

## NAG Library Function Document

### nag\_glm\_tran\_model (g02gkc)

#### 1 Purpose

nag\_glm\_tran\_model (g02gkc) calculates the estimates of the arguments of a generalized linear model for given constraints from the singular value decomposition results.

#### 2 Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_glm_tran_model (Integer ip, Integer nclin, const double v[],
    Integer tdv, const double c[], Integer tdc, double b[], double scale,
    double se[], double cov[], NagError *fail)
```

#### 3 Description

nag\_glm\_tran\_model (g02gkc) computes the estimates given a set of linear constraints for a generalized linear model which is not of full rank. It is intended for use after a call to nag\_glm\_normal (g02gac), nag\_glm\_binomial (g02gbc), nag\_glm\_poisson (g02gcc) or nag\_glm\_gamma (g02gdc).

In the case of a model not of full rank the functions use a singular value decomposition (SVD) to find the parameter estimates,  $\hat{\beta}_{svd}$ , and their variance-covariance matrix. Details of the SVD are made available, in the form of the matrix  $P^*$ :

$$P^* = \begin{pmatrix} D^{-1}P_1^T \\ P_0^T \end{pmatrix}$$

as described by nag\_glm\_normal (g02gac), nag\_glm\_binomial (g02gbc), nag\_glm\_poisson (g02gcc) and nag\_glm\_gamma (g02gdc).

Alternative solutions can be formed by imposing constraints on the arguments. If there are  $p$  arguments and the rank of the model is  $k$ , then  $n_c = p - k$  constraints will have to be imposed to obtain a unique solution.

Let  $C$  be a  $p$  by  $n_c$  matrix of constraints, such that

$$C^T\beta = 0,$$

then the new parameter estimates  $\hat{\beta}_c$  are given by:

$$\begin{aligned} \hat{\beta}_c &= A\hat{\beta}_{svd} \\ &= \left( I - P_0(C^T P_0)^{-1} \right) \hat{\beta}_{svd}, \end{aligned}$$

where  $I$  is the identity matrix, and the variance-covariance matrix is given by:

$$AP_1 D^{-2} P_1^T A^T$$

provided  $(C^T P_0)^{-1}$  exists.

#### 4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

McCullagh P and Nelder J A (1983) *Generalized Linear Models* Chapman and Hall

Searle S R (1971) *Linear Models* Wiley

## 5 Arguments

- 1: **ip** – Integer *Input*  
*On entry:* the number of terms in the linear model,  $p$ .  
*Constraint:*  $\mathbf{ip} \geq 1$ .
- 2: **nclin** – Integer *Input*  
*On entry:* the number of constraints to be imposed on the arguments,  $n_c$ .  
*Constraint:*  $0 < \mathbf{nclin} < \mathbf{ip}$ .
- 3: **v[ip × tdv]** – const double *Input*  
**Note:** the  $(i, j)$ th element of the matrix  $V$  is stored in  $\mathbf{v}[(i - 1) \times \mathbf{tdv} + j - 1]$ .  
*On entry:*  $\mathbf{v}$  as returned by `nag_glm_normal (g02gac)`, `nag_glm_binomial (g02gbc)`, `nag_glm_poisson (g02gcc)` or `nag_glm_gamma (g02gdc)`.
- 4: **tdv** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array  $\mathbf{v}$ .  
*Constraint:*  $\mathbf{tdv} \geq \mathbf{ip} + 6$ .  
**tdv** should be as supplied to `nag_glm_normal (g02gac)`, `nag_glm_binomial (g02gbc)`, `nag_glm_poisson (g02gcc)` or `nag_glm_gamma (g02gdc)`.
- 5: **c[ip × tdc]** – const double *Input*  
**Note:** the  $(i, j)$ th element of the matrix  $C$  is stored in  $\mathbf{c}[(i - 1) \times \mathbf{tdc} + j - 1]$ .  
*On entry:* the  $\mathbf{nclin}$  constraints stored by column, i.e., the  $i$ th constraint is stored in the  $i$ th column of  $\mathbf{c}$ .
- 6: **tdc** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array  $\mathbf{c}$ .  
*Constraint:*  $\mathbf{tdc} \geq \mathbf{nclin}$ .
- 7: **b[ip]** – double *Input/Output*  
*On entry:* the parameter estimates computed by using the singular value decomposition,  $\hat{\beta}_{svd}$ .  
*On exit:* the parameter estimates of the arguments with the constraints imposed,  $\hat{\beta}_c$ .
- 8: **scale** – double *Input*  
*On entry:* the estimate of the scale argument.  
For results from `nag_glm_normal (g02gac)` and `nag_glm_gamma (g02gdc)` then **scale** is the scale argument, for the model  $\sigma^2$  and  $\hat{\nu}^{-1}$  respectively.  
For results from `nag_glm_binomial (g02gbc)` and `nag_glm_poisson (g02gcc)` then **scale** should be set to 1.0.  
*Constraint:*  $\mathbf{scale} > 0.0$ .
- 9: **se[ip]** – double *Output*  
*On exit:* the standard error of the parameter estimates in  $\mathbf{b}$ .

- 10: **cov**[(**ip**) × (**ip** + 1)/2] – double *Output*  
*On exit:* the upper triangular part of the variance-covariance matrix of the **ip** parameter estimates given in **b**. They are stored packed by column, i.e., the covariance between the parameter estimate given in **b**[*i*] and the parameter estimate given in **b**[*j*],  $j \geq i$ , is stored in **cov**[ $j(j+1)/2 + i$ ], for  $i = 0, 1, \dots, \mathbf{ip} - 1$  and  $j = i, \dots, \mathbf{ip} - 1$ .
- 11: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_2\_INT\_ARG\_GE

On entry, **nclin** = *<value>* while **ip** = *<value>*. These arguments must satisfy **nclin** < **ip**.

### NE\_2\_INT\_ARG\_LT

On entry, **tdc** = *<value>* while **nclin** = *<value>*. These arguments must satisfy **tdc** ≥ **nclin**.

On entry, **tdv** = *<value>* while **ip** = *<value>*. These arguments must satisfy **tdv** ≥ **ip** + 6.

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_INT\_ARG\_LE

On entry, **nclin** = *<value>*.

Constraint: **nclin** > 0.

### NE\_INT\_ARG\_LT

On entry, **ip** = *<value>*.

Constraint: **ip** ≥ 1.

### NE\_MAT\_NOT\_FULL\_RANK

Matrix **c** does not give a model of full rank.

### NE\_REAL\_ARG\_LE

On entry, **scale** must not be less than or equal to 0.0: **scale** = *<value>*.

## 7 Accuracy

It should be noted that due to rounding errors an argument that should be zero when the constraints have been imposed may be returned as a value of order *machine precision*.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

`nag_glm_tran_model` (g02gkc) is intended for use in situations in which dummy (0-1) variables have been used such as in the analysis of designed experiments when you do not wish to change the arguments of the model to give a full rank model. The function is not intended for situations in which the relationships between the independent variables are only approximate.

## 10 Example

A loglinear model is fitted to a 3 by 5 contingency table by `nag_glm_poisson` (`g02gcc`). The model consists of terms for rows and columns. The table is:

141	67	114	79	39
131	66	143	72	35
36	14	38	28	16

The constraints that the sum of row effects and the sum of column effects are zero are then read in and the parameter estimates with these constraints imposed are computed by `nag_glm_tran_model` (`g02gkc`) and printed.

### 10.1 Program Text

```

/* nag_glm_tran_model (g02gkc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 */

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

#define X(I, J) x[(I) *tdx + J]
#define C(I, J) c[(I) *tdc + J]
#define V(I, J) v[(I) *tdv + J]
int main(void)
{
  Integer  exit_status = 0, i, ip, j, m, max_iter, n, nclin, print_iter, rank;
  Integer  *sx = 0, tdc, tdv, tdx;
  NagError fail;
  double   dev, df, eps, ex_power;
  double   *b = 0, *c = 0, *cov = 0, *se = 0, tol, *v = 0, *wtptr;
  double   *x = 0, *y = 0;

  INIT_FAIL(fail);

  printf("nag_glm_tran_model (g02gkc) Example Program Results\n\n");
  /* Skip heading in data file */
  scanf("%*[\n]");
  scanf("%ld %ld %ld", &n, &m, &print_iter);

  if (n >= 2 && m >= 1)
  {
    if (!(x = NAG_ALLOC(n*m, double)) ||
        !(y = NAG_ALLOC(n, double)) ||
        !(sx = NAG_ALLOC(m, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdx = m;
  }
  else
  {
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
  wtptr = (double *) 0;
  for (i = 0; i < n; i++)
  {

```

```

        for (j = 0; j < m; j++)
            scanf("%lf", &X(i, j));
        scanf("%lf", &y[i]);
    }
for (j = 0; j < m; j++)
    scanf("%ld", &sx[j]);
scanf("%ld", &ip);
if (!(b = NAG_ALLOC(ip, double)) ||
    !(cov = NAG_ALLOC(ip*(ip+1)/2, double)) ||
    !(se = NAG_ALLOC(ip, double)) ||
    !(v = NAG_ALLOC(n*(ip+6), double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
tdv = ip+6;
/* Set control parameters */
max_iter = 10;
tol = 5e-5;
eps = 1e-6;
ex_power = 0.0;
/* Fit Log-linear model using nag_glm_poisson (g02gcc) */

/* nag_glm_poisson (g02gcc).
 * Fits a generalized linear model with Poisson errors
 */
nag_glm_poisson(Nag_Log, Nag_MeanInclude, n, x, tdx,
                m, sx, ip, y, wtptr, (double *) 0, ex_power, &dev, &df, b,
                &rank, se, cov, v, tdv, tol, max_iter, print_iter, "", eps,
                &fail);

if (fail.code == NE_NOERROR || fail.code == NE_LSQ_ITER_NOT_CONV ||
    fail.code == NE_RANK_CHANGED || fail.code == NE_ZERO_DOF_ERROR)
{
    printf("\nDeviance = %13.4e\n", dev);
    printf("Degrees of freedom = %3.1f\n\n", df);
    /* Input constraints */
    nclin = ip - rank;

    if (!(c = NAG_ALLOC(ip*nclin, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tdc = nclin;

    for (i = 0; i < ip; ++i)
        for (j = 0; j < nclin; ++j)
            scanf("%lf", &C(i, j));

    /* nag_glm_tran_model (g02gkc).
     * Estimates and standard errors of parameters of a general
     * linear model for given constraints
     */
    nag_glm_tran_model(ip, nclin, v, tdv, c,
                      tdc, b, 1.0e0, se, cov, &fail);

    if (fail.code == NE_NOERROR)
    {
        printf("      Estimate      Standard error\n\n");
        for (i = 0; i < ip; i++)
            printf(" %14.4f%14.4f\n", b[i], se[i]);
        printf("\n");
    }
    else
    {
        printf("Error from nag_glm_tran_model (g02gkc).\n%s\n",
              fail.message);
        exit_status = 1;
    }
}

```

```

        goto END;
    }
}
else
{
    printf("Error from nag_glm_poisson (g02gcc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(sx);
NAG_FREE(c);
NAG_FREE(b);
NAG_FREE(cov);
NAG_FREE(se);
NAG_FREE(v);
return exit_status;
}

```

## 10.2 Program Data

nag\_glm\_tran\_model (g02gkc) Example Program Data

```

15 8 0
1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 141.
1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 67.
1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 114.
1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 79.
1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 39.
0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 131.
0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 66.
0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 143.
0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 72.
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 35.
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 36.
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 14.
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 38.
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 28.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 16.
1 1 1 1 1 1 1 1 9
0.0 0.0
1.0 0.0
1.0 0.0
1.0 0.0
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0

```

### **10.3 Program Results**

nag\_glm\_tran\_model (g02gkc) Example Program Results

Deviance = 9.0379e+00  
Degrees of freedom = 8.0

Estimate	Standard error
3.9831	0.0396
0.3961	0.0458
0.4118	0.0457
-0.8079	0.0622
0.5112	0.0562
-0.2285	0.0727
0.4680	0.0569
-0.0316	0.0675
-0.7191	0.0887

---