NAG Library Function Document

nag_multiple_conjugate_hermitian (c06gqc)

1   Purpose

nag_multiple_conjugate_hermitian (c06gqc) forms the complex conjugates of m Hermitian sequences, each containing n data values.

2   Specification

```
#include <nag.h>
#include <nagc06.h>
void nag_multiple_conjugate_hermitian (Integer m, Integer n, double x[],
      NagError *fail)
```

3   Description

This is a utility function for use in conjunction with nag_fft_multiple_real (c06fpc) and nag_fft_multiple_hermitian (c06fqc) to calculate inverse discrete Fourier transforms.

4   References

None.

5   Arguments

1:    **m** – Integer   *Input*

On entry: the number of Hermitian sequences to be conjugated, m.

Constraint: **m** ≥ 1.

2:    **n** – Integer   *Input*

On entry: the number of data values in each Hermitian sequence, n.

Constraint: **n** ≥ 1.

3:    **x**[**m** $\times$ **n**] – double   *Input/Output*

On entry: the m data sequences must be stored in **x** consecutively. If the n data values z_j^p are written as $x_j^p + iy_j^p$, $p = 1, 2, \ldots, m$, then for $0 \leq j \leq n/2$, x_j^p is contained in $\mathbf{x}[(p-1) \times n + j]$, and for $1 \leq j \leq (n-1)/2$, y_j^p is contained in $\mathbf{x}[(p-1) \times n + n - j]$.

On exit: the imaginary parts y_j^p are negated. The real parts x_j^p are not referenced.

4:    **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_INT_ARG_LT

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

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

7   Accuracy

Exact.

8   Parallelism and Performance

nag_multiple_conjugate_hermitian (c06gqc) is not threaded in any implementation.

9   Further Comments

None.

10   Example

This program reads in sequences of real data values which are assumed to be Hermitian sequences of complex data stored in Hermitian form. The sequences are expanded into full complex form using nag_multiple_hermitian_to_complex (c06gsc) and printed. The sequences are then conjugated (using nag_multiple_conjugate_hermitian (c06gqc)) and the conjugated sequences are expanded into complex form using nag_multiple_hermitian_to_complex (c06gsc) and printed out.

10.1 Program Text

```
/* nag_multiple_conjugate_hermitian (c06gqc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
  Integer exit_status = 0, i, j, m, n;
  NagError fail;
  double *u = 0, *v = 0, *x = 0;

  INIT_FAIL(fail);

  printf("nag_multiple_conjugate_hermitian (c06gqc) Example Program"
         " Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  while (scanf_s("%" NAG_IFMT "%" NAG_IFMT "", &m, &n) != EOF)
```

```
#else
  while (scanf("%" NAG_IFMT "%" NAG_IFMT "", &m, &n) != EOF)
#endif
  {
    if (m >= 1 && n >= 1) {
      if (!(u = NAG_ALLOC(m * n, double)) ||
          !(v = NAG_ALLOC(m * n, double)) || !(x = NAG_ALLOC(m * n, double)))
      {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
      }
    }
    else {
      printf("Invalid m or n.\n");
      exit_status = 1;
      return exit_status;
    }
    printf("\n\nm = %2" NAG_IFMT "  n = %2" NAG_IFMT "\n", m, n);
    /* Read in data and print out. */
    for (j = 0; j < m; ++j)
      for (i = 0; i < n; ++i)
#ifdef _WIN32
        scanf_s("%lf", &x[j * n + i]);
#else
        scanf("%lf", &x[j * n + i]);
#endif
    printf("\nOriginal data values\n\n");
    for (j = 0; j < m; ++j) {
      printf("     ");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", x[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\n");
    }
    /* Convert Hermitian form data to full complex */
    /* nag_multiple_hermitian_to_complex (c06gsc).
     * Convert Hermitian sequences to general complex sequences
     */
    nag_multiple_hermitian_to_complex(m, n, x, u, v, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_multiple_hermitian_to_complex"
             " (c06gsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\nOriginal data written in full complex form\n\n");
    for (j = 0; j < m; ++j) {
      printf("Real");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", u[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\nImag");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", v[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\n\n");
    }
    /* Calculate conjugates of Hermitian data */
    /* nag_multiple_conjugate_hermitian (c06gqc).
     * Complex conjugate of multiple Hermitian sequences
     */
    nag_multiple_conjugate_hermitian(m, n, x, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_multiple_conjugate_hermitian (c06gqc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
```

```
    printf("\nConjugated data values\n\n");
    for (j = 0; j < m; ++j) {
      printf("     ");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", x[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\n");
    }
    /* Convert conjugated Hermitian data to full complex */
    /* nag_multiple_hermitian_to_complex (c06gsc), see above. */
    nag_multiple_hermitian_to_complex(m, n, x, u, v, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_multiple_hermitian_to_complex"
             " (c06gsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\nConjugated data written in full complex form\n\n");
    for (j = 0; j < m; ++j) {
      printf("Real");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", u[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\nImag");
      for (i = 0; i < n; ++i)
        printf("%10.4f%s", v[j * n + i],
               (i % 6 == 5 && i != n - 1 ? "\n     " : ""));
      printf("\n\n");
    }
  END:
    NAG_FREE(u);
    NAG_FREE(v);
    NAG_FREE(x);
  }
  return exit_status;
}
```

10.2 Program Data

```
nag_multiple_conjugate_hermitian (c06gqc) Example Program Data
     3      6
    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
```

10.3 Program Results

```
nag_multiple_conjugate_hermitian (c06gqc) Example Program Results


m =  3  n =  6

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
```

```
Conjugated 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

Conjugated 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
```