# NAG Library Function Document

# nag_gaps_test (g08edc)

## 1 Purpose

nag_gaps_test (g08edc) performs a gaps test on a sequence of observations.

## 2 Specification

```
#include <nag.h>
#include <nagg08.h>
void nag_gaps_test (Integer n, const double x[], Integer num_gaps,
     Integer max_gap, double lower, double upper, double length, double *chi,
     double *df, double *prob, NagError *fail)
```

## 3 Description

Gaps tests are used to test for cyclical trend in a sequence of observations. nag_gaps_test (g08edc) computes certain statistics for the gaps test.

The term gap is used to describe the distance between two numbers in the sequence that lie in the interval $(r_l, r_u)$. That is, a gap ends at $x_j$ if $r_l \leq x_j \leq r_u$. The next gap then begins at $x_{j+1}$. The interval $(r_l, r_u)$ should lie within the region of all possible numbers. For example if the test is carried out on a sequence of $(0, 1)$ random numbers then the interval $(r_l, r_u)$ must be contained in the whole interval $(0, 1)$. Let $t_{\text{len}}$ be the length of the interval which specifies all possible numbers.

nag_gaps_test (g08edc) counts the number of gaps of different lengths. Let $c_i$ denote the number of gaps of length $i$, for $i = 1, 2, \ldots, k-1$. The number of gaps of length $k$ or greater is then denoted by $c_k$. An unfinished gap at the end of a sequence is not counted. The following is a trivial example.

Suppose we called nag_gaps_test (g08edc) with the following sequence and with $r_l = 0.30$ and $r_u = 0.60$:

0.20 0.40 0.45 0.40 0.15 0.75 0.95 0.230.27 0.40 0.25 0.10 0.34 0.39 0.61 0.12.

nag_gaps_test (g08edc) would have counted the gaps of the following lengths:

2, 1, 1, 6, 3 and 1.

When the counting of gaps is complete nag_gaps_test (g08edc) computes the expected values of the counts. An approximate $\chi^2$ statistic with $k$ degrees of freedom is computed where

$$X^2 = \frac{\sum_{i=1}^{k}(c_i - e_i)^2}{e_i},$$

where

$e_i = ngaps \times p \times (1-p)^{i-1}$, if $i < k$;

$e_i = ngaps \times (1-p)^{i-1}$, if $i = k$;

$ngaps =$ the number of gaps found and

$p = (r_u - r_l)/t_{\text{len}}$.

The use of the $\chi^2$-distribution as an approximation to the exact distribution of the test statistic improves as the expected values increase.

You may specify the total number of gaps to be found. If the specified number of gaps is found before the end of a sequence nag_gaps_test (g08edc) will exit before counting any further gaps.

## 4 References

Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

Morgan B J T (1984) *Elements of Simulation* Chapman and Hall

Ripley B D (1987) *Stochastic Simulation* Wiley

## 5 Arguments

1: **n** – Integer *Input*

*On entry*: $n$, the length of the current sequence of observations.

*Constraint*: **n** $\geq 1$.

2: **x**[**n**] – const double *Input*

*On entry*: the sequence of observations.

3: **num_gaps** – Integer *Input*

*On entry*: the maximum number of gaps to be sought. If **num_gaps** $\leq 0$ then there is no limit placed on the number of gaps that are found.

*Constraint*: **num_gaps** $\leq$ **n**.

4: **max_gap** – Integer *Input*

*On entry*: $k$, the length of the longest gap for which tabulation is desired.

*Constraint*: $1 <$ **max_gap** $\leq$ **n**.

5: **lower** – double *Input*

*On entry*: the lower limit of the interval to be used to define the gaps, $r_l$.

6: **upper** – double *Input*

*On entry*: the upper limit of the interval to be used to define the gaps, $r_u$.

*Constraint*: **upper** $>$ **lower**.

7: **length** – double *Input*

*On entry*: the total length of the interval which contains all possible numbers that may arise in the sequence.

*Constraint*: **length** $> 0.0$ and **upper** $-$ **lower** $<$ **length**.

8: **chi** – double * *Output*

*On exit*: contains the $\chi^2$ test statistic, $X^2$, for testing the null hypothesis of randomness.

9: **df** – double * *Output*

*On exit*: contains the degrees of freedom for the $\chi^2$ statistic.

10: **prob** – double * *Output*

*On exit*: contains the upper tail probability associated with the $\chi^2$ test statistic, i.e., the significance level.

11: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_2_INT_ARG_GT**

On entry, **num_gaps** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **num_gaps** ≤ **n**.

**NE_2_REAL_ARG_GE**

On entry, **lower** = ⟨*value*⟩ and **upper** = ⟨*value*⟩.
Constraint: **upper** > **lower**.

**NE_3_REAL_ARG_CONS**

On entry, **lower** = ⟨*value*⟩, **upper** = ⟨*value*⟩ and **length** = ⟨*value*⟩.
Constraint: **upper** − **lower** < **length**.

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_G08ED_FREQ_LT_ONE**

The expected frequency of at least one class is less than one.
This implies that the $\chi^2$ may not be a very good approximation to the distribution of the test statistics.
All statistics are returned and may still be of use.

**NE_G08ED_FREQ_ZERO**

The expected frequency in class $i$ = ⟨*value*⟩ is zero. The value of (**upper** − **lower**)/**length** may be too close to 0.0 or 1.0. or **max_gap** is too large relative to the number of gaps found.

**NE_G08ED_GAPS**

The number of gaps requested were not found, only ⟨*value*⟩ out of the requested ⟨*value*⟩ where found.
All statistics are returned and may still be of use.

**NE_G08ED_GAPS_ZERO**

No gaps were found. Try using a longer sequence, or increase the size of the interval **upper** − **lower**.

**NE_INT_2**

On entry, **max_gap** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: 1 < **max_gap** ≤ **n**.

On entry, **max_gap** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: 1 ≤ **max_gap** ≤ **n**.

**NE_INT_ARG_LT**

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 1$.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL_ARG_LE**

On entry, **length** $= \langle value \rangle$.
Constraint: **length** $> 0.0$.

## 7 Accuracy

The computations are believed to be stable. The computation of **prob** given the values of **chi** and **df** will obtain a relative accuracy of five significant places for most cases.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

The time taken by nag_gaps_test (g08edc) increases with the number of observations $n$.

## 10 Example

The following program performs the gaps test on 5000 pseudorandom numbers taken from a uniform distribution $U(0, 1)$, generated by nag_rand_uniform (g05sqc). All gaps of length 10 or more are counted together.

### 10.1 Program Text

```
/* nag_gaps_test (g08edc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 *
 * Mark 8 revised, 2004
 *
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>
#include <nagg08.h>

int main(void)
{
  /* Integer scalar and array declarations */
  Integer     exit_status = 0;
```

```
  Integer     lstate;
  Integer     *state = 0;

  /* NAG structures */
  NagError    fail;

  /* Double scalar and array declarations */
  double      chi, df, length, lower, p, upper, *x = 0;

  /* Choose the base generator */
  Nag_BaseRNG genid = Nag_Basic;
  Integer     subid = 0;

  /* Set the seed */
  Integer     seed[] = { 424232 };
  Integer     lseed = 1;

  /* Set the size of the (randomly generated) dataset */
  Integer     n = 5000;

  /* Set the maximum number of gaps (0 = no limit) */
  Integer     num_gaps = 0;

  /* Set the length of the maximum gap */
  Integer     max_gap = 10;

  /* Initialise the error structure */
  INIT_FAIL(fail);

  printf("nag_gaps_test (g08edc) Example Program Results\n");

  /* Get the length of the state array */
  lstate = -1;
  nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Allocate arrays */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(state = NAG_ALLOC(lstate, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise the generator to a repeatable sequence */
  nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Generate vector of n uniform variates between 0.0 and 1.0 */
  nag_rand_uniform(n, 0.0, 1.0, state, x, &fail);

  /* Set the length of interval which contains all possible values.
     The data is generated from the range 0.0 to 1.0, so length is 1.0
   */
  length = 1.0;

  /* Set lower and upper limit for the interval used for the gap test */
  lower = 0.4;
```

```
  upper = 0.6;

  /* nag_gaps_test (g08edc).
   * Performs the gaps test for randomness
   */
  nag_gaps_test(n, x, num_gaps, max_gap, lower, upper, length, &chi, &df, &p,
                &fail);

  /* Display the results */
  if (fail.code != NE_NOERROR && fail.code != NE_G08ED_GAPS &&
      NE_G08ED_FREQ_LT_ONE)
    {
      printf("Error from nag_gaps_test (g08edc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  printf("\n");
  printf("Chisq = %10.4f\n", chi);
  printf("DF    = %7.1f\n", df);
  printf("Prob  = %10.4f\n", p);
  if (fail.code == NE_G08ED_FREQ_LT_ONE)
    printf("Error from nag_gaps_test (g08edc).\n%s\n", fail.message);

 END:
  NAG_FREE(x);
  NAG_FREE(state);

  return exit_status;
}
```

## 10.2 Program Data

None.

## 10.3 Program Results

```
nag_gaps_test (g08edc) Example Program Results

Chisq =     7.0401
DF    =     9.0
Prob  =     0.6329
```