# NAG Toolbox

# nag_nonpar_gofstat_anddar (g08ch)

## 1 Purpose

nag_nonpar_gofstat_anddar (g08ch) calculates the Anderson–Darling goodness-of-fit test statistic.

## 2 Syntax

```
[result, y, ifail] = nag_nonpar_gofstat_anddar(issort, y, 'n', n)
[result, y, ifail] = g08ch(issort, y, 'n', n)
```

## 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 Parameters

### 5.1 Compulsory Input Parameters

1: **issort** – LOGICAL

Set **issort** = *true* if the observations are sorted in ascending order; otherwise the function will sort the observations.

2: **y**(**n**) – REAL (KIND=nag_wp) array

$y_i$, for $i = 1, 2, \ldots, n$, the $n$ observations.

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

### 5.2 Optional Input Parameters

1: **n** – INTEGER

*Default*: the dimension of the array **y**.

$n$, the number of observations.

*Constraint*: $\mathbf{n} > 1$.

### 5.3 Output Parameters

1: **result**

The result of the function.

2: **y**(**n**) – REAL (KIND=nag_wp) array

If **issort** $= false$, the data sorted in ascending order; otherwise the array is unchanged.

3: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

Constraint: $\mathbf{n} > 1$.

**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.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

Not applicable.

## 8 Further Comments

None.

## 9 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.

## 9.1 Program Text

```
     function g08ch_example

fprintf('g08ch example results\n\n');

x = [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];
n = nag_int(numel(x));

% Maximum likelihood estimate of mean
beta = mean(x);
% PIT, using exponential CDF with mean beta
y = 1 - exp(-x/beta);
% Let g08ch sort the (approximately) uniform variates
issort = false;

% Calculate a-squared
[a2, y, ifail] = g08ch( ...
                        issort, y);

aa2 = (1+0.6/numel(y))*a2;

% Number of simulations
nsim = nag_int(888);
% Initialize the base generator to a repeatable sequence
seed  = [nag_int(206033)];
genid = nag_int(1);
subid = nag_int(-1);
[state, ifail] = g05kf( ...
                        genid, subid, seed);
[state, xsim, ifail] = g05sf( ...
                              n*nsim, beta, state);

% Simulations loop
nupper = 0;
for j=1:nsim
  k = (j-1)*n;
  x = xsim(k+1:k+n);
  % Maximum likelihood estimate of mean
  sbeta = mean(x);
  % PIT
  y = 1 - exp(-x/sbeta);
  % Calculate a-squared
  [sa2, y, ifail] = g08ch( ...
                           issort, y);
  if sa2 > aa2
    nupper = nupper + 1;
  end
end

% Simulated upper tail probability value
p = nupper/(nsim+1);

% Results
fprintf('H0: data from exponential distribution with mean %10.4e\n', beta);
fprintf('Test statistic, A-squared: %8.4f\n', a2);
fprintf('Upper tail probability:    %8.4f\n', p);
```

## 9.2 Program Results

```
     g08ch example results

H0: data from exponential distribution with mean 1.5240e+00
Test statistic, A-squared:   0.1616
Upper tail probability:      1.0000
```