Module 10.2: nag_nlin_eqn

Roots of a Single Nonlinear Equation

`nag_nlin_eqn` provides a procedure for computing a single root of a continuous function in a given interval.

Contents

Procedure: nag_nlin_eqn_sol

1 Description

nag_nlin_eqn_sol attempts to obtain an approximation to a simple root of the continuous function $f(x)$ given an initial interval $[a, b]$ such that $f(a) \times f(b) \leq 0$. This condition implies that there is at least one root of f in the interval $[a, b]$.

The procedure combines the methods of bisection, linear interpolation and linear extrapolation (see Dahlquist and Björck [2]), to find a sequence of subintervals $[a_i, b_i]$ of the initial interval such that the final interval contains the root. The approximation, x, to the root, α, is such that one or both of the following criteria are satisfied:

- $|x - \alpha| <$ x_tol,
- $|f(x)| <$ f_tol.

Since the intervals $[a_i, b_i]$ are determined only so that $f(a_i) \times f(b_i) \leq 0$, it is possible that the final interval may contain a discontinuity or a pole of f (violating the requirement that f be continuous). This procedure also checks if the sign change is likely to correspond to a pole of f.

2 Usage

```
USE nag_nlin_eqn

CALL nag_nlin_eqn_sol(f, x, a, b  [, optional arguments])
```

3 Arguments

3.1 Mandatory Arguments

f — function

f must evaluate the function f at the point x.

```
function f(x)

real(kind=wp), intent(in) ::  x
```
Input: the point x at which the function must be evaluated.

```
real(kind=wp) ::  f
```
Result: the value of the function at the point x.

x — real(kind=*wp*), intent(out)

Output: the approximation to the root.

a — real(kind=*wp*), intent(in)

b — real(kind=*wp*), intent(in)

Input: the lower bound, a, and the upper bound, b, of the interval.

Constraints: a $\neq$ b, f(a)$\times$ f(b) ≤ 0.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

x_tol — real(kind=*wp*), intent(in), optional

Input: the absolute tolerance to which the root is required (see Section 1).

Constraints: `x_tol` > 0.0.

Default: `x_tol` = `SQRT(EPSILON(1.0_wp))`.

f_tol — real(kind=*wp*), intent(in), optional

Input: a value such that if $|f(x)| <$ `f_tol`, `x` is accepted as the root (see Section 6.2).

Constraints: `f_tol` ≥ 0.0.

Default: `f_tol` = 0.0.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
399	An unexpected Library error. Check all procedure calls. If you cannot find any error in your calling program, please report this error to NAG.

Failures (error%level = 2):

error%code	Description
201	$f(x)$ changes sign at a pole. A change in sign of $f(x)$ has been determined as occurring near the point defined by the final value of `x`. There is some evidence that this sign change corresponds to a pole of $f(x)$.

Warnings (error%level = 1):

error%code	Description
101	Too much accuracy requested. `x_tol` is too small for the computer being used. The final value of `x` is an accurate approximation to the root.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

6 Further Comments

6.1 Algorithmic Detail

This procedure is a modified version of procedure ZEROIN given in Bus and Dekker [1].

6.2 Accuracy

This depends on the values of `x_tol` and `f_tol`. If full machine accuracy is required, `x_tol` and `f_tol` may be set very small, resulting in an error exit with `error%code` = 101. This may involve more iterations than are required for a less accurate solution. You should use `x_tol` to control the accuracy (with the default value for `f_tol`) unless you have considerable knowledge of the size of $f(x)$ for values of x near the root.

Example 1: Solution of a single transcendental equation

To calculate the root of $e^{-x} - x$ within the interval [0,1].

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
MODULE nlin_eqn_ex01_mod

  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)

CONTAINS

  FUNCTION f(x)

    ! .. Implicit None Statement ..
    IMPLICIT NONE
    ! .. Intrinsic Functions ..
    INTRINSIC EXP
    ! .. Scalar Arguments ..
    REAL (wp), INTENT (IN) :: x
    ! .. Function Return Value ..
    REAL (wp) :: f
    ! .. Executable Statements ..

    f = EXP(-x) - x

  END FUNCTION f

END MODULE nlin_eqn_ex01_mod

PROGRAM nag_nlin_eqn_ex01

  ! Example Program Text for nag_nlin_eqn
  ! NAG fl90, Release 3. NAG Copyright 1997.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_out
  USE nag_nlin_eqn, ONLY : nag_nlin_eqn_sol
  USE nlin_eqn_ex01_mod, ONLY : f, wp
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Local Scalars ..
  REAL (wp) :: a, b, x
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_nlin_eqn_ex01'

  a = 0.0_wp
  b = 1.0_wp

  ! Find a zero of the function in the interval [a,b]

  CALL nag_nlin_eqn_sol(f,x,a,b)
```

```
    WRITE (nag_std_out,'(/,1X,A,F12.5)') 'Zero = ', x

  END PROGRAM nag_nlin_eqn_ex01
```

2 Program Data

None.

3 Program Results

```
Example Program Results for nag_nlin_eqn_ex01

Zero =      0.56714
```

Additional Examples

Not all example programs supplied with NAG *fl*90 appear in full in this module document. The following additional examples, associated with this module, are available.

`nag_nlin_eqn_ex02`

Solution of a single transcendental equation, with user-supplied `x_tol`.

References

[1] Bus J C P and Dekker T J (1975) Two efficient algorithms with guaranteed convergence for finding a zero of a function *ACM Trans. Math. Software* **1** 330–345

[2] Dahlquist G and Björck Å (1974) *Numerical Methods* Prentice-Hall