nag_zero_nonlin_eqns_1 (c05tbc)

1. Purpose

nag_zero_nonlin_eqns_1 (c05tbc) finds a solution of a system of nonlinear equations by a modification of the Powell hybrid method.

2. Specification

```
#include <nag.h>
#include <nagc05.h>

void nag_zero_nonlin_eqns_1(Integer n, double x[], double fvec[],
             void (*f)(Integer n, double x[], double fvec[],
                    Integer *userflag),
             double xtol, Nag_User *comm, NagError *fail)
```

3. Description

The system of equations is defined as:

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad \text{for} \quad i = 1, 2, \ldots, n.$$

nag_zero_nonlin_eqns_1 is based upon the MINPACK routine HYBRD1 (Moré *et al* (1980)). It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. Under reasonable conditions this guarantees global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is updated by the rank-1 method of Broyden. At the starting point the Jacobian is approximated by forward differences, but these are not used again until the rank-1 method fails to produce satisfactory progress. For more details see Powell (1970).

4. Parameters

n

Input: the number of equations, n.
Constraint: $\mathbf{n} > 0$.

x[n]

Input: an initial guess at the solution vector.
Output: the final estimate of the solution vector.

fvec[n]

Output: the function values at the final point, **x**.

f

The function **f**, supplied by the user, must return the values of the f_i at a point x.
The specification of **f** is:

```
void f(Integer n, double x[], double fvec[], Integer *userflag)
```

n

Input: the number of equations, n.

x[n]

Input: the components of the point x at which the functions must be evaluated.

fvec[n]

Output: the function values $f_i(x)$ (unless **userflag** is set to a negative value by **f**).

userflag

Input: $\mathbf{userflag} > 0$.
Output: in general, **userflag** should not be reset by **f**. If, however, the user wishes to terminate execution (perhaps because some illegal point **x** has been reached), then **userflag** should be set to a negative integer. This value will be returned through **fail.errnum**.

xtol

Input: the accuracy in **x** to which the solution is required.
Suggested value: the square root of the ***machine precision***.
Constraint: **xtol** $\geq$ 0.0.

comm

Input/Output: pointer to a structure of type Nag_User with the following member:

p - Pointer

Input/Output: the pointer **p**, of type Pointer, allows the user to communicate information to and from the user-defined function **f()**. An object of the required type should be declared by the user, e.g. a structure, and its address assigned to the pointer **p** by means of a cast to Pointer in the calling program, e.g. `comm.p = (Pointer)&s`. The type pointer will be `void *` with a C compiler that defines `void *` and `char *` otherwise.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LE

On entry, **n** must not be less than or equal to 0: **n** = $\langle$*value*$\rangle$.

NE_REAL_ARG_LT

On entry, **xtol** must not be less than 0.0: **xtol** = $\langle$*value*$\rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_USER_STOP

User requested termination, user flag value = $\langle$*value*$\rangle$.

NE_TOO_MANY_FUNC_EVAL

There have been at least 200 $*$ (**n**+1) evaluations of **f**().

Consider restarting the calculation from the point held in **x**.

NE_XTOL_TOO_SMALL

No further improvement in the solution is possible. **xtol** is too small: **xtol** = $\langle$*value*$\rangle$.

NE_NO_IMPROVEMENT

The iteration is not making good progress.

This failure exit may indicate that the system does not have a zero, or that the solution is very close to the origin (see Section 6.1). Otherwise, rerunning nag_zero_nonlin_eqns_1 from a different starting point may avoid the region of difficulty.

6. Further Comments

The time required by nag_zero_nonlin_eqns_1 to solve a given problem depends on n, the behaviour of the functions, the accuracy requested and the starting point. The number of arithmetic operations executed by nag_zero_nonlin_eqns_1 to process each call of **f** is about 11.5 $\times$ n^2. Unless **f** can be evaluated quickly, the timing of nag_zero_nonlin_eqns_1 will be strongly influenced by the time spent in **f**.

Ideally the problem should be scaled so that at the solution the function values are of comparable magnitude.

6.1. Accuracy

If $\hat{x}$ is the true solution, nag_zero_nonlin_eqns_1 tries to ensure that

$$\|x - \hat{x}\| \leq \mathbf{xtol} \times \|\hat{x}\|.$$

If this condition is satisfied with **xtol** = 10^{-k}, then the larger components of x have k significant decimal digits. There is a danger that the smaller components of x may have large relative errors, but the fast rate of convergence of nag_zero_nonlin_eqns_1 usually avoids this possibility.

If **xtol** is less than ***machine precision***, and the above test is satisfied with the ***machine precision*** in place of **xtol**, then the routine exits with **NE_XTOL_TOO_SMALL**.

Note: this convergence test is based purely on relative error, and may not indicate convergence if the solution is very close to the origin.

The test assumes that the functions are reasonably well behaved. If this condition is not satisfied, then nag_zero_nonlin_eqns_1 may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning nag_zero_nonlin_eqns_1 with a tighter tolerance.

6.2. References

Moré J J, Garbow B S and Hillstrom K E (1980) *User Guide for MINPACK-1* Argonne National Laboratory, ANL-80-74.

Powell M J D (1970) A Hybrid Method for Nonlinear Algebraic Equations *Numerical Methods for Nonlinear Algebraic Equations* P Rabinowitz (ed) Gordon and Breach.

7. See Also

nag_zero_nonlin_eqns_deriv_1 (c05ubc)

8. Example

To determine the values $x_1, \ldots, x_9$ which satisfy the tridiagonal equations:

$$\begin{array}{llllll}
(3-2x_1)x_1 & - & 2x_2 & & & = -1 \\
-x_{i-1} & + & (3-2x_i)x_i & - & 2x_{i+1} & = -1, \quad i = 2,3,\ldots,8 \\
 & & -x_8 & + & (3-2x_9)x_9 & = -1.
\end{array}$$

8.1. Program Text

```
/* nag_zero_nonlin_eqns_1(c05tbc) Example Program
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagc05.h>
#include <nagx02.h>

#ifdef NAG_PROTO
static void f(Integer n, double x[], double fvec[], Integer *userflag, Nag_User *comm);
#else
static void f();
#endif

#define NMAX 9
main()
{

  double fvec[NMAX];
  Integer i, j;
  double x[NMAX];
  double xtol;
  static NagError fail;
  Nag_User comm;

  Integer n = NMAX;

  Vprintf("c05tbc Example Program Results\n");
  /* The following starting values provide a rough solution. */
  for (j=0; j<n; j++)
```

```
      x[j] = -1.0;
    xtol = sqrt(X02AJC);
    c05tbc(n, x, fvec, f, xtol, &comm, &fail);
    if (fail.code == NE_NOERROR)
      {
        Vprintf("Final approximate solution\n\n");
        for (j=0; j<n; j++)
          Vprintf("%12.4f%s",x[j], (j%3==2 || j==n-1) ? "\n" : " " );
        exit(EXIT_SUCCESS);
      }
    else
      {
        Vprintf("%s\n", fail.message);
        if (fail.code == NE_TOO_MANY_FUNC_EVAL ||
            fail.code == NE_XTOL_TOO_SMALL ||
            fail.code ==  NE_NO_IMPROVEMENT)
          {
            Vprintf("Approximate solution\n\n");
            for (i=0; i<n;  i++)
              Vprintf("%12.4f%s",x[i], (i%3==2 || i==n-1) ? "\n" : " " );
          }
        exit(EXIT_FAILURE);
      }
  }
  #ifdef NAG_PROTO
  static void f(Integer n, double x[], double fvec[], Integer *userflag, Nag_User *comm)
  #else
       static void f(n,x,fvec,userflag, comm)
       Integer n;
       double x[], fvec[];
       Integer *userflag;
       Nag_User *comm;
  #endif

  {
    Integer k;

    for (k=0; k<n; ++k)
      {
        fvec[k] = (3.0-x[k]*2.0)*x[k]+1.0;
        if (k>0)
          fvec[k] -= x[k-1];
        if (k<n-1)
          fvec[k] -= x[k+1]*2.0;
      }
  }
```

8.2. Program Data

None.

8.3. Program Results

```
c05tbc Example Program Results
Final approximate solution

     -0.5707      -0.6816      -0.7017
     -0.7042      -0.7014      -0.6919
     -0.6658      -0.5960      -0.4164
```