.3
2 3 -0.5
1 1 -2.6
1 2  3.7
1 2  0.9
3 1 -1.1
2 2  2.1
1 3  4.6
2 3  4.6
1 2  5.1
1 3  0.9                                :: DAT
F1 Helmert
F2 Polynomial                           :: new contrasts to use
```

10.3 Program Results

```
G22YCF Example Program Results

Transpose of First 10 Rows of the First Design Matrix (X)
Column Name        1    2    3    4    5    6    7    8    9   10
-----------------------------------------------------------------
F1_SF1           0.0  0.0 -1.0  1.0  0.0  0.0 -1.0 -1.0 -1.0  1.0
F1_SF2           1.0  1.0 -1.0  0.0  1.0  1.0 -1.0 -1.0 -1.0  0.0
F2_SF1          -1.0  0.0  0.0 -1.0  0.0  1.0  1.0  1.0 -1.0  0.0
F2_SF2          -1.0  1.0  1.0 -1.0  1.0  0.0  0.0  0.0 -1.0  1.0
CON             -2.4  0.2 -1.4 -5.4  0.2  1.4  6.8  6.7  5.3 -1.3
F1_SF1.F2_SF1   -0.0  0.0 -0.0 -1.0  0.0  0.0 -1.0 -1.0  1.0  0.0
F1_SF1.F2_SF2   -0.0  0.0 -1.0 -1.0  0.0  0.0 -0.0 -0.0  1.0  1.0
F1_SF2.F2_SF1   -1.0  0.0 -0.0 -0.0  0.0  1.0 -1.0 -1.0  1.0  0.0
F1_SF2.F2_SF2   -1.0  1.0 -1.0 -0.0  1.0  0.0 -0.0 -0.0  1.0  0.0
F1_SF1.CON      -0.0  0.0  1.4 -5.4  0.0  0.0 -6.8 -6.7 -5.3 -1.3
F1_SF2.CON      -2.4  0.2  1.4 -0.0  0.2  1.4 -6.8 -6.7 -5.3 -0.0
F2_SF1.CON       2.4  0.0 -0.0  5.4  0.0  1.4  6.8  6.7 -5.3 -0.0
F2_SF2.CON       2.4  0.2 -1.4  5.4  0.2  0.0  0.0  0.0 -5.3 -1.3
Intercept flag = M

Transpose of First 10 Rows of the Second Design Matrix (X)
Column Name        1    2    3    4    5    6    7    8    9   10
-----------------------------------------------------------------
F1_H1            0.0  0.0 -1.0  1.0  0.0  0.0 -1.0 -1.0 -1.0  1.0
F1_H2            2.0  2.0 -1.0 -1.0  2.0  2.0 -1.0 -1.0 -1.0 -1.0
F2_P1           -0.7  0.7  0.7 -0.7  0.7  0.0  0.0  0.0 -0.7  0.7
F2_P2            0.4  0.4  0.4  0.4  0.4 -0.8 -0.8 -0.8  0.4  0.4
CON             -2.4  0.2 -1.4 -5.4  0.2  1.4  6.8  6.7  5.3 -1.3
F1_H1.F2_P1     -0.0  0.0 -0.7 -0.7  0.0  0.0 -0.0 -0.0  0.7  0.7
F1_H1.F2_P2      0.0  0.0 -0.4  0.4  0.0 -0.0  0.8  0.8 -0.4  0.4
F1_H2.F2_P1     -1.4  1.4 -0.7  0.7  1.4  0.0 -0.0 -0.0  0.7 -0.7
F1_H2.F2_P2      0.8  0.8 -0.4 -0.4  0.8 -1.6  0.8  0.8 -0.4 -0.4
F1_H1.CON       -0.0  0.0  1.4 -5.4  0.0  0.0 -6.8 -6.7 -5.3 -1.3
F1_H2.CON       -4.8  0.4  1.4  5.4  0.4  2.8 -6.8 -6.7 -5.3  1.3
F2_P1.CON        1.7  0.1 -1.0  3.8  0.1  0.0  0.0  0.0 -3.7 -0.9
F2_P2.CON       -1.0  0.1 -0.6 -2.2  0.1 -1.1 -5.6 -5.5  2.2 -0.5
Intercept flag = M
```

11 Optional Parameters

As well as the optional parameters common to all G22 handles described in **g22zmf** and **g22znf**, a number of additional optional parameters can be specified for a G22 handle holding the description of a design matrix as returned by **g22ycf** in **hxdesc**.

The value of an optional parameter can be queried using **g22znf**.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 11.1.

Formula
Min Number of Columns
Number of Columns
Number of Observations
Storage Order

11.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

Keywords and character values are case and white space insensitive.

Formula a Read Only

This optional parameter returns a verbose formula string describing the model, $\mathcal{M}$, used to create the design matrix. This formula will only contain variable names, the operators '$+$' and '.' and any contrast identifiers present.

Min Number of Columns i Read Only

This optional parameter returns the minimum number of columns required to hold the design matrix, X. In most cases **Min Number of Columns** $=$ **Number of Columns**. The one exception is when **ifail** $= 71$, that is the size of **x** was too small but the data matrix given in **dat** can be used as the design matrix. In this case, **Number of Columns** $= m_x = m_d$ and **Min Number of Columns** holds the number of columns that would be required if only the relevant parts of **dat** were copied into a new array.

Number of Columns i Read Only

This optional parameter returns m_x, the number of columns in the design matrix.

Number of Observations i Read Only

This optional parameter returns n, the number of observations in the design matrix.

Storage Order a Read Only

This optional parameter returns how the design matrix, X, is stored in **x**.

If **Storage Order** $=$ OBSVAR, X_{ij}, the value for the jth variable of the ith observation of the design matrix is stored in $\mathbf{x}(i,j)$.

If **Storage Order** $=$ VAROBS, X_{ij}, the value for the jth variable of the ith observation of the design matrix is stored in $\mathbf{x}(j,i)$.

It should be noted that **Storage Order** is not writeable. If you wish to change the storage order of the design matrix you need to change **Storage Order** in **hform** as described in Section 11 in **g22yaf** prior to calling **g22ycf**.