

# NAG Library Routine Document

## C06GSF

**Note:** before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details.

### 1 Purpose

C06GSF takes  $m$  Hermitian sequences, each containing  $n$  data values, and forms the real and imaginary parts of the  $m$  corresponding complex sequences.

### 2 Specification

```
SUBROUTINE C06GSF (M, N, X, U, V, IFAIL)
  INTEGER          M, N, IFAIL
  REAL (KIND=nag_wp) X(M*N), U(M*N), V(M*N)
```

### 3 Description

This is a utility routine for use in conjunction with C06FPF and C06FQF (see the C06 Chapter Introduction).

### 4 References

None.

### 5 Parameters

- 1: M – INTEGER *Input*  
*On entry:*  $m$ , the number of Hermitian sequences to be converted into complex form.  
*Constraint:*  $M \geq 1$ .
- 2: N – INTEGER *Input*  
*On entry:*  $n$ , the number of data values in each Hermitian sequence.  
*Constraint:*  $N \geq 1$ .
- 3: X(M × N) – REAL (KIND=nag\_wp) array *Input*  
*On entry:* the data must be stored in X as if in a two-dimensional array of dimension (1 : M, 0 : N – 1); each of the  $m$  sequences is stored in a **row** of the array in Hermitian form. If the  $n$  data values  $z_j^p$  are written as  $x_j^p + iy_j^p$ , then for  $0 \leq j \leq n/2$ ,  $x_j^p$  is contained in X( $p, j$ ), and for  $1 \leq j \leq (n - 1)/2$ ,  $y_j^p$  is contained in X( $p, n - j$ ). (See also Section 2.1.2 in the C06 Chapter Introduction.)
- 4: U(M × N) – REAL (KIND=nag\_wp) array *Output*  
 5: V(M × N) – REAL (KIND=nag\_wp) array *Output*  
*On exit:* the real and imaginary parts of the  $m$  sequences of length  $n$ , are stored in U and V respectively, as if in two-dimensional arrays of dimension (1 : M, 0 : N – 1); each of the  $m$  sequences is stored as if in a **row** of each array. In other words, if the real parts of the  $p$ th sequence are denoted by  $x_j^p$ , for  $j = 0, 1, \dots, n - 1$  then the  $mn$  elements of the array U contain the values

$$x_0^1, x_0^2, \dots, x_0^m, x_1^1, x_1^2, \dots, x_1^m, \dots, x_{n-1}^1, x_{n-1}^2, \dots, x_{n-1}^m$$

## 6: IFAIL – INTEGER

*Input/Output*

*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

*On exit:* IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

## 6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

On entry,  $M < 1$ .

IFAIL = 2

On entry,  $N < 1$ .

## 7 Accuracy

Exact.

## 8 Further Comments

None.

## 9 Example

This example reads in sequences of real data values which are assumed to be Hermitian sequences of complex data stored in Hermitian form. The sequences are then expanded into full complex form using C06GSF and printed.

### 9.1 Program Text

```

Program c06gsfe

!      C06GSF Example Program Text
!
!      Mark 24 Release. NAG Copyright 2012.
!
!      .. Use Statements ..
!      Use nag_library, Only: c06gsf, nag_wp
!      .. Implicit None Statement ..
!      Implicit None
!      .. Parameters ..
!      Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
!      Integer                    :: i, ieof, ifail, j, m, n
!      .. Local Arrays ..
!      Real (Kind=nag_wp), Allocatable :: u(:), v(:), x(:)
!      .. Executable Statements ..
!      Write (nout,*) 'C06GSF Example Program Results'
!      Skip heading in data file

```

```

      Read (nin,*)
loop: Do
  Read (nin,*,Iostat=ieof) m, n
  If (ieof<0) Exit loop

  Allocate (u(m*n),v(m*n),x(m*n))
  Do j = 1, m
    Read (nin,*)(x(i*m+j),i=0,n-1)
  End Do
  Write (nout,*)
  Write (nout,*) 'Original data values'
  Write (nout,*)
  Do j = 1, m
    Write (nout,99999) '      ', (x(i*m+j),i=0,n-1)
  End Do
  Write (nout,*)
  Write (nout,*) 'Original data written in full complex form'

!      ifail: behaviour on error exit
!      =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
  ifail = 0
  Call c06gsf(m,n,x,u,v,ifail)

  Do j = 1, m
    Write (nout,*)
    Write (nout,99999) 'Real ', (u(i*m+j),i=0,n-1)
    Write (nout,99999) 'Imag ', (v(i*m+j),i=0,n-1)
  End Do
  Deallocate (u,v,x)
End Do loop

99999 Format (1X,A,6F10.4)
End Program c06gsfe

```

## 9.2 Program Data

C06GSF Example Program Data

3	6						: m, n
0.3854	0.6772	0.1138	0.6751	0.6362	0.1424		
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723		
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815		: x

## 9.3 Program Results

C06GSF Example Program Results

Original data values

0.3854	0.6772	0.1138	0.6751	0.6362	0.1424
0.5417	0.2983	0.1181	0.7255	0.8638	0.8723
0.9172	0.0644	0.6037	0.6430	0.0428	0.4815

Original data written in full complex form

Real	0.3854	0.6772	0.1138	0.6751	0.1138	0.6772
Imag	0.0000	0.1424	0.6362	0.0000	-0.6362	-0.1424
Real	0.5417	0.2983	0.1181	0.7255	0.1181	0.2983
Imag	0.0000	0.8723	0.8638	0.0000	-0.8638	-0.8723
Real	0.9172	0.0644	0.6037	0.6430	0.6037	0.0644
Imag	0.0000	0.4815	0.0428	0.0000	-0.0428	-0.4815