NAG Library Function Document

nag_inteq_fredholm2_smooth (d05abc)

1 Purpose

nag_inteq_fredholm2_smooth (d05abc) solves any linear nonsingular Fredholm integral equation of the second kind with a smooth kernel.

2 Specification

```c
#include <nag.h>
#include <nagd05.h>

void nag_inteq_fredholm2_smooth(double lambda, double a, double b,
                                 Integer n,
                                 double (*k)(double x, double s, Nag_Comm *comm),
                                 double (*g)(double x, Nag_Comm *comm),
                                 Nag_Boolean odorev, Nag_Boolean ev,
                                 double f[], double c[],
                                 Nag_Comm *comm, NagError *fail)
```

3 Description

nag_inteq_fredholm2_smooth (d05abc) uses the method of El–Gendi (1969) to solve an integral equation of the form

\[ f(x) - \lambda \int_a^b k(x, s)f(s) \, ds = g(x) \]

for the function \( f(x) \) in the range \( a \leq x \leq b \).

An approximation to the solution \( f(x) \) is found in the form of an \( n \) term Chebyshev series \( \sum_{i=1}^{n} c_i T_i(x) \), where ' indicates that the first term is halved in the sum. The coefficients \( c_i \), for \( i = 1, 2, \ldots, n \), of this series are determined directly from approximate values \( f_i \), for \( i = 1, 2, \ldots, n \), of the function \( f(x) \) at the first \( n \) of a set of \( m+1 \) Chebyshev points

\[ x_i = \frac{1}{2}(a + b + (b - a) \times \cos([i - 1] \times \pi/m)), \quad i = 1, 2, \ldots, m + 1. \]

The values \( f_i \) are obtained by solving a set of simultaneous linear algebraic equations formed by applying a quadrature formula (equivalent to the scheme of Clenshaw and Curtis (1960)) to the integral equation at each of the above points.

In general \( m = n - 1 \). However, advantage may be taken of any prior knowledge of the symmetry of \( f(x) \). Thus if \( f(x) \) is symmetric (i.e., even) about the mid-point of the range \( (a, b) \), it may be approximated by an even Chebyshev series with \( m = 2n - 1 \). Similarly, if \( f(x) \) is anti-symmetric (i.e., odd) about the mid-point of the range of integration, it may be approximated by an odd Chebyshev series with \( m = 2n \).

4 References

Numer. Math. 2 197–205

Comput. J. 12 282–287
5 Arguments

1: \textbf{lambda} – double \hspace{1cm} \textit{Input}
   
   On entry: the value of the parameter \( \lambda \) of the integral equation.

2: \textit{a} – double \hspace{1cm} \textit{Input}
   
   On entry: \( a \), the lower limit of integration.

3: \textit{b} – double \hspace{1cm} \textit{Input}
   
   On entry: \( b \), the upper limit of integration.
   
   Constraint: \( b > a \).

4: \textit{n} – Integer \hspace{1cm} \textit{Input}
   
   On entry: the number of terms in the Chebyshev series which approximates the solution \( f(x) \).
   
   Constraint: \( n \geq 1 \).

5: \textit{k} – function, supplied by the user \hspace{1cm} \textit{External Function}
   
   \( k \) must compute the value of the kernel \( k(x, s) \) of the integral equation over the square \( a \leq x \leq b, \ a \leq s \leq b \).

   The specification of \( k \) is:
   
   \begin{verbatim}
   double k (double x, double s, Nag_Comm *comm)
   \end{verbatim}

   1: \textit{x} – double \hspace{1cm} \textit{Input}
      
      On entry: the values of \( x \) and \( s \) at which \( k(x, s) \) is to be calculated.

   2: \textit{s} – double \hspace{1cm} \textit{Input}
      
      Pointer to structure of type Nag_Comm; the following members are relevant to \( k \).

      \begin{verbatim}
      user – double *
      iuser – Integer *
      p – Pointer
      \end{verbatim}

      The type Pointer will be void *. Before calling nag_inteq_fredholm2_smooth (d05abc) you may allocate memory and initialize these pointers with various quantities for use by \( k \) when called from nag_inteq_fredholm2_smooth (d05abc) (see Section 3.2.1.1 in the Essential Introduction).

6: \textit{g} – function, supplied by the user \hspace{1cm} \textit{External Function}
   
   \( g \) must compute the value of the function \( g(x) \) of the integral equation in the interval \( a \leq x \leq b \).

   The specification of \( g \) is:
   
   \begin{verbatim}
   double g (double x, Nag_Comm *comm)
   \end{verbatim}

   1: \textit{x} – double \hspace{1cm} \textit{Input}
      
      On entry: the value of \( x \) at which \( g(x) \) is to be calculated.

   2: \textit{comm} – Nag_Comm *
      
      Pointer to structure of type Nag_Comm; the following members are relevant to \( g \).
user – double *
iuser – Integer *
p – Pointer

The type Pointer will be void *. Before calling nag_inteq_fredholm2_smooth (d05abc) you may allocate memory and initialize these pointers with various quantities for use by g when called from nag_inteq_fredholm2_smooth (d05abc) (see Section 3.2.1.1 in the Essential Introduction).

7: odorev – Nag_Boolean
   Input
   On entry: indicates whether it is known that the solution \( f(x) \) is odd or even about the mid-point of the range of integration. If odorev is Nag_TRUE then an odd or even solution is sought depending upon the value of ev.

8: ev – Nag_Boolean
   Input
   On entry: is ignored if odorev is Nag_FALSE. Otherwise, if ev is Nag_TRUE, an even solution is sought, whilst if ev is Nag_FALSE, an odd solution is sought.

9: f[n] – double
   Output
   On exit: the approximate values \( f_i \), for \( i = 1, 2, \ldots, n \), of the function \( f(x) \) at the first \( n \) of \( m + 1 \) Chebyshev points (see Section 3), where
   
   \[
   m = 2n - 1 \quad \text{if} \quad \text{odorev} = \text{Nag_TRUE} \quad \text{and} \quad \text{ev} = \text{Nag_TRUE}. \\
   m = 2n \quad \text{if} \quad \text{odorev} = \text{Nag_TRUE} \quad \text{and} \quad \text{ev} = \text{Nag_FALSE}. \\
   m = n - 1 \quad \text{if} \quad \text{odorev} = \text{Nag_FALSE}.
   \]

10: c[n] – double
   Output
   On exit: the coefficients \( c_i \), for \( i = 1, 2, \ldots, n \), of the Chebyshev series approximation to \( f(x) \). When odorev is Nag_TRUE, this series contains polynomials of even order only or of odd order only, according to ev being Nag_TRUE or Nag_FALSE respectively.

11: comm – Nag_Comm *
   The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

12: fail – NagError *
   Input/Output
   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 \( \text{value} \) had an illegal value.

NE_EIGENVALUES
   A failure has occurred due to proximity of an eigenvalue.

NE_INT
   On entry, \( n = \langle \text{value} \rangle \).
   Constraint: \( n \geq 1 \).
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.

NE_REAL_2
On entry, a = \langle value \rangle and b = \langle value \rangle.
Constraint: b > a.

7 Accuracy
No explicit error estimate is provided by the function but it is possible to obtain a good indication of the accuracy of the solution either
(i) by examining the size of the later Chebyshev coefficients $c_i$, or
(ii) by comparing the coefficients $c_i$ or the function values $f_i$ for two or more values of $n$.

8 Parallelism and Performance
nag_inteq_fredholm2_smooth (d05abc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_inteq_fredholm2_smooth (d05abc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments
The time taken by nag_inteq_fredholm2_smooth (d05abc) depends upon the value of $n$ and upon the complexity of the kernel function $k(x, s)$.

10 Example
This example solves Love’s equation:

$$f(x) + \frac{1}{\pi} \int_{-1}^{1} \frac{f(s)}{1 + (x - s)^2} ds = 1.$$ 

It will solve the slightly more general equation:

$$f(x) - \lambda \int_{a}^{b} k(x, s)f(s) ds = 1$$

where $k(x, s) = \alpha / \left( \alpha^2 + (x - s)^2 \right)$. The values $\lambda = -1/\pi, a = -1, b = 1, \alpha = 1$ are used below.

It is evident from the symmetry of the given equation that $f(x)$ is an even function. Advantage is taken of this fact both in the application of nag_inteq_fredholm2_smooth (d05abc), to obtain the $f_i \approx f(x_i)$.
and the $c_i$, and in subsequent applications of nag_sum_cheby_series (c06dcc) to obtain $f(x)$ at selected points.

The program runs for $n = 5$ and $n = 10$.

### 10.1 Program Text

```c
/* nag_inteq_fredholm2_smooth (d05abc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group. *
 * Mark 23, 2011. */

#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagc06.h>
#include <nagd05.h>
#include <nagx01.h>

#ifdef __cplusplus
extern "C" {
#endif

static double NAG_CALL k(double x, double s, Nag_Comm *comm);
static double NAG_CALL g(double x, Nag_Comm *comm);

#ifdef __cplusplus
}
#endif

int main(void)
{
    /* Scalars */
    double a = -1.0, b = 1.0;
    double lambda, x0;
    Integer exit_status = 0;
    Integer i, lx, n;
    Nag_Boolean ev = Nag_TRUE, odorev = Nag_TRUE;

    /* Arrays */
    static double ruser[2] = {-1.0, -1.0};
    double *c = 0, *chebr = 0, *f = 0, *x = 0;

    /* NAG types */
    Nag_Comm comm;
    NagError fail;
    Nag_Series s = Nag_SeriesEven;

    INIT_FAIL(fail);

    printf("nag_inteq_fredholm2_smooth (d05abc) Example Program Results\n");

    /* For communication with user-supplied functions: */
    comm.user = ruser;

    x0 = 0.5 * (a + b);

    /* Set up uniform grid to evaluate Chebyshev polynomials. */
    lx = (Integer) (4.000001 * (b - x0)) + 1;

    if (! (x = NAG_ALLOC(lx, double)) ||
          (chebr = NAG_ALLOC(lx, double)) )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    x[0] = x0;
    for (i = 1; i < lx; i++)
        x[i] = x[i - 1] + 0.25;

    ...
printf("\nSolution is even\n");

lambda = -1.0 / nag_pi;

for (n = 5; n <= 10; n += 5)
{
    if (f = NAG_ALLOC(n, double)) ||
        !(c = NAG_ALLOC(n, double))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /*
    nag_inteq_fredholm2_smooth (d05abc).
    Linear non-singular Fredholm integral equation, second kind,
    smooth kernel.
    */
    nag_inteq_fredholm2_smooth(lambda, a, b, n, k, g, odorev, ev, f, c, &comm,
            &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_inteq_fredholm2_smooth (d05abc).\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }

    printf("\nResults for n = %"NAG_IFMT"\n\n", n);
    printf("Solution on first %2"NAG_IFMT" Chebyshev points and Chebyshev" "
coefficients\n",n);
    printf("%3s%12s%18s%12s\n","i","x","f[i]","c[i]");
    for (i = 0; i < n; i++)
    {
        double y = cos(nag_pi*(double) (i)/(double)(2*n-1));
        printf("%3"NAG_IFMT"%15.5f%15.5f%15.5e\n", i, y, f[i], c[i]);
    }
    printf("\n");

    /*
    Evaluate and print solution on uniform grid.
    nag_sum_cheby_series (c06dcc).
    Sum of a Chebyshev series at a set of points.
    */
    nag_sum_cheby_series(x, lx, a, b, c, n, s, chebr, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_sum_cheby_series (c06dcc).\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }

    printf("\nSolution on evenly spaced grid\n");
    printf("\n x f(x)\n");
    for (i = 0; i < lx; i++)
        printf("%8.4f%15.5f\n", x[i], chebr[i]);
    printf("\n");

    NAG_FREE(c);
    NAG_FREE(f);
}

END:
NAG_FREE(c);
NAG_FREE(f);
NAG_FREE(chebr);
NAG_FREE(x);

return exit_status;
}

static double NAG_CALL k(double x, double s, Nag_Comm *comm)
{
  /* Scalars */
  double alpha = 1.0;

  if (comm->user[0] == -1.0)
  {
    printf("(User-supplied callback k, first invocation.)\n");
    comm->user[0] = 0.0;
  }
  return alpha / (pow(alpha, 2) + pow(x - s, 2));
}

static double NAG_CALL g(double x, Nag_Comm *comm)
{
  if (comm->user[1] == -1.0)
  {
    printf("(User-supplied callback g, first invocation.)\n");
    comm->user[1] = 0.0;
  }
  return 1.0;
}

10.2 Program Data

None.

10.3 Program Results

nag_inteq_fredholm2_smooth (d05abc) Example Program Results

Solution is even
(User-supplied callback g, first invocation.)
(User-supplied callback k, first invocation.)

Results for n = 5

Solution on first 5 Chebyshev points and Chebyshev coefficients

<table>
<thead>
<tr>
<th>i</th>
<th>x</th>
<th>f[i]</th>
<th>c[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000</td>
<td>0.75572</td>
<td>1.41519e+00</td>
</tr>
<tr>
<td>1</td>
<td>0.93969 0.74534</td>
<td>4.93850e-02</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.76604 0.71729</td>
<td>-1.04759e-03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.50000 0.68318</td>
<td>-2.32823e-04</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.17365 0.66050</td>
<td>2.08908e-05</td>
<td></td>
</tr>
</tbody>
</table>

Solution on evenly spaced grid

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.65741</td>
</tr>
<tr>
<td>0.2500</td>
<td>0.66383</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.68318</td>
</tr>
<tr>
<td>0.7500</td>
<td>0.71488</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.75572</td>
</tr>
</tbody>
</table>

Results for n = 10

Solution on first 10 Chebyshev points and Chebyshev coefficients

<table>
<thead>
<tr>
<th>i</th>
<th>x</th>
<th>f[i]</th>
<th>c[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000</td>
<td>0.75572</td>
<td>1.41519e+00</td>
</tr>
<tr>
<td>1</td>
<td>0.98636 0.75335</td>
<td>4.93851e-02</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.94582 0.74638</td>
<td>-1.04752e-03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.87947 0.73524</td>
<td>-2.32755e-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>f(x)</td>
<td>y''(x)</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>4</td>
<td>0.78914</td>
<td>0.72081</td>
<td>1.99861e-05</td>
</tr>
<tr>
<td>5</td>
<td>0.67728</td>
<td>0.70451</td>
<td>9.86781e-07</td>
</tr>
<tr>
<td>6</td>
<td>0.54695</td>
<td>0.68824</td>
<td>-2.37962e-07</td>
</tr>
<tr>
<td>7</td>
<td>0.40170</td>
<td>0.67403</td>
<td>1.85813e-09</td>
</tr>
<tr>
<td>8</td>
<td>0.24549</td>
<td>0.66360</td>
<td>2.44835e-09</td>
</tr>
<tr>
<td>9</td>
<td>0.08258</td>
<td>0.65811</td>
<td>-1.65272e-10</td>
</tr>
</tbody>
</table>

Solution on evenly spaced grid

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.65741</td>
</tr>
<tr>
<td>0.2500</td>
<td>0.66383</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.68318</td>
</tr>
<tr>
<td>0.7500</td>
<td>0.71488</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.75572</td>
</tr>
</tbody>
</table>