

# NAG Library Routine Document

## G01WAF

**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

G01WAF calculates the mean and, optionally, the standard deviation using a rolling window for an arbitrary sized data stream.

### 2 Specification

```
SUBROUTINE G01WAF (M, NB, X, IWT, WT, PN, RMEAN, RSD, LRS, RCOMM,      &
                   LRCOMM, IFAIL)
INTEGER               M, NB, IWT, PN, LRS, LRCOMM, IFAIL
REAL (KIND=nag_wp) X(NB), WT(*), RMEAN(max(0,NB+min(0,PN-M+1))),    &
                   RSD(LRS), RCOMM(LRCOMM)
```

### 3 Description

Given a sample of  $n$  observations, denoted by  $x = \{x_i : i = 1, 2, \dots, n\}$  and a set of weights,  $w = \{w_j : j = 1, 2, \dots, m\}$ , G01WAF calculates the mean and, optionally, the standard deviation, in a rolling window of length  $m$ .

For the  $i$ th window the mean is defined as

$$\mu_i = \frac{\sum_{j=1}^m w_j x_{i+j-1}}{W} \quad (1)$$

and the standard deviation as

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^m w_j (x_{i+j-1} - \mu_i)^2}{W - \frac{\sum_{j=1}^m w_j^2}{W}}} \quad (2)$$

with  $W = \sum_{j=1}^m w_j$ .

Four different types of weighting are possible:

(i) **No weights ( $w_j = 1$ )**

When no weights are required both the mean and standard deviations can be calculated in an iterative manner, with

$$\begin{aligned}\mu_{i+1} &= \mu_i + \frac{(x_{i+m} - x_i)}{m} \\ \sigma_{i+1}^2 &= (m-1)\sigma_i^2 + (x_{i+m} - \mu_i)^2 - (x_i - \mu_i)^2 - \frac{(x_{i+m} - x_i)^2}{m}\end{aligned}$$

where the initial values  $\mu_1$  and  $\sigma_1$  are obtained using the one pass algorithm of West (1979).

**(ii) Each observation has its own weight**

In this case, rather than supplying a vector of  $m$  weights a vector of  $n$  weights is supplied instead,  $v = \{v_j : j = 1, 2, \dots, n\}$  and  $w_j = v_{i+j-1}$  in (1) and (2).

If the standard deviations are not required then the mean is calculated using the iterative formula:

$$\begin{aligned}W_{i+1} &= W_i + (v_{i+m} - v_i) \\ \mu_{i+1} &= \mu_i + W_i^{-1}(v_{i+m}x_{i+m} - v_ix_i)\end{aligned}$$

where  $W_1 = \sum_{i=1}^m v_i$  and  $\mu_1 = W_1^{-1} \sum_{i=1}^m v_ix_i$ .

If both the mean and standard deviation are required then the one pass algorithm of West (1979) is used in each window.

**(iii) Each position in the window has its own weight**

This is the case as described in (1) and (2), where the weight given to each observation differs depending on which summary is being produced. When these types of weights are specified both the mean and standard deviation are calculated by applying the one pass algorithm of West (1979) multiple times.

**(iv) Each position in the window has a weight equal to its position number ( $w_j = j$ )**

This is a special case of (iii).

If the standard deviations are not required then the mean is calculated using the iterative formula:

$$\begin{aligned}S_{i+1} &= S_i + (x_{i+m} - x_i) \\ \mu_{i+1} &= \mu_i + \frac{2(mx_{i+m} - S_i)}{m(m+1)}\end{aligned}$$

where  $S_1 = \sum_{i=1}^m x_i$  and  $\mu_1 = 2(m^2 + m)^{-1}S_1$ .

If both the mean and standard deviation are required then the one pass algorithm of West is applied multiple times.

For large datasets, or where all the data is not available at the same time,  $x$  (and if each observation has its own weight,  $v$ ) can be split into arbitrary sized blocks and G01WAF called multiple times.

## 4 References

Chan T F, Golub G H and Leveque R J (1982) *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances* Compstat, Physica-Verlag

West D H D (1979) Updating mean and variance estimates: An improved method *Comm. ACM* **22** 532–555

## 5 Arguments

1: M – INTEGER

*Input*

*On entry:*  $m$ , the length of the rolling window.

If  $PN \neq 0$ , M must be unchanged since the last call to G01WAF.

*Constraint:*  $M \geq 1$ .

## 2: NB – INTEGER

Input

*On entry:*  $b$ , the number of observations in the current block of data. The size of the block of data supplied in  $X$  (and when  $IWT = 1$ ,  $WT$ ) can vary; therefore  $NB$  can change between calls to G01WAF.

*Constraints:*

$NB \geq 0$ ;  
if  $LRCOMM = 0$ ,  $NB \geq M$ .

## 3: X(NB) – REAL (KIND=nag\_wp) array

Input

*On entry:* the current block of observations, corresponding to  $x_i$ , for  $i = k + 1, \dots, k + b$ , where  $k$  is the number of observations processed so far and  $b$  is the size of the current block of data.

## 4: IWT – INTEGER

Input

*On entry:* the type of weighting to use.

$IWT = 0$

No weights are used.

$IWT = 1$

Each observation has its own weight.

$IWT = 2$

Each position in the window has its own weight.

$IWT = 3$

Each position in the window has a weight equal to its position number.

If  $PN \neq 0$ ,  $IWT$  must be unchanged since the last call to G01WAF.

*Constraint:*  $IWT = 0, 1, 2$  or  $3$ .

## 5: WT(\*) – REAL (KIND=nag\_wp) array

Input

**Note:** the dimension of the array  $WT$  must be at least  $NB$  if  $IWT = 1$  and at least  $M$  if  $IWT = 2$ .

*On entry:* the user-supplied weights.

If  $IWT = 1$ ,  $WT(i) = \nu_{i+k}$ , for  $i = 1, 2, \dots, b$ .

If  $IWT = 2$ ,  $WT(j) = w_j$ , for  $j = 1, 2, \dots, m$ .

Otherwise,  $WT$  is not referenced.

*Constraints:*

if  $IWT = 1$ ,  $WT(i) \geq 0$ , for  $i = 1, 2, \dots, NB$ ;  
if  $IWT = 2$ ,  $WT(1) \neq 0$  and  $\sum_{j=1}^m WT(j) > 0$ ;  
if  $IWT = 2$  and  $LRS D \neq 0$ ,  $WT(j) \geq 0$ , for  $j = 1, 2, \dots, M$ .

## 6: PN – INTEGER

Input/Output

*On entry:*  $k$ , the number of observations processed so far. On the first call to G01WAF, or when starting to summarise a new dataset,  $PN$  must be set to 0.

If  $PN \neq 0$ , it must be the same value as returned by the last call to G01WAF.

*On exit:*  $k + b$ , the updated number of observations processed so far.

*Constraint:*  $PN \geq 0$ .

## 7: RMEAN(max(0, NB + min(0, PN – M + 1))) – REAL (KIND=nag\_wp) array

Output

*On exit:*  $\mu_l$ , the (weighted) moving averages, for  $l = 1, 2, \dots, b + \min(0, k - m + 1)$ . Therefore,  $\mu_l$  is the mean of the data in the window that ends on  $X(l + m - \min(k, m - 1) - 1)$ .

If, on entry,  $PN \geq M - 1$ , i.e., at least one windows worth of data has been previously processed, then  $RMEAN(l)$  is the summary corresponding to the window that ends on  $X(l)$ . On the other hand, if, on entry,  $PN = 0$ , i.e., no data has been previously processed, then  $RMEAN(l)$  is the summary corresponding to the window that ends on  $X(M + l - 1)$  (or, equivalently, starts on  $X(l)$ ).

- 8: RSD(LRSD) – REAL (KIND=nag\_wp) array *Output*

*On exit:* if  $LRSD \neq 0$  then  $\sigma_l$ , the (weighted) standard deviation. The ordering of RSD is the same as the ordering of RMEAN.

If  $LRSD = 0$ , RSD is not referenced.

- 9: LRSD – INTEGER *Input*

*On entry:* the dimension of the array RSD as declared in the (sub)program from which G01WAF is called. If the standard deviations are not required then LRSD should be set to zero.

*Constraint:*  $LRSD = 0$  or  $LRSD \geq \max(0, NB + \min(0, PN - M + 1))$ .

- 10: RCOMM(LRCOMM) – REAL (KIND=nag\_wp) array *Communication Array*

*On entry:* communication array, used to store information between calls to G01WAF. If  $LRCOMM = 0$ , RCOMM is not referenced and all the data must be supplied in one go.

- 11: LRCOMM – INTEGER *Input*

*On entry:* the dimension of the array RCOMM as declared in the (sub)program from which G01WAF is called.

*Constraint:*  $LRCOMM = 0$  or  $LRCOMM \geq 2M + 20$ .

- 12: 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

On entry,  $M = \langle value \rangle$ .

Constraint:  $M \geq 1$ .

IFAIL = 12

On entry,  $M = \langle value \rangle$ .

On entry at previous call,  $M = \langle value \rangle$ .

Constraint: if  $PN > 0$ , M must be unchanged since previous call.

IFAIL = 21

On entry, NB =  $\langle value \rangle$ .  
Constraint: NB  $\geq$  0.

IFAIL = 22

On entry, NB =  $\langle value \rangle$ , M =  $\langle value \rangle$ .  
Constraint: if LRCOMM = 0, NB  $\geq$  M.

IFAIL = 41

On entry, IWT =  $\langle value \rangle$ .  
Constraint: IWT = 0, 1, 2 or 3.

IFAIL = 42

On entry, IWT =  $\langle value \rangle$ .  
On entry at previous call, IWT =  $\langle value \rangle$ .  
Constraint: if PN > 0, IWT must be unchanged since previous call.

IFAIL = 51

On entry, WT( $\langle value \rangle$ ) =  $\langle value \rangle$ .  
Constraint: WT( $i$ )  $\geq$  0.

IFAIL = 52

On entry, WT(1) =  $\langle value \rangle$ .  
Constraint: if IWT = 2, WT(1) > 0.

IFAIL = 53

On entry, at least one window had all zero weights.

IFAIL = 54

On entry, unable to calculate at least one standard deviation due to the weights supplied.

IFAIL = 55

On entry, sum of weights supplied in WT is  $\langle value \rangle$ .  
Constraint: if IWT = 2, the sum of the weights > 0.

IFAIL = 61

On entry, PN =  $\langle value \rangle$ .  
Constraint: PN  $\geq$  0.

IFAIL = 62

On entry, PN =  $\langle value \rangle$ .  
On exit from previous call, PN =  $\langle value \rangle$ .  
Constraint: if PN > 0, PN must be unchanged since previous call.

IFAIL = 91

On entry, LRSD =  $\langle value \rangle$ .  
Constraint: LRSD = 0 or LRSD  $\geq$   $\langle value \rangle$ .

IFAIL = 101

RCOMM has been corrupted between calls.

IFAIL = 111

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

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

G01WAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G01WAF 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

The more data that is supplied to G01WAF in one call, i.e., the larger NB is, the more efficient the routine will be.

## 10 Example

This example calculates Spencer's 15-point moving average for the change in rate of the Earth's rotation between 1821 and 1850. The data is supplied in three chunks, the first consisting of five observations, the second 10 observations and the last 15 observations.

### 10.1 Program Text

```

Program g01wafe

!      G01WAF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
      Use nag_library, Only: g01waf, nag_wp
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..

```

```

Integer, Parameter :: nin = 5, nout = 6
! .. Local Scalars ..
Integer :: i, ierr, ifail, iwt, lrcomm, lrstd, &
m, nb, nsummaries, offset, pn
Logical :: want_sd
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: rcomm(:), rmean(:), rsd(:), wt(:), &
x(:)
! .. Intrinsic Procedures ..
Intrinsic :: allocated, max, min
! .. Executable Statements ..
Write (nout,*) 'G01WAF Example Program Results'
Write (nout,*)

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

! Read in the problem type
Read (nin,*) iwt, m

! Read in a flag indicating whether we want the standard deviations
Read (nin,*) want_sd

! Initial handling of weights
Select Case (iwt)
Case (1)
! Weights will be read in with the data
Case (2)
! Each observation in the rolling window has its own weight
Allocate (wt(m))
Read (nin,*) wt(1:m)
Case Default
! No weights need supplying
Allocate (wt(0))
End Select

lrcomm = 2*m + 20
Allocate (rcomm(lrcomm))

! Print some titles
If (want_sd) Then
Write (nout,99997) '
Write (nout,99997) ' Interval          Mean          Standard'
Write (nout,99997) '-----'
Else
Write (nout,99997) ' Interval          Mean '
Write (nout,99997) '-----'
End If

! Loop over each block of data
pn = 0
blk_lp: Do
! Read in the number of observations in this block
Read (nin,*,Iostat=ierr) nb
If (ierr/=0) Then
Exit blk_lp
End If
! Reallocate X to the required size
If (allocated(x)) Then
Deallocate (x)
End If
Allocate (x(nb))

! Read in the data for this block
Read (nin,*) x(1:nb)

! If (iwt==1) Then
! User supplied weights are present

! Reallocate WT to the required size
If (allocated(wt)) Then

```

```

        Deallocate (wt)
      End If
      Allocate (wt(nb))

!      Read in the weights for this block
      Read (nin,*) wt(1:nb)
    End If

!      Calculate the number of summaries we can produce
      nsummaries = max(0,nb+min(0,pn-m+1))
      If (want_sd) Then
        lrsd = nsummaries
      Else
        lrsd = 0
      End If

!      Reallocate the output arrays
      If (allocated(rmean)) Then
        Deallocate (rmean)
      End If
      Allocate (rmean(nsummaries))

      If (allocated(rsd)) Then
        Deallocate (rsd)
      End If
      Allocate (rsd(lrsd))

!      Calculate summary statistics for this block of data
      ifail = 0
      Call g01waf(m,nb,x,iwt,wt,pn,rmean,rsd,lrsd,rcomm,lrcomm,ifail)

!      Number of results printed so far
      offset = max(0,pn-nb-m+1)

!      Display the results for this block of data
      If (want_sd) Then
        Do i = 1, nsummaries
          Write (nout,99998) '[', i + offset, ',', i + m - 1 + offset, ']', &
            rmean(i), rsd(i)
        End Do
      Else
        Do i = 1, nsummaries
          Write (nout,99998) '[', i + offset, ',', i + m - 1 + offset, ']', &
            rmean(i)
        End Do
      End If
    End Do blk_lp

    Write (nout,*)
    Write (nout,99999) 'Total number of observations : ', pn
    Write (nout,99999) 'Length of window : ', m

99999 Format (1X,A,I5)
99998 Format (1X,A,2(I3,A),2(4X,F10.1))
99997 Format (1X,A)
      End Program g01wafe

```

## 10.2 Program Data

G01WAF Example Program Data

2 15	:: IWT,M
FALSE	:: If TRUE sd's are calculated
-3.0 -6.0 -5.0 3.0 21.0 46.0 67.0	
74.0 67.0 46.0 21.0 3.0 -5.0 -6.0 -3.0	:: WT
5	:: NB
-2170.0 -1770.0 -1660.0 -1360.0 -1100.0	:: End of X for first block
10	:: NB
-950.0 -640.0 -370.0 -140.0 -250.0	



```

-510.0 -620.0 -730.0 -880.0 -1130.0 :: End of X for second block
15                                     :: NB
-1200.0 -830.0 -330.0 -190.0 210.0
 170.0  440.0  440.0  780.0  880.0
1220.0 1260.0 1140.0  850.0  640.0 :: End of X for third block

```

### 10.3 Program Results

G01WAF Example Program Results

Interval	Mean
[ 1, 15]	-427.6
[ 2, 16]	-332.5
[ 3, 17]	-337.1
[ 4, 18]	-438.2
[ 5, 19]	-604.4
[ 6, 20]	-789.4
[ 7, 21]	-935.4
[ 8, 22]	-990.6
[ 9, 23]	-927.1
[10, 24]	-752.1
[11, 25]	-501.3
[12, 26]	-227.2
[13, 27]	23.2
[14, 28]	236.2
[15, 29]	422.4
[16, 30]	604.2

```

Total number of observations : 30
Length of window           : 15

```

This example plot shows the smoothing effect of using different length rolling windows on the mean and standard deviation. Two different window lengths,  $m = 5$  and  $10$ , are used to produce the unweighted rolling mean and standard deviations for the change in rate of the Earth's rotation between 1821 and 1850. The values of the rolling mean and standard deviations are plotted at the centre points of their respective windows.

