Module 7.3: nag_conv
Convolution and Correlation

`nag_conv` provides a procedure for computing the *convolution* or *correlation* of two real or complex vectors.

Contents

Introduction

1 Convolutions and Correlations

The discrete *convolution* of two real or complex vectors $x = (x_0, x_1, \ldots, x_{n-1})^T$ and $y = (y_0, y_1, \ldots, y_{n-1})^T$ is defined by:

$$z_k = \sum_{j=0}^{n-1} x_j y_{k-j}, \qquad k = 0, 1, \ldots, n-1 \tag{1}$$

where the sequences x_j and y_j are assumed to be periodic with period n, i.e.,:

$$\left.\begin{array}{ccccccccccc} x_j & = & x_{j+n} & = & x_{j-n} & = & x_{j+2n} & = & x_{j-2n} & = & \cdots \\ y_j & = & y_{j+n} & = & y_{j-n} & = & y_{j+2n} & = & y_{j-2n} & = & \cdots \end{array}\right\} \text{ for } j = 0, 1, \ldots, n-1.$$

The sequence z_k is itself periodic with period n and can therefore be represented by a vector $z = (z_0, z_1, \ldots, z_{n-1})^T$.

Similarly, the *correlation* of x and y is:

$$w_k = \sum_{j=0}^{n-1} \bar{x}_j y_{k+j}, \qquad k = 0, 1, \ldots, n-1 \tag{2}$$

and can be represented by the vector $w = (w_0, w_1, \ldots, w_{n-1})^T$.

If $\hat{x}, \hat{y}, \hat{z}$ and $\hat{w}$ are the vectors representing the discrete Fourier transforms of these sequences then:

$$\hat{z}_k = \hat{x}_k \hat{y}_k \tag{3}$$

and

$$\hat{w}_k = \bar{\hat{x}}_k \hat{y}_k. \tag{4}$$

Under certain circumstances (see Brigham [1]) these *discrete* convolutions and correlations can be used as approximations to the convolution or correlation integrals defined by:

$$z(s) = \int_{-\infty}^{\infty} x(t) y(s-t) dt \tag{5}$$

and

$$w(s) = \int_{-\infty}^{\infty} \bar{x}(t) y(s+t) dt, \quad -\infty < s < \infty. \tag{6}$$

2 Trigonometric Coefficients

In order to calculate the convolution or correlation, `nag_fft_conv` uses the discrete Fourier transform (DFT). Computing a DFT involves computation of a number of trigonometric coefficients, which can take a significant proportion of the total CPU-time. `nag_fft_conv` can either compute the trigonometric coefficients internally (in which case they are recomputed at each call) or it can allow the coefficients to be pre-computed by the procedure `nag_fft_trig` from the module `nag_fft` (7.1), and supplied in an optional argument (which is more efficient if several calls are made for vectors of the same length).

If you are using `nag_fft_conv`, and wish to pre-compute the trigonometric coefficients by calling `nag_fft_trig`, it is not necessary to add a separate `USE` statement for the module `nag_fft` (7.1). The procedure `nag_fft_trig` is also available through the `USE` statement for this module.

Procedure: nag_fft_conv

1 Description

`nag_fft_conv` uses the discrete Fourier transform to return the circular convolution or correlation of two real or complex vectors $x = (x_0, x_1, \ldots, x_{n-1})^T$ and $y = (y_0, y_1, \ldots, y_{n-1})^T$, representing periodic sequences of period n.

2 Usage

```
USE nag_conv
[value =] nag_fft_conv(x, y  [, optional arguments])
```

The function result is an array of the same shape and type as that of `x` and `y`.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array `x` must have exactly n elements.

This procedure derives the value of the following problem parameter from the shape of the supplied arrays.

$n \geq 1$ — the length of the vectors x and y

3.1 Mandatory Argument

x(n) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

y(n) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the vectors x and y to be convolved or correlated. If `x` and `y` are declared with bounds $(0 : n-1)$, then `x`(i) must contain x_i, and `y`(i) must contain y_i.

Constraints: `x` and `y` must be of the same type.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

correl — logical, intent(in), optional

Input: specifies whether the convolution or correlation is to be calculated.

If `correl` = `.true.`, then the correlation is calculated;

if `correl` = `.false.`, then the convolution is calculated.

Default: `correl` = `.false.`.

trig($2n$) — real(kind=*wp*), intent(in), optional

Input: trigonometric coefficients required for the computation of transforms of length n.

Default: if `trig` is not present, the coefficients are computed internally.

Constraints: `trig` must have been set by a prior call to the procedure `nag_fft_trig` from the module `nag_fft` (7.1). This procedure is accessible through the `USE` statement for this module.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.
320	The procedure was unable to allocate enough memory.

5 Examples of Usage

Complete examples of the use of this procedure appear in Examples 1 and 2 of this module document.

6 Further Comments

6.1 Algorithmic Detail

The procedure uses a variant of the fast Fourier transform (FFT) algorithm (see Brigham [1]) known as the Stockham self-sorting algorithm, which is described in Temperton [3] and Temperton [2]. Special coding is provided for the cases where n has factors 2, 3, 4, 5, or 6.

6.2 Accuracy

The results should be accurate to within a small multiple of `EPSILON(1.0_`*wp*`)`.

6.3 Timing

The time taken by the procedure is approximately proportional to $n \log n$, but also depends on the factors of n. The procedure is somewhat faster than average if the only prime factors of n are 2, 3, and 5; and fastest of all if n is a power of 2.

If several calls are made to this procedure to compute transforms of the same length n, supplying the trigonometric coefficients through the optional argument `trig` results in a saving of CPU-time.

Example 1: Convolution and Correlation of Real Vectors

This program calculates the convolution and correlation for two real vectors.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_conv_ex01

  ! Example Program Text for nag_conv
  ! NAG fl90, Release 4. NAG Copyright 2000.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  USE nag_conv, ONLY : nag_fft_conv
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, n
  ! .. Local Arrays ..
  REAL (wp), ALLOCATABLE :: result(:), x(:), y(:)
  ! .. Executable Statements ..
  WRITE (nag_std_out,*) 'Example Program Results for nag_conv_ex01'

  READ (nag_std_in,*)          ! Skip heading in data file
  READ (nag_std_in,*) n

  ALLOCATE (x(n),y(n),result(n)) ! Allocate storage

  READ (nag_std_in,*) (x(i),y(i),i=1,n)

  result = nag_fft_conv(x,y)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) 'Convolution'
  WRITE (nag_std_out,'(5f10.4)') result

  result = nag_fft_conv(x,y,correl=.TRUE.)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) 'Correlation'
  WRITE (nag_std_out,'(5f10.4)') result

  DEALLOCATE (x,y,result)      ! Deallocate storage

END PROGRAM nag_conv_ex01
```

2 Program Data

```
Example Program Data for nag_conv_ex01
   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  : (x(i),y(i),i=1,n)
```

3 Program Results

```
 Example Program Results for nag_conv_ex01

 Convolution
    0.5000    1.0000    1.5000    2.0000    2.0000
    1.5000    1.0000    0.5000    0.0000

 Correlation
    2.0000    1.5000    1.0000    0.5000    0.0000
    0.5000    1.0000    1.5000    2.0000
```

Example 2: Convolution and Correlation of Complex Vectors

This program calculates the convolution and correlation for two complex vectors.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_conv_ex02

  ! Example Program Text for nag_conv
  ! NAG fl90, Release 4. NAG Copyright 2000.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  USE nag_conv, ONLY : nag_fft_conv
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, n
  ! .. Local Arrays ..
  COMPLEX (wp), ALLOCATABLE :: result(:), x(:), y(:)
  ! .. Executable Statements ..
  WRITE (nag_std_out,*) 'Example Program Results for nag_conv_ex02'

  READ (nag_std_in,*)          ! Skip heading in data file
  READ (nag_std_in,*) n

  ALLOCATE (x(n),y(n),result(n)) ! Allocate storage

  READ (nag_std_in,*) (x(i),y(i),i=1,n)

  result = nag_fft_conv(x,y)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) 'Convolution'
  WRITE (nag_std_out,'(3(2X,''('',F9.4,'','',F9.4,'')''))') result

  result = nag_fft_conv(x,y,correl=.TRUE.)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) 'Correlation'
  WRITE (nag_std_out,'(3(2X,''('',F9.4,'','',F9.4,'')''))') result

  DEALLOCATE (x,y,result)      ! Deallocate storage

END PROGRAM nag_conv_ex02
```

2 Program Data

```
Example Program Data for nag_conv_ex02
    9                    : n
 (1.00, 2.0)   (0.50, 2.0)
 (1.00, 2.0)   (0.50, 2.0)
 (1.00, 2.0)   (0.50, 2.0)
 (1.00, 2.0)   (0.50, 2.0)
 (1.00, 2.0)   (0.00, 0.0)
 (0.00, 0.0)   (0.00, 0.0)
 (0.00, 0.0)   (0.00, 0.0)
 (0.00, 0.0)   (0.00, 0.0)
 (0.00, 0.0)   (0.00, 0.0) : (x(i),y(i),i=1,n)
```

3 Program Results

```
 Example Program Results for nag_conv_ex02

 Convolution
  (  -3.5000,   3.0000)  (  -7.0000,   6.0000)  ( -10.5000,   9.0000)
  ( -14.0000,  12.0000)  ( -14.0000,  12.0000)  ( -10.5000,   9.0000)
  (  -7.0000,   6.0000)  (  -3.5000,   3.0000)  (  -0.0000,   0.0000)

 Correlation
  (  18.0000,   4.0000)  (  13.5000,   3.0000)  (   9.0000,   2.0000)
  (   4.5000,   1.0000)  (   0.0000,   0.0000)  (   4.5000,   1.0000)
  (   9.0000,   2.0000)  (  13.5000,   3.0000)  (  18.0000,   4.0000)
```

References

[1] Brigham E O (1973) *The Fast Fourier Transform* Prentice-Hall

[2] Temperton C (1983) Self-sorting mixed-radix fast fourier transforms *J. Comput. Phys.* **52** 1–23

[3] Temperton C (1983) Fast mixed-radix real Fourier transforms *J. Comput. Phys.* **52** 340–350