NAG Library Routine Document

G01SKF

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

G01SKF returns a number of the lower tail, upper tail and point probabilities for the Poisson distribution.

2 Specification

```
SUBROUTINE G01SKF (LL, L, LK, K, PLEK, PGTK, PEQK, IVALID, IFAIL)
INTEGER            LL, LK, K(LK), IVALID(*), IFAIL
REAL (KIND=nag_wp) L(LL), PLEK(*), PGTK(*), PEQK(*)
```

3 Description

Let $X = \{X_i : i = 1, 2, \ldots, m\}$ denote a vector of random variables each having a Poisson distribution with parameter λ_i (> 0). Then

$$\text{Prob}\{X_i = k_i\} = e^{-\lambda_i}\frac{{\lambda_i}^{k_i}}{k_i!}, \quad k_i = 0, 1, 2, \ldots$$

The mean and variance of each distribution are both equal to λ_i.

G01SKF computes, for given λ_i and k_i the probabilities: $\text{Prob}\{X_i \leq k_i\}$, $\text{Prob}\{X_i > k_i\}$ and $\text{Prob}\{X_i = k_i\}$ using the algorithm described in KnÏsel (1986).

The input arrays to this routine are designed to allow maximum flexibility in the supply of vector arguments by re-using elements of any arrays that are shorter than the total number of evaluations required. See Section 2.6 in the G01 Chapter Introduction for further information.

4 References

KnÏsel L (1986) Computation of the chi-square and Poisson distribution *SIAM J. Sci. Statist. Comput.* **7** 1022–1036

5 Arguments

1: LL – INTEGER *Input*

On entry: the length of the array L

Constraint: LL > 0.

2: L(LL) – REAL (KIND=nag_wp) array *Input*

On entry: λ_i, the parameter of the Poisson distribution with $\lambda_i = \text{L}(j)$, $j = ((i-1) \bmod \text{LL}) + 1$, for $i = 1, 2, \ldots, \max(\text{LL}, \text{LK})$.

Constraint: $0.0 < \text{L}(j) \leq 10^6$, for $j = 1, 2, \ldots, \text{LL}$.

3: LK – INTEGER *Input*

On entry: the length of the array K

Constraint: LK > 0.

4: K(LK) – INTEGER array *Input*

On entry: k_i, the integer which defines the required probabilities with $k_i = \mathrm{K}(j)$, $j = ((i-1) \bmod \mathrm{LK}) + 1$.

Constraint: $\mathrm{K}(j) \geq 0$, for $j = 1, 2, \ldots, \mathrm{LK}$.

5: PLEK(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PLEK must be at least $\max(\mathrm{LL}, \mathrm{LK})$.

On exit: $\mathrm{Prob}\{X_i \leq k_i\}$, the lower tail probabilities.

6: PGTK(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PGTK must be at least $\max(\mathrm{LL}, \mathrm{LK})$.

On exit: $\mathrm{Prob}\{X_i > k_i\}$, the upper tail probabilities.

7: PEQK(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PEQK must be at least $\max(\mathrm{LL}, \mathrm{LK})$.

On exit: $\mathrm{Prob}\{X_i = k_i\}$, the point probabilities.

8: IVALID(∗) – INTEGER array *Output*

Note: the dimension of the array IVALID must be at least $\max(\mathrm{LL}, \mathrm{LK})$.

On exit: IVALID(i) indicates any errors with the input arguments, with

IVALID(i) $= 0$
No error.

IVALID(i) $= 1$

On entry, $\lambda_i \leq 0.0$.

IVALID(i) $= 2$

On entry, $k_i < 0$.

IVALID(i) $= 3$

On entry, $\lambda_i > 10^6$.

9: 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 $= 1$

On entry, at least one value of L or K was invalid.
Check IVALID for more information.

IFAIL $= 2$

On entry, array size $= \langle value \rangle$.
Constraint: LL > 0.

IFAIL $= 3$

On entry, array size $= \langle value \rangle$.
Constraint: LK > 0.

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

Results are correct to a relative accuracy of at least 10^{-6} on machines with a precision of 9 or more decimal digits (provided that the results do not underflow to zero).

8 Parallelism and Performance

G01SKF is not threaded in any implementation.

9 Further Comments

The time taken by G01SKF to calculate each probability depends on λ_i and k_i. For given λ_i, the time is greatest when $k_i \approx \lambda_i$, and is then approximately proportional to $\sqrt{\lambda_i}$.

10 Example

This example reads a vector of values for λ and k, and prints the corresponding probabilities.

10.1 Program Text

```
    Program g01skfe
!     G01SKF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01skf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lk, ll, lout
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: l(:), peqk(:), pgtk(:), plek(:)
      Integer, Allocatable             :: ivalid(:), k(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, mod, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G01SKF Example Program Results'
      Write (nout,*)

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

!     Read in the input vectors
      Read (nin,*) ll
      Allocate (l(ll))
      Read (nin,*) l(1:ll)

      Read (nin,*) lk
      Allocate (k(lk))
      Read (nin,*) k(1:lk)

!     Allocate memory for output
      lout = max(ll,lk)
      Allocate (peqk(lout),pgtk(lout),plek(lout),ivalid(lout))

!     Calculate probability
      ifail = -1
      Call g01skf(ll,l,lk,k,plek,pgtk,peqk,ivalid,ifail)

      If (ifail==0 .Or. ifail==1) Then
!       Display titles
        Write (nout,*)                                                    &
          '    L         K       PLEK      PGTK      PEQK    IVALID'
        Write (nout,*) repeat('-',58)

!       Display results
        Do i = 1, lout
          Write (nout,99999) l(mod(i-1,ll)+1), k(mod(i-1,lk)+1), plek(i),      &
            pgtk(i), peqk(i), ivalid(i)
        End Do
      End If

99999 Format (1X,F6.2,4X,I6,3(4X,F6.3),4X,I3)
    End Program g01skfe
```

10.2 Program Data

```
G01SKF Example Program Data
4                       :: LL
0.75 9.20 34.0 175.0    :: L
4                       :: LK
3 12 25 175             :: K
```

10.3 Program Results

```
 G01SKF Example Program Results

     L         K       PLEK      PGTK      PEQK     IVALID
 -------------------------------------------------------------
   0.75         3      0.993     0.007     0.033       0
   9.20        12      0.861     0.139     0.078       0
  34.00        25      0.067     0.933     0.021       0
 175.00       175      0.520     0.480     0.030       0
```