NAG Library Routine Document

G01APF

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

G01APF finds approximate quantiles from a large arbitrary-sized data stream using an out-of-core algorithm.

2 Specification

```
SUBROUTINE G01APF (IND, RV, NB, EPS, NP, Q, QV, NQ, RCOMM, LRCOMM,          &
                   ICOMM, LICOMM, IFAIL)
INTEGER            IND, 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.

G01APF uses a slightly modified version of an algorithm described in a paper by Zhang and Wang (2007) to determine ϵ-approximate quantiles of a large arbitrary-sized data stream of real values, where ϵ is a user-defined approximation factor. Let m denote the number of data elements processed so far then, 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 $[(q-\epsilon)m, (q+\epsilon)m]$. In case of more than one ϵ-approximate quantile being available, the one closest to qm is used.

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 initial entry: must be set to 0.

On entry: indicates the action required in the current call to G01APF.

IND $= 0$
: Initialize the communication arrays and attempt to process the first NB values from the data stream. EPS, RV and NB must be set and LICOMM must be at least 10.

IND $= 1$
: Attempt to process the next block of NB values from the data stream. The calling program must update RV and (if required) NB, and re-enter G01APF with all other parameters unchanged.

IND $= 2$
: Continue calculation following the reallocation of either or both of the communication arrays RCOMM and ICOMM.

IND $= 3$
Calculate the NQ ϵ-approximate quantiles specified in Q. The calling program must set Q and NQ and re-enter G01APF 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 the call.

IND $= 1$
G01APF has processed NP data points and expects to be called again with additional data.

IND $= 2$
Either one or more of the communication arrays RCOMM and ICOMM is too small. The new minimum lengths of RCOMM and ICOMM have been returned in ICOMM(1) and ICOMM(2) respectively. If the new minimum length is greater than the current length then the corresponding communication array needs to be reallocated, its contents preserved and G01APF called again with all other parameters unchanged.

If there is more data to be processed, it is recommended that LRCOMM and LICOMM are made significantly bigger than the minimum to limit the number of reallocations.

IND $= 3$
G01APF has returned the requested ϵ-approximate quantiles in QV. These quantiles are based on NP data points.

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

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

Note: the dimension of the array RV must be at least NB if IND $= 0$, 1 or 2.

On entry: if IND $= 0$, 1 or 2, the vector containing the current block of data, otherwise RV is not referenced.

3: NB – INTEGER *Input*

On entry: if IND $= 0$, 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 G01APF.

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

4: EPS – REAL (KIND=nag_wp) *Input*

On entry: approximation factor ϵ.

Constraint: EPS > 0.0 and EPS ≤ 1.0.

5: NP – INTEGER *Output*

On exit: m, the number of elements processed so far.

6: 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 $i = 1, 2, \ldots, \text{NQ}$.

7: 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).

8: NQ – INTEGER *Input*

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

Constraint: if IND = 3, NQ > 0.

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

On entry: if IND = 1 or 2 then the first l elements of RCOMM as supplied to G01APF must be identical to the first l elements of RCOMM returned from the last call to G01APF, where l is the value of LRCOMM used in the last call. In other words, the contents of RCOMM must not be altered between calls to this routine. If RCOMM needs to be reallocated then its contents must be preserved. If IND = 0 then RCOMM need not be set.

On exit: RCOMM holds information required by subsequent calls to G01APF

10: LRCOMM – INTEGER *Input*

On entry: the dimension of the array RCOMM as declared in the (sub)program from which G01APF is called.

Constraints:

if IND = 0, LRCOMM $\geq$ 1;
otherwise LRCOMM $\geq$ ICOMM(1).

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

On entry: if IND = 1 or 2 then the first l elements of ICOMM as supplied to G01APF must be identical to the first l elements of ICOMM returned from the last call to G01APF, where l is the value of LICOMM used in the last call. In other words, the contents of ICOMM must not be altered between calls to this routine. If ICOMM needs to be reallocated then its contents must be preserved. If IND = 0 then ICOMM need not be set.

On exit: ICOMM(1) holds the minimum required length for RCOMM and ICOMM(2) holds the minimum required length for ICOMM. The remaining elements of ICOMM are used for communication between subsequent calls to G01APF.

12: LICOMM – INTEGER *Input*

On entry: the dimension of the array ICOMM as declared in the (sub)program from which G01APF is called.

Constraints:

if IND = 0, LICOMM $\geq$ 10;
otherwise LICOMM $\geq$ ICOMM(2).

13: 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 G01APF 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 $= \langle value \rangle$.
Constraint: IND $= 0$, 1, 2 or 3.

IFAIL $= 2$

On entry, EPS $= \langle value \rangle$.
Constraint: $0.0 <$ EPS ≤ 1.0.

IFAIL $= 3$

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

IFAIL $= 4$

On entry, LICOMM $= \langle value \rangle$.
Constraint: LICOMM ≥ 10.

IFAIL $= 5$

On entry, LRCOMM $= \langle value \rangle$.
Constraint: LRCOMM ≥ 1.

IFAIL $= 6$

The contents of ICOMM have been altered between calls to this routine.

IFAIL $= 7$

The contents of RCOMM have been altered between calls to this routine.

IFAIL $= 8$

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 $= 9$

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

IFAIL $= 10$

On entry, IND $= 3$ and Q$(\langle value \rangle) = \langle value \rangle$.
Constraint: if IND $= 3$ then $0.0 \leq$ 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

G01APF 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 G01APF scales as $\mathrm{NP}\log(1/\epsilon\log(\epsilon\mathrm{NP}))$.

It is not possible to determine in advance the final size of the communication arrays RCOMM and ICOMM without knowing the size of the dataset. However, if a rough size (n) is known, the speed of the computation can be increased if the sizes of the communication arrays are not smaller than

$$\begin{aligned}\mathrm{LRCOMM} &= (\log_2(n\times\mathrm{EPS}+1.0)-2)\times\lceil 1.0/\mathrm{EPS}\rceil+1+x+2\times\min(x,\lceil x/2.0\rceil+1)\times y+1\\ \mathrm{LICOMM} &= (\log_2(n\times\mathrm{EPS}+1.0)-2)\times(2\times(\lceil 1.0/\mathrm{EPS}\rceil+1)+1)+\\ &\quad 2\times(x+2\times\min(x,\lceil x/2.0\rceil+1)\times y)+y+11\end{aligned}$$

where

$$\begin{aligned}x &= \max(1,\lfloor\log(\mathrm{EPS}\times n)/\mathrm{EPS}\rfloor)\\ y &= \log_2(n/x+1.0)+1.\end{aligned}$$

10 Example

This example computes a list of ϵ-approximate quantiles. The data is processed in blocks of 20 observations at a time to simulate a situation in which the data is made available in a piecemeal fashion.

10.1 Program Text

```
    Program g01apfe

!     G01APF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01apf, 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,       &
                                          ltcomm, n, nb, np, nq
      Logical                          :: repeat
```

```
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: q(:), qv(:), rcomm(:), rv(:),       &
                                          trcomm(:)
      Integer, Allocatable             :: icomm(:), ticomm(:)
!     .. Executable Statements ..
      Write (nout,*) 'G01APF Example Program Results'
      Write (nout,*)

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

!     Read in approximation factor
      Read (nin,*) 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)

      lrcomm = 100
      licomm = 400
      nb = 20
      Allocate (rcomm(lrcomm),icomm(licomm),rv(nb))

      ind = 0
      repeat = .True.
      n = 0

m_lp: Do While (repeat)

        If (ind==0 .Or. ind==1) Then
d_lp:     Do i = 1, nb
            Read (nin,*,Iostat=ifail) rv(i)
            If (ifail/=0) Then
              Exit d_lp
            End If
          End Do d_lp

          If (i==1) Then
            Exit m_lp
          Else If (i-1<nb) Then
            nb = i - 1
            repeat = .False.
          End If
          n = n + nb
        End If

!       Call the routine
        ifail = 1
        Call g01apf(ind,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 g01apf(ind,rv,nb,eps,np,q,qv,nq,rcomm,lrcomm,icomm,licomm,      &
            ifail)
        End If

!       If ind=2, the communication arrays are too small.
!       Allocate more memory, copy the content back to the communication
!       arrays and call the routine again with the same rv
```

```
        If (ind==2) Then
          If (lrcomm<icomm(1)) Then
            ltcomm = lrcomm
            lrcomm = icomm(1)
            Allocate (trcomm(ltcomm))
            trcomm(1:ltcomm) = rcomm(1:ltcomm)
            Deallocate (rcomm)
            Allocate (rcomm(lrcomm))
            rcomm(1:ltcomm) = trcomm(1:ltcomm)
            Deallocate (trcomm)
          End If
          If (licomm<icomm(2)) Then
            ltcomm = licomm
            licomm = icomm(2)
            Allocate (ticomm(ltcomm))
            ticomm(1:ltcomm) = icomm(1:ltcomm)
            Deallocate (icomm)
            Allocate (icomm(licomm))
            icomm(1:ltcomm) = ticomm(1:ltcomm)
            Deallocate (ticomm)
          End If
        End If
      End Do m_lp

!     Call NAG again with ind=3 to calculate quantiles specified in vector q
      ind = 3
      ifail = 0
      Call g01apf(ind,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 g01apfe
```

10.2 Program Data

```
G01APF Example Program Data
0.2                                                          : EPS
3                                                            : NQ
0.25 0.5 1.0                                                 : Q
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
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
33.24
58.67
54.12
39.48
43.73
24.15
55.72
 8.87
40.47
46.18
20.36
 6.95
36.86
49.24
56.83
43.87
29.86
22.49
25.29
33.17
```

10.3 Program Results

```
 G01APF Example Program Results

 Input data:
 60 observations
 eps =  0.20

 Quantile     Result
    0.25      22.49
    0.50      39.54
    1.00      59.95
```