NAG Library Function Document

nag_fresnel_c_vector (s20arc)

1 Purpose

nag_fresnel_c_vector (s20arc) returns an array of values for the Fresnel integral $C(x)$.

2 Specification

```c
#include <nag.h>
#include <nags.h>

void nag_fresnel_c_vector (Integer n, const double x[], double f[], NagError *fail)
```

3 Description

nag_fresnel_c_vector (s20arc) evaluates an approximation to the Fresnel integral

\[ C(x) = \int_{0}^{x} \cos\left(\frac{\pi t^2}{4}\right) dt \]

for an array of arguments $x_i$, for $i = 1, 2, \ldots, n$.

**Note:** $C(x) = -C(-x)$, so the approximation need only consider $x \geq 0.0$.

The function is based on three Chebyshev expansions:

For $0 < x \leq 3$,

\[ C(x) = x \sum_{r=0} a_r T_r(t), \quad \text{with } t = 2\left(\frac{x}{3}\right)^4 - 1. \]

For $x > 3$,

\[ C(x) = \frac{1}{2} + \frac{f(x)}{x} \sin\left(\frac{\pi x^2}{2}\right) - \frac{g(x)}{x^3} \cos\left(\frac{\pi x^2}{2}\right), \]

where $f(x) = \sum_{r=0} b_r T_r(t)$,

and $g(x) = \sum_{r=0} c_r T_r(t)$,

with $t = 2\left(\frac{3}{x}\right)^4 - 1$.

For small $x$, $C(x) \approx x$. This approximation is used when $x$ is sufficiently small for the result to be correct to *machine precision*.

For large $x$, $f(x) \approx \frac{1}{\pi}$ and $g(x) \approx \frac{1}{\pi^2 x}$. Therefore for moderately large $x$, when $\frac{1}{\pi^2 x}$ is negligible compared with $\frac{1}{2}$, the second term in the approximation for $x > 3$ may be dropped. For very large $x$, when $\frac{1}{\pi x}$ becomes negligible, $C(x) \approx \frac{1}{2}$. However there will be considerable difficulties in calculating $\sin\left(\frac{\pi x^2}{2}\right)$ accurately before this final limiting value can be used. Since $\sin\left(\frac{\pi x^2}{2}\right)$ is periodic, its value is essentially determined by the fractional part of $x^2$. If $x^2 = N + \theta$, where $N$ is an integer and $0 \leq \theta < 1$, then $\sin\left(\frac{\pi x^2}{2}\right)$ depends on $\theta$ and on $N$ modulo $4$. By exploiting this fact, it is possible to retain some
significance in the calculation of \( \sin(\frac{\pi x^2}{2}) \) either all the way to the very large \( x \) limit, or at least until the integer part of \( \frac{x}{2} \) is equal to the maximum integer allowed on the machine.

4 References

5 Arguments
1: \( n \) – Integer
   \( \text{Input} \)
   \( \text{On entry}: n, \text{the number of points.} \)
   \( \text{Constraint}: n \geq 0. \)
2: \( x \) – const double
   \( \text{Input} \)
   \( \text{On entry}: \text{the argument } x_i \text{ of the function, for } i = 1, 2, \ldots, n. \)
3: \( f \) – double
   \( \text{Output} \)
   \( \text{On exit}: C(x_i), \text{the function values.} \)
4: \( \text{fail} \) – NagError *
   \( \text{Input/Output} \)
   \( \text{The NAG error argument (see Section 3.6 in the Essential Introduction).} \)

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 \( \langle \text{value} \rangle \) had an illegal value.

NE_INT
   On entry, \( n = \langle \text{value} \rangle \).
   Constraint: \( n \geq 0. \)

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
Let \( \delta \) and \( \epsilon \) be the relative errors in the argument and result respectively.
If $\delta$ is somewhat larger than the machine precision (i.e. if $\delta$ is due to data errors etc.), then $\epsilon$ and $\delta$ are approximately related by:

$$
\epsilon \simeq \left| \frac{x \cos \left( \frac{\pi}{2} x^2 \right)}{C(x)} \right| \delta.
$$

Figure 1 shows the behaviour of the error amplification factor $\left| \frac{x \cos \left( \frac{\pi}{2} x^2 \right)}{C(x)} \right|$. However, if $\delta$ is of the same order as the machine precision, then rounding errors could make $\epsilon$ slightly larger than the above relation predicts.

For small $x$, $\epsilon \simeq \delta$ and there is no amplification of relative error. For moderately large values of $x$,

$$
\epsilon \simeq 2x \cos \left( \frac{\pi}{2} x^2 \right) \delta
$$

and the result will be subject to increasingly large amplification of errors. However the above relation breaks down for large values of $x$ (i.e., when $\frac{1}{x^2}$ is of the order of the machine precision); in this region the relative error in the result is essentially bounded by $\frac{2}{\pi x}$.

Hence the effects of error amplification are limited and at worst the relative error loss should not exceed half the possible number of significant figures.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.
10 Example

This example reads values of $x$ from a file, evaluates the function at each value of $x_i$ and prints the results.

10.1 Program Text

/* nag_fresnel_c_vector (s20arc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nags.h>

int main(void)
{
    Integer exit_status = 0;
    Integer i, n;
    double *f = 0, *x = 0;
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_fresnel_c_vector (s20arc) Example Program Results\n");
    printf("\n");
    printf(" x  f\n");
    printf("\n");
    #ifndef _WIN32
        scanf_s("%*[\n"]);
    #else
        scanf("%*[\n"]);
    #endif
    /* Allocate memory */
    if (!(x = NAG_ALLOC(n, double)) ||
        !(f = NAG_ALLOC(n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    for (i=0; i<n; i++)
    {
        #ifndef _WIN32
            scanf_s("%lf", &x[i]);
        #else
            scanf("%lf", &x[i]);
        #endif
        #ifndef _WIN32
            scanf_s("%*[\n"]);
        #else
            scanf("%*[\n"]);
        #endif

        { /* Skip heading in data file */
            #ifdef _WIN32
                scanf_s("%*\n");
            #else
                scanf("%*\n");
            #endif
            #ifdef _WIN32
                scanf_s("%*\n");
            #else
                scanf("%*\n");
            #endif
            /* Skip heading in data file */
            #ifdef _WIN32
                scanf_s("%*[\n"]);
            #else
                scanf("%*[\n"]);
            #endif
            #ifdef _WIN32
                scanf_s("%*\n");
            #else
                scanf("%*\n");
            #endif
        }
    }

    for (i=0; i<n; i++)
    {
        *f = nag_fresnel_c(n, *x);
        printf("%lf\n", *f);
    }
    printf("\n");
    return 0;
}

END:
nag_fresnel_c_vector (s20arc) Example Program Data

11
0.0 0.5 1.0 2.0 4.0 5.0 6.0 8.0 10.0 -1.0 1000.0

nag_fresnel_c_vector (s20arc) Example Program Results

<table>
<thead>
<tr>
<th>x</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000e+00</td>
<td>0.000e+00</td>
</tr>
<tr>
<td>5.000e-01</td>
<td>4.923e-01</td>
</tr>
<tr>
<td>1.000e+00</td>
<td>7.799e-01</td>
</tr>
<tr>
<td>2.000e+00</td>
<td>4.883e-01</td>
</tr>
<tr>
<td>4.000e+00</td>
<td>4.984e-01</td>
</tr>
<tr>
<td>5.000e+00</td>
<td>5.636e-01</td>
</tr>
<tr>
<td>6.000e+00</td>
<td>4.995e-01</td>
</tr>
<tr>
<td>8.000e+00</td>
<td>4.998e-01</td>
</tr>
<tr>
<td>1.000e+01</td>
<td>4.999e-01</td>
</tr>
<tr>
<td>-1.000e+00</td>
<td>-7.799e-01</td>
</tr>
<tr>
<td>1.000e+03</td>
<td>5.000e-01</td>
</tr>
</tbody>
</table>