

NAG Library Function Document

nag_fft_init_trig (c06gzc)

1 Purpose

nag_fft_init_trig (c06gzc) calculates the trigonometric coefficients required for the computation of discrete Fourier transforms.

2 Specification

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

3 Description

This is a utility function for use in conjunction with one or more of nag_fft_multiple_real (c06fpc), nag_fft_multiple_hermitian (c06fqc), nag_fft_multiple_complex (c06frc), nag_fft_2d_complex (c06fuc), nag_fft_multiple_sine (c06hac), nag_fft_multiple_cosine (c06hbc), nag_fft_multiple_qtr_sine (c06hcc) and nag_fft_multiple_qtr_cosine (c06hdc). nag_fft_init_trig (c06gzc) initializes the array **trig** with trigonometric coefficients according to the value of **n** and must be called prior to the first call of one of the above listed functions.

4 References

None.

5 Arguments

- | | | |
|----|--|---------------------|
| 1: | n – Integer | <i>Input</i> |
| | <i>On entry:</i> the value of <i>n</i> in the Fourier transform function being called. | |
| | <i>Constraint:</i> $n \geq 1$. | |
| 2: | trig [$2 \times n$] – double | <i>Output</i> |
| | <i>On exit:</i> the trigonometric coefficients are stored in trig . | |
| 3: | fail – NagError * | <i>Input/Output</i> |
| | The NAG error argument (see Section 3.6 in the Essential Introduction). | |

6 Error Indicators and Warnings

NE_INT_ARG_LT

On entry, **n** = $\langle value \rangle$.
 Constraint: $n \geq 1$.

7 Accuracy

Exact.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

The program reads in 3 real data sequences and prints their discrete Fourier transforms in Hermitian format as calculated by `nag_fft_multiple_real` (c06fpc). A call is made to `nag_fft_init_trig` (c06gzc) to initialize the array `trig` prior to calling `nag_fft_multiple_real` (c06fpc). The transforms are then printed out in full complex form after a call to `nag_multiple_hermitian_to_complex` (c06gsc).

10.1 Program Text

```

/* nag_fft_init_trig (c06gzc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

#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     *trig = 0, *u = 0, *v = 0, *x = 0;

    INIT_FAIL(fail);

    printf("nag_fft_init_trig (c06gzc) Example Program Results\n");
#ifdef _WIN32
    scanf_s("%*[\n]"); /* Skip heading in data file */
#else
    scanf("%*[\n]"); /* Skip heading in data file */
#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 (!(trig = NAG_ALLOC(2*n, double)) ||
                !(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;
        }
    }
}

```

```

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");
    }
/* nag_fft_init_trig (c06gzc).
 * Initialization function for other c06 functions
 */
nag_fft_init_trig(n, trig, &fail);
if (fail.code != NE_NOERROR)
    {
    printf("Error from nag_fft_init_trig (c06gzc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
    }
/* Initialize trig array */
/* Calculate transform */
/* nag_fft_multiple_real (c06fpc).
 * Multiple one-dimensional real discrete Fourier transforms
 */
nag_fft_multiple_real(m, n, x, trig, &fail);
if (fail.code != NE_NOERROR)
    {
    printf("Error from nag_fft_multiple_real (c06fpc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
    }

printf("\nDiscrete Fourier transforms in Hermitian format\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 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("\nFourier transforms 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(trig);
    NAG_FREE(u);
    NAG_FREE(v);
    NAG_FREE(x);
}

return exit_status;
}

```

10.2 Program Data

nag_fft_init_trig (c06gzc) 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_fft_init_trig (c06gzc) 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

Discrete Fourier transforms in Hermitian format

1.0737	-0.1041	0.1126	-0.1467	-0.3738	-0.0044
1.3961	-0.0365	0.0780	-0.1521	-0.0607	0.4666
1.1237	0.0914	0.3936	0.1530	0.3458	-0.0508

Fourier transforms in full complex form

Real	1.0737	-0.1041	0.1126	-0.1467	0.1126	-0.1041
Imag	0.0000	-0.0044	-0.3738	0.0000	0.3738	0.0044
Real	1.3961	-0.0365	0.0780	-0.1521	0.0780	-0.0365
Imag	0.0000	0.4666	-0.0607	0.0000	0.0607	-0.4666
Real	1.1237	0.0914	0.3936	0.1530	0.3936	0.0914
Imag	0.0000	-0.0508	0.3458	0.0000	-0.3458	0.0508
