# NAG Library Function Document

# nag_2_sample_ks_test (g08cdc)

## 1 Purpose

nag_2_sample_ks_test (g08cdc) performs the two sample Kolmogorov–Smirnov distribution test.

## 2 Specification

```
#include <nag.h>
#include <nagg08.h>
void nag_2_sample_ks_test (Integer n1, const double x[], Integer n2,
     const double y[], Nag_TestStatistics dtype, double *d, double *z,
     double *p, NagError *fail)
```

## 3 Description

The data consist of two independent samples, one of size $n_1$, denoted by $x_1, x_2, \ldots, x_{n_1}$, and the other of size $n_2$ denoted by $y_1, y_2, \ldots, y_{n_2}$. Let $F(x)$ and $G(x)$ represent their respective, unknown, distribution functions. Also let $S_1(x)$ and $S_2(x)$ denote the values of the sample cumulative distribution functions at the point $x$ for the two samples respectively.

The Kolmogorov–Smirnov test provides a test of the null hypothesis $H_0 : F(x) = G(x)$ against one of the following alternative hypotheses:

(i) $H_1 : F(x) \neq G(x)$.

(ii) $H_2 : F(x) > G(x)$. This alternative hypothesis is sometimes stated as, 'The $x$'s tend to be smaller than the $y$'s', i.e., it would be demonstrated in practical terms if the values of $S_1(x)$ tended to exceed the corresponding values of $S_2(x)$.

(iii) $H_3 : F(x) < G(x)$. This alternative hypothesis is sometimes stated as, 'The $x$'s tend to be larger than the $y$'s', i.e., it would be demonstrated in practical terms if the values of $S_2(x)$ tended to exceed the corresponding values of $S_1(x)$.

One of the following test statistics is computed depending on the particular alternative null hypothesis specified (see the description of the argument **dtype** in Section 5).

For the alternative hypothesis $H_1$.

$D_{n_1,n_2}$ – the largest absolute deviation between the two sample cumulative distribution functions.

For the alternative hypothesis $H_2$.

$D^+_{n_1,n_2}$ – the largest positive deviation between the sample cumulative distribution function of the first sample, $S_1(x)$, and the sample cumulative distribution function of the second sample, $S_2(x)$. Formally $D^+_{n_1,n_2} = \max\{S_1(x) - S_2(x), 0\}$.

For the alternative hypothesis $H_3$.

$D^-_{n_1,n_2}$ – the largest positive deviation between the sample cumulative distribution function of the second sample, $S_2(x)$, and the sample cumulative distribution function of the first sample, $S_1(x)$. Formally $D^-_{n_1,n_2} = \max\{S_2(x) - S_1(x), 0\}$.

nag_2_sample_ks_test (g08cdc) also returns the standardized statistic $Z = \sqrt{(n_1 + n_2/n_1 n_2)} \times D$ where $D$ may be $D_{n_1,n_2}$, $D^+_{n_1,n_2}$ or $D^-_{n_1,n_2}$ depending on the choice of the alternative hypothesis. The distribution of this statistic converges asymptotically to a distribution given by Smirnov as $n_1$ and $n_2$ increase (see Feller (1948), Kendall and Stuart (1973), Kim and Jenrich (1973), Smirnov (1933) or Smirnov (1948)).

The probability, under the null hypothesis, of obtaining a value of the test statistic as extreme as that observed, is computed. If $\max(n_1, n_2) \leq 2500$ and $n_1 n_2 \leq 10000$ then an exact method given by Kim and Jenrich is used. Otherwise $p$ is computed using the approximations suggested by Kim and Jenrich (see Kim and Jenrich (1973)). Note that the method used is only exact for continuous theoretical distributions. This method computes the two-sided probability. The one-sided probabilities are estimated by halving the two-sided probability. This is a good estimate for small $p$, that is $p \leq 0.10$, but it becomes very poor for larger $p$.

## 4 References

Conover W J (1980) *Practical Nonparametric Statistics* Wiley

Feller W (1948) On the Kolmogorov–Smirnov limit theorems for empirical distributions *Ann. Math. Statist.* **19** 179–181

Kendall M G and Stuart A (1973) *The Advanced Theory of Statistics (Volume 2)* (3rd Edition) Griffin

Kim P J and Jenrich R I (1973) Tables of exact sampling distribution of the two sample Kolmogorov–Smirnov criterion $D_{mn}(m < n)$ *Selected Tables in Mathematical Statistics* **1** 80–129 American Mathematical Society

Siegel S (1956) *Non-parametric Statistics for the Behavioral Sciences* McGraw–Hill

Smirnov N (1933) Estimate of deviation between empirical distribution functions in two independent samples *Bull. Moscow Univ.* **2(2)** 3–16

Smirnov N (1948) Table for estimating the goodness of fit of empirical distributions *Ann. Math. Statist.* **19** 279–281

## 5 Arguments

1: **n1** – Integer *Input*

*On entry*: the number of observations in the first sample, $n_1$.

*Constraint*: **n1** $\geq 1$.

2: **x**[**n1**] – const double *Input*

*On entry*: the observations from the first sample, $x_1, x_2, \ldots, x_{n_1}$.

3: **n2** – Integer *Input*

*On entry*: the number of observations in the second sample, $n_2$.

*Constraint*: **n2** $\geq 1$.

4: **y**[**n2**] – const double *Input*

*On entry*: the observations from the second sample, $y_1, y_2, \ldots, y_{n_2}$.

5: **dtype** – Nag_TestStatistics *Input*

*On entry*: the statistic to be computed, i.e., the choice of alternative hypothesis.

**dtype** = Nag_TestStatisticsDAbs
Computes $D_{n_1 n_2}$, to test against $H_1$.

**dtype** = Nag_TestStatisticsDPos
Computes $D^+_{n_1 n_2}$, to test against $H_2$.

**dtype** = Nag_TestStatisticsDNeg
Computes $D^-_{n_1 n_2}$, to test against $H_3$.

*Constraint*: **dtype** = Nag_TestStatisticsDAbs, Nag_TestStatisticsDPos or Nag_TestStatisticsDNeg.

6: **d** – double * *Output*

*On exit*: the Kolmogorov–Smirnov test statistic ($D_{n_1n_2}$, $D^+_{n_1n_2}$ or $D^-_{n_1n_2}$ according to the value of **dtype**).

7: **z** – double * *Output*

*On exit*: a standardized value, $Z$, of the test statistic, $D$, without any correction for continuity.

8: **p** – double * *Output*

*On exit*: the tail probability associated with the observed value of $D$, where $D$ may be $D_{n_1,n_2}$, $D^+_{n_1,n_2}$ or $D^-_{n_1,n_2}$ depending on the value of **dtype** (see Section 3).

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument **dtype** had an illegal value.

**NE_G08CD_CONV**

The iterative procedure used in the approximation of the probability for large **n1** and **n2** did not converge. For the two-sided test, $\mathbf{p} = 1.0$ is returned. For the one-sided test, $\mathbf{p} = 0.5$ is returned.

**NE_INT_ARG_LT**

On entry, **n1** must not be less than 1: $\mathbf{n1} = \langle value \rangle$.

On entry, **n2** must not be less than 1: $\mathbf{n2} = \langle value \rangle$.

**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.

## 7 Accuracy

The large sample distributions used as approximations to the exact distribution should have a relative error of less than 5% for most cases.

## 8 Parallelism and Performance

nag_2_sample_ks_test (g08cdc) is not threaded in any implementation.

## 9 Further Comments

The time taken by nag_2_sample_ks_test (g08cdc) increases with $n_1$ and $n_2$, until $n_1n_2 > 10000$ or $\max(n_1, n_2) \geq 2500$. At this point one of the approximations is used and the time decreases significantly. The time then increases again modestly with $n_1$ and $n_2$.

# 10 Example

The following example computes the two-sided Kolmogorov–Smirnov test statistic for two independent samples of size 100 and 50 respectively. The first sample is from a uniform distribution $U(0,2)$. The second sample is from a uniform distribution $U(0.25,2.25)$. The test statistic, $D_{n_1,n_2}$, the standardized test statistic, $Z$, and the tail probability, $p$, are computed and printed.

## 10.1 Program Text

```
/* nag_2_sample_ks_test (g08cdc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 *
 */

#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 m, n, lstate;
  Integer *state = 0;

  /* Double scalar and array declarations */
  double d, p, z;
  double *x = 0, *y = 0;

  /* Character array declarations */
  char nag_enum_arg[40];

  /* NAG structures and data types */
  Nag_TestStatistics ntype;
  NagError fail;

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

  /* Set the seed */
  Integer seed[] = { 423523 };
  Integer lseed = 1;

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

  printf("nag_2_sample_ks_test (g08cdc) 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;
  }

  /* Skip heading in data file */
#ifdef _WIN32
```

```
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Get the problem size */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT " %" NAG_IFMT "", &n, &m);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT "", &n, &m);
#endif

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

  /* Initialize 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 and 2 */
  nag_rand_uniform(n, 0.0, 2.0, state, x, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_uniform (g05sqc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Generate vector of m uniform variates between 0.25 and 2.25 */
  nag_rand_uniform(m, 0.25, 2.25, state, y, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_uniform (g05sqc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  ntype = (Nag_TestStatistics) nag_enum_name_to_value(nag_enum_arg);

  /* nag_2_sample_ks_test (g08cdc).
   * Performs the two-sample Kolmogorov-Smirnov test
   */
  nag_2_sample_ks_test(n, x, m, y, ntype, &d, &z, &p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_2_sample_ks_test (g08cdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  printf("Test statistic D = %8.4f\n", d);
  printf("Z statistic      = %8.4f\n", z);
  printf("Tail probability = %8.4f\n", p);

END:
```

```
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(state);

  return exit_status;
}
```

## 10.2 Program Data

```
nag_2_sample_ks_test (g08cdc) Example Program Data
 100 50
 Nag_TestStatisticsDAbs
```

## 10.3 Program Results

```
nag_2_sample_ks_test (g08cdc) Example Program Results

Test statistic D =   0.1800
Z statistic      =   0.0312
Tail probability =   0.2222
```