NAG Library Function Document

nag_sum_convcorr_real (c06fkc)

1 Purpose

nag_sum_convcorr_real (c06fkc) calculates the circular convolution or correlation of two real vectors of period n.

2 Specification

```
#include <nag.h>
#include <nagc06.h>
```

```
void nag_sum_convcorr_real (Nag_VectorOp job, double x[], double y[],
     Integer n, NagError *fail)
```

3 Description

nag_sum_convcorr_real (c06fkc) computes:

if **job** $=$ Nag_Convolution, the discrete **convolution** of x and y, defined by

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j} = \sum_{j=0}^{n-1} x_{k-j} y_j;$$

if **job** $=$ Nag_Correlation, the discrete **correlation** of x and y defined by

$$w_k = \sum_{j=0}^{n-1} x_j y_{k+j}.$$

Here x and y are real vectors, assumed to be periodic, with period n, i.e., $x_j = x_{j \pm n} = x_{j \pm 2n} = \ldots$; z and w are then also periodic with period n.

Note: this usage of the terms 'convolution' and 'correlation' is taken from Brigham (1974). The term 'convolution' is sometimes used to denote both these computations.

If $\hat{x}$, $\hat{y}$, $\hat{z}$ and $\hat{w}$ are the discrete Fourier transforms of these sequences, i.e.,

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \times \exp\left(-i\frac{2\pi jk}{n}\right), \text{ etc.,}$$

then $\hat{z}_k = \sqrt{n}.\hat{x}_k \hat{y}_k$ and $\hat{w}_k = \sqrt{n}.\bar{\hat{x}}_k \hat{y}_k$ (the bar denoting complex conjugate).

This function calls the same auxiliary functions as nag_sum_fft_realherm_1d (c06pac) to compute discrete Fourier transforms.

4 References

Brigham E O (1974) *The Fast Fourier Transform* Prentice–Hall

5 Arguments

1: **job** – Nag_VectorOp *Input*

On entry: the computation to be performed.

job = Nag_Convolution

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j};$$

job = Nag_Correlation

$$w_k = \sum_{j=0}^{n-1} x_j y_{k+j}.$$

Constraint: **job** = Nag_Convolution or Nag_Correlation.

2: **x**[**n**] – double *Input/Output*

On entry: the elements of one period of the vector x. **x**[j] must contain x_j, for $j = 0, 1, \ldots, n-1$.

On exit: the corresponding elements of the discrete convolution or correlation.

3: **y**[**n**] – double *Input/Output*

On entry: the elements of one period of the vector y. **y**[j] must contain y_j, for $j = 0, 1, \ldots, n-1$.

On exit: the discrete Fourier transform of the convolution or correlation returned in the array **x**; the transform is stored in Hermitian form; if the components of the transform z_k are written as $a_k + ib_k$, then for $0 \leq k \leq n/2$, a_k is contained in **y**[k], and for $1 \leq k \leq n/2 - 1$, b_k is contained in **y**[$n-k$]. (See also Section 2.1.2 in the c06 Chapter Introduction.)

4: **n** – Integer *Input*

On entry: n, the number of values in one period of the vectors **x** and **y**.

Constraint: **n** > 1.

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

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

6 Error Indicators and Warnings

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.

⟨*value*⟩ is an invalid value of **job**.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** > 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.

7 Accuracy

The results should be accurate to within a small multiple of the ***machine precision***.

8 Parallelism and Performance

nag_sum_convcorr_real (c06fkc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_sum_convcorr_real (c06fkc) 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 function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken is approximately proportional to $n \times \log(n)$, but also depends on the factorization of n. nag_sum_convcorr_real (c06fkc) is faster if the only prime factors of n are 2, 3 or 5; and fastest of all if n is a power of 2.

10 Example

This example reads in the elements of one period of two real vectors x and y, and prints their discrete convolution and correlation (as computed by nag_sum_convcorr_real (c06fkc)). In realistic computations the number of data values would be much larger.

10.1 Program Text

```
/* nag_sum_convcorr_real (c06fkc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagc06.h>

int main(void)
{
  /* Scalars */
  Integer   exit_status = 0, j, n;
  /* Arrays */
  double    *xa = 0, *xb = 0, *ya = 0, *yb = 0;
  /* Nag Types */
  NagError  fail;

  INIT_FAIL(fail);

  printf("nag_sum_convcorr_real (c06fkc) Example Program Results\n");
```

```
#ifdef _WIN32
  scanf_s("%*[^\n]""%"NAG_IFMT"%*[^\n]", &n);
#else
  scanf("%*[^\n]""%"NAG_IFMT"%*[^\n]", &n);
#endif
  if (n<2)
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }

  if (!(xa = NAG_ALLOC(n, double)) ||
      !(xb = NAG_ALLOC(n, double)) ||
      !(ya = NAG_ALLOC(n, double)) ||
      !(yb = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (j = 0; j < n; ++j)
    {
#ifdef _WIN32
      scanf_s("%lf%lf", &xa[j], &ya[j]);
#else
      scanf("%lf%lf", &xa[j], &ya[j]);
#endif
      xb[j] = xa[j];
      yb[j] = ya[j];
    }

  /* nag_sum_convcorr_real (c06fkc).
   * Circular convolution or correlation of two real vectors
   */
  nag_sum_convcorr_real(Nag_Convolution, xa, ya, n, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_sum_convcorr_real (c06fkc).\n%s\n",
             fail.message);
      exit_status = 2;
      goto END;
    }
  /* nag_sum_convcorr_real (c06fkc), see above. */
  nag_sum_convcorr_real(Nag_Correlation, xb, yb, n, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_sum_convcorr_real (c06fkc).\n%s\n",
             fail.message);
      exit_status = 3;
      goto END;
    }

  printf("\n          Convolution   Correlation\n\n");
  for (j = 0; j < n; ++j)
    printf("%5"NAG_IFMT" %13.5f %13.5f\n", j, xa[j], xb[j]);

 END:
  NAG_FREE(xa);
  NAG_FREE(xb);
  NAG_FREE(ya);
  NAG_FREE(yb);

  return exit_status;
}
```

10.2 Program Data

```
nag_sum_convcorr_real (c06fkc) Example Program Data
    9                  : n
      1.00      0.50
      1.00      0.50
      1.00      0.50
      1.00      0.50
      1.00      0.00
      0.00      0.00
      0.00      0.00
      0.00      0.00
      0.00      0.00    : xa, ya
```

10.3 Program Results

```
nag_sum_convcorr_real (c06fkc) Example Program Results

          Convolution   Correlation

    0       0.50000       2.00000
    1       1.00000       1.50000
    2       1.50000       1.00000
    3       2.00000       0.50000
    4       2.00000       0.00000
    5       1.50000       0.50000
    6       1.00000       1.00000
    7       0.50000       1.50000
    8       0.00000       2.00000
```