# NAG Library Routine Document

# G08CHF

**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

G08CHF calculates the Anderson–Darling goodness-of-fit test statistic.

## 2 Specification

```
FUNCTION G08CHF (N, ISSORT, Y, IFAIL)
REAL (KIND=nag_wp) G08CHF
INTEGER            N, IFAIL
REAL (KIND=nag_wp) Y(N)
LOGICAL            ISSORT
```

## 3 Description

Denote by $A^2$ the Anderson–Darling test statistic for $n$ observations $y_1, y_2, \ldots, y_n$ of a variable $Y$ assumed to be standard uniform and sorted in ascending order, then:

$$A^2 = -n - S;$$

where:

$$S = \sum_{i=1}^{n} \frac{2i-1}{n} [\ln y_i + \ln(1 - y_{n-i+1})].$$

When observations of a random variable $X$ are non-uniformly distributed, the probability integral transformation (PIT):

$$Y = F(X),$$

where $F$ is the cumulative distribution function of the distribution of interest, yields a uniformly distributed random variable $Y$. The PIT is true only if all parameters of a distribution are known as opposed to estimated; otherwise it is an approximation.

## 4 References

Anderson T W and Darling D A (1952) Asymptotic theory of certain 'goodness-of-fit' criteria based on stochastic processes *Annals of Mathematical Statistics* **23** 193–212

## 5 Arguments

1: N – INTEGER *Input*

*On entry*: $n$, the number of observations.

*Constraint*: N $> 1$.

2: ISSORT – LOGICAL *Input*

*On entry*: set ISSORT = .TRUE. if the observations are sorted in ascending order; otherwise the function will sort the observations.

3: Y(N) – REAL (KIND=nag_wp) array *Input/Output*

*On entry*: $y_i$, for $i = 1, 2, \ldots, n$, the $n$ observations.

*On exit*: if ISSORT = .FALSE., the data sorted in ascending order; otherwise the array is unchanged.

*Constraint*: if ISSORT = .TRUE., the values must be sorted in ascending order. Each $y_i$ must lie in the interval $(0, 1)$.

4: 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, N $= \langle value \rangle$.
Constraint: N $> 1$.

IFAIL $= 3$

ISSORT = .TRUE. and the data in Y is not sorted in ascending order.

IFAIL $= 9$

The data in Y must lie in the interval $(0, 1)$.

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

G08CHF is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example calculates the $A^2$ statistic for data assumed to arise from an exponential distribution with a sample parameter estimate and simulates its $p$-value using the NAG basic random number generator.

### 10.1 Program Text

```
    Program g08chfe

!     G08CHF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05sff, g08chf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: genid = 1, lseed = 1, mstate = 17,   &
                                          nin = 5, nout = 6, subid = -1
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: a2, aa2, beta, nupper, p, sa2, sbeta
      Integer                          :: i, ifail, j, k, lstate, n, nsim,     &
                                          n_pseudo
      Logical                          :: issort
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: x(:), xsim(:), y(:)
      Integer                          :: seed(lseed), state(17)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: exp, real, sum
!     .. Executable Statements ..
      Write (nout,*) 'G08CHF Example Program Results'
      Write (nout,*)

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

!     Read number of observations
      Read (nin,*) n

!     Memory allocation
      Allocate (x(n),y(n))

!     Read observations
      Read (nin,*)(x(i),i=1,n)

!     Maximum likelihood estimate of mean
      beta = sum(x(1:n))/real(n,kind=nag_wp)

!     PIT, using exponential CDF with mean beta
      Do i = 1, n
        y(i) = 1.0E0_nag_wp - exp(-x(i)/beta)
      End Do

!     Let g08chf sort the (approximately) uniform variates
      issort = .False.

!     Calculate A-squared
      ifail = 0
      a2 = g08chf(n,issort,y,ifail)
```

```
      aa2 = (1.0E0_nag_wp+0.6E0_nag_wp/real(n,kind=nag_wp))*a2

!     Number of simulations
      nsim = 888

!     Generate exponential variates using a repeatable seed
      Allocate (xsim(n*nsim))
      seed(1) = 206033
      lstate = mstate
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)
      n_pseudo = n*nsim
      ifail = 0
      Call g05sff(n_pseudo,beta,state,xsim,ifail)

!     Simulations loop
      nupper = 0.0E0_nag_wp
      Do j = 1, nsim
        k = (j-1)*n
!       Maximum likelihood estimate of mean
        sbeta = sum(xsim(k+1:k+n))/real(n,kind=nag_wp)
!       PIT
        Do i = 1, n
          y(i) = 1.0E0_nag_wp - exp(-xsim(k+i)/sbeta)
        End Do
!       Calculate A-squared
        ifail = 0
        sa2 = g08chf(n,issort,y,ifail)
        If (sa2>aa2) Then
          nupper = nupper + 1.0E0_nag_wp
        End If
      End Do

!     Simulated upper tail probability value
      p = nupper/real(nsim+1,kind=nag_wp)

!     Results
      Write (nout,'(1X,A,E11.4)')                                          &
        'H0: data from exponential distribution with mean', beta
      Write (nout,'(1X,A,1X,F8.4)') 'Test statistic, A-squared: ', a2
      Write (nout,'(1X,A,1X,F8.4)') 'Upper tail probability:    ', p

    End Program g08chfe
```

## 10.2 Program Data

```
G08CHF Example Program Data
26 :: n
0.4782745 1.2858962 1.1163891 2.0410619 2.2648109 0.0833660 1.2527554
0.4031288 0.7808981 0.1977674 3.2539440 1.8113504 1.2279834 3.9178773
1.4494309 0.1358438 1.8061778 6.0441929 0.9671624 3.2035042 0.8067364
0.4179364 3.5351774 0.3975414 0.6120960 0.1332589 :: end of observations
```

## 10.3 Program Results

```
 G08CHF Example Program Results

 H0: data from exponential distribution with mean 0.1524E+01
 Test statistic, A-squared:    0.1616
 Upper tail probability:       0.9798
```