NAG Library Routine Document

G01ANF

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

G01ANF finds approximate quantiles from a data stream of known size using an out-of-core algorithm.

2 Specification

```
SUBROUTINE G01ANF (IND, N, RV, NB, EPS, NP, Q, QV, NQ, RCOMM, LRCOMM,      &
                   ICOMM, LICOMM, IFAIL)
INTEGER            IND, N, NB, NP, NQ, LRCOMM, ICOMM(LICOMM), LICOMM,      &
                   IFAIL
REAL (KIND=nag_wp) RV(*), EPS, Q(*), QV(*), RCOMM(LRCOMM)
```

3 Description

A quantile is a value which divides a frequency distribution such that there is a given proportion of data values below the quantile. For example, the median of a dataset is the 0.5 quantile because half the values are less than or equal to it.

G01ANF uses a slightly modified version of an algorithm described in a paper by Zhang and Wang (2007) to determine ϵ-approximate quantiles of a data stream of n real values, where n is known. Given any quantile $q \in [0.0, 1.0]$, an ϵ-approximate quantile is defined as an element in the data stream whose rank falls within $[(q-\epsilon)n, (q+\epsilon)n]$. In case of more than one ϵ-approximate quantile being available, the one closest to qn is returned.

4 References

Zhang Q and Wang W (2007) A fast algorithm for approximate quantiles in high speed data streams *Proceedings of the 19th International Conference on Scientific and Statistical Database Management* IEEE Computer Society 29

5 Arguments

1: IND – INTEGER *Input/Output*

On entry: indicates the action required in the current call to G01ANF.

IND $= 0$
: Return the required length of RCOMM and ICOMM in ICOMM(1) and ICOMM(2) respectively. N and EPS must be set and LICOMM must be at least 2.

IND $= 1$
: Initialise the communication arrays and process the first NB values from the data stream as supplied in RV.

IND $= 2$
: Process the next block of NB values from the data stream. The calling program must update RV and (if required) NB, and re-enter G01ANF with all other parameters unchanged.

IND $= 3$
Calculate the NQ ϵ-approximate quantiles specified in Q. The calling program must set Q and NQ and re-enter G01ANF with all other parameters unchanged. This option can be chosen only when NP $\geq \lceil \exp(1.0)/\text{EPS} \rceil$.

On exit: indicates output from a successful call.

IND $= 1$
Lengths of RCOMM and ICOMM have been returned in ICOMM(1) and ICOMM(2) respectively.

IND $= 2$
G01ANF has processed NP data points and expects to be called again with additional data (i.e., NP $<$ N).

IND $= 3$
G01ANF has returned the requested ϵ-approximate quantiles in QV. These quantiles are based on NP data points.

IND $= 4$
Routine has processed all N data points (i.e., NP $=$ N).

Constraint: on entry IND $= 0$, 1, 2 or 3.

2: N – INTEGER *Input*

On entry: n, the total number of values in the data stream.

Constraint: N > 0.

3: RV$(*)$ – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array RV must be at least NB if IND $= 1$ or 2.

On entry: if IND $= 1$ or 2, the vector containing the current block of data, otherwise RV is not referenced.

4: NB – INTEGER *Input*

On entry: if IND $= 1$ or 2, the size of the current block of data. The size of blocks of data in array RV can vary; therefore NB can change between calls to G01ANF.

Constraint: if IND $= 1$ or 2, NB > 0.

5: EPS – REAL (KIND=nag_wp) *Input*

On entry: approximation factor ϵ.

Constraint: EPS $\geq \exp(1.0)/\text{N}$ and EPS ≤ 1.0.

6: NP – INTEGER *Output*

On exit: the number of elements processed so far.

7: Q$(*)$ – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array Q must be at least NQ if IND $= 3$.

On entry: if IND $= 3$, the quantiles to be calculated, otherwise Q is not referenced. Note that Q$(i) = 0.0$, corresponds to the minimum value and Q$(i) = 1.0$ to the maximum value.

Constraint: if IND $= 3$, $0.0 \leq \text{Q}(i) \leq 1.0$, for $i = 1, 2, \ldots, \text{NQ}$.

8: QV(∗) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array QV must be at least NQ if IND = 3.

On exit: if IND = 3, QV(i) contains the ϵ-approximate quantiles specified by the value provided in Q(i).

9: NQ – INTEGER *Input*

On entry: if IND = 3, the number of quantiles requested, otherwise NQ is not referenced.

Constraint: if IND = 3, NQ > 0.

10: RCOMM(LRCOMM) – REAL (KIND=nag_wp) array *Communication Array*

11: LRCOMM – INTEGER *Input*

On entry: the dimension of the array RCOMM as declared in the (sub)program from which G01ANF is called.

Constraint: if IND ≠ 0, LRCOMM must be at least equal to the value returned in ICOMM(1) by a call to G01ANF with IND = 0. This will not be more than $x + 2 \times \min(x, \lceil x/2.0 \rceil + 1) \times \log_2(\mathrm{N}/x + 1.0) + 1$, where $x = \max(1, \lfloor \log(\mathrm{EPS} \times \mathrm{N})/\mathrm{EPS} \rfloor)$.

12: ICOMM(LICOMM) – INTEGER array *Communication Array*

13: LICOMM – INTEGER *Input*

On entry: the dimension of the array ICOMM as declared in the (sub)program from which G01ANF is called.

Constraints:

if IND = 0, LICOMM ≥ 2;
otherwise LICOMM must be at least equal to the value returned in ICOMM(2) by a call to G01ANF with IND = 0. This will not be more than $2 \times (x + 2 \times \min(x, \lceil x/2.0 \rceil + 1) \times y) + y + 6$, where $x = \max(1, \lfloor \log(\mathrm{EPS} \times \mathrm{N})/\mathrm{EPS} \rfloor)$ and $y = \log_2(\mathrm{N}/x + 1.0) + 1$.

14: 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.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

As an out-of-core routine G01ANF will only perform certain argument checks when a data checkpoint (including completion of data input) is signaled. As such it will usually be inappropriate to halt program execution when an error is detected since any errors may be subsequently resolved without losing any processing already carried out. Therefore setting IFAIL to a value of −1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

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, IND = ⟨*value*⟩.
Constraint: IND = 0, 1, 2 or 3.

IFAIL $= 2$

On entry, N $= \langle value \rangle$.
Constraint: N > 0.

IFAIL $= 3$

On entry, EPS $= \langle value \rangle$.
Constraint: $\exp(1.0)/\text{N} \leq \text{EPS} \leq 1.0$.

IFAIL $= 4$

On entry, IND $= 1$ or 2 and NB $= \langle value \rangle$.
Constraint: if IND $= 1$ or 2 then NB > 0.

IFAIL $= 5$

On entry, LICOMM is too small: LICOMM $= \langle value \rangle$.

IFAIL $= 6$

On entry, LRCOMM is too small: LRCOMM $= \langle value \rangle$.

IFAIL $= 7$

Number of data elements streamed, $\langle value \rangle$ is not sufficient for a quantile query when EPS $= \langle value \rangle$.
Supply more data or reprocess the data with a higher EPS value.

IFAIL $= 8$

On entry, IND $= 3$ and NQ $= \langle value \rangle$.
Constraint: if IND $= 3$ then NQ > 0.

IFAIL $= 9$

On entry, IND $= 3$ and Q($\langle value \rangle$) $= \langle value \rangle$.
Constraint: if IND $= 3$ then $0.0 \leq \text{Q}(i) \leq 1.0$ for all i.

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

G01ANF 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 average time taken by G01ANF is $\mathrm{N}\log\left(1/\epsilon\log\left(\epsilon\mathrm{N}\right)\right)$.

10 Example

This example calculates ϵ-approximate quantile for $q = 0.25$, 0.5 and 1.0 for a data stream of 60 values. The stream is read in four blocks of varying size.

10.1 Program Text

```
    Program g01anfe

!     G01ANF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01anf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: eps
      Integer                          :: i, ifail, ind, licomm, lrcomm, n,    &
                                          nb, np, nq, nrv, onb
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: q(:), qv(:), rcomm(:), rv(:)
      Integer, Allocatable             :: icomm(:)
!     .. Executable Statements ..
      Write (nout,*) 'G01ANF Example Program Results'
      Write (nout,*)

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

!     Read in stream size and approximation factor
      Read (nin,*) n, eps

!     Read in number of elements in the output vector qv
      Read (nin,*) nq
      Allocate (qv(nq),q(nq))

!     Read in vector q
      Read (nin,*) q(1:nq)

!     Dummy allocation for the communication arrays
      lrcomm = 1
      licomm = 2
      nb = 1
      Allocate (rv(nb),rcomm(lrcomm),icomm(licomm))

!     Call NAG routine for the first time to obtain lrcomm and licomm
      ind = 0
      ifail = 0
      Call g01anf(ind,n,rv,nb,eps,np,q,qv,nq,rcomm,lrcomm,icomm,licomm,ifail)

!     Reallocate the communication arrays to the required size
      lrcomm = icomm(1)
      licomm = icomm(2)
      Deallocate (rcomm,icomm)
      Allocate (rcomm(lrcomm),icomm(licomm))
```

```
!     Read in number of vectors with dataset blocks
      Read (nin,*) nrv

      onb = 0
d_lp: Do i = 1, nrv
!       Read in number of elements in the first/next vector rv
        Read (nin,*) nb

        If (onb/=nb) Then
!         Reallocate RV if required
          Deallocate (rv)
          Allocate (rv(nb))
        End If
        onb = nb

!       Read in vector rv
        Read (nin,*) rv(1:nb)

!       Repeat calls to NAG routine for every dataset block rv
!       until n observations have been passed
        ifail = 1
        Call g01anf(ind,n,rv,nb,eps,np,q,qv,nq,rcomm,lrcomm,icomm,licomm,      &
          ifail)
        If (ifail/=0) Then
!         This routine is most likely to be used to process large datasets,
!         certain parameter checks will only be done once all the data has
!         been processed. Calling the routine with a hard failure (IFAIL=0)
!         would cause any processing to be lost as the program terminates.
!         It is likely that a soft failure would be more appropriate. This
!         would allow any issues with the input parameters to be resolved
!         without losing any processing already carried out.

!         In this small example we are just calling the routine again with
!         a hard failure so that the error messages are displayed.
          ifail = 0
          Call g01anf(ind,n,rv,nb,eps,np,q,qv,nq,rcomm,lrcomm,icomm,licomm,    &
            ifail)
        End If
        If (ind==4) Then
          Exit d_lp
        End If
      End Do d_lp

!     Call NAG routine again to calculate quantiles specified in vector q
      ind = 3
      ifail = 0
      Call g01anf(ind,n,rv,nb,eps,np,q,qv,nq,rcomm,lrcomm,icomm,licomm,ifail)

!     Print the results
      Write (nout,*) 'Input data:'
      Write (nout,99999) n, ' observations'
      Write (nout,99998) 'eps = ', eps
      Write (nout,*)
      Write (nout,*) 'Quantile     Result'
      Write (nout,99997)(q(i),qv(i),i=1,nq)

99999 Format (1X,I2,A)
99998 Format (1X,A,F5.2)
99997 Format (1X,F7.2,4X,F7.2)
    End Program g01anfe
```

10.2 Program Data

```
G01ANF Example Program Data
60  0.2                                                    : N, EPS
3                                                          : NQ
0.25 0.5 1.0                                               : Q
4                                                          : NRV
16                                                         : 1st LRV
34.01  57.95  44.88  22.04  28.84   4.43   0.32  20.82
```

```
20.53  13.08  7.99   54.03  23.21  26.73  39.72   0.97     : 1st RV
24                                                        : 2nd LRV
39.05  38.78  19.38  51.34  24.08  12.41  58.11  35.90
40.38  27.41  19.80   6.02  45.33  36.34  43.14  53.84
39.49   9.04  36.74  58.72  59.95  15.41  33.05  39.54     : 2nd RV
8                                                         : 3rd LRV
33.24  58.67  54.12  39.48  43.73  24.15  55.72   8.87     : 3rd RV
12                                                        : 4th LRV
40.47  46.18  20.36   6.95  36.86  49.24  56.83  43.87
29.86  22.49  25.29  33.17                                 : 4th RV
```

10.3 Program Results

```
 G01ANF Example Program Results

 Input data:
 60 observations
 eps =  0.20

 Quantile     Result
    0.25      22.49
    0.50      36.86
    1.00      59.95
```