Chapter c05 – Roots of One or More Transcendental Equations

1. Scope of the Chapter

This chapter is concerned with the solution of nonlinear equations, including systems of nonlinear equations in several variables. The nonlinear functions which occur in the equations, must be real-valued functions of real variables, and only real-valued solutions will be found. (Complex equations must be expressed in terms of an equivalent larger system of real equations.)

2. Background

The chapter divides naturally into two parts.

2.1. A Single Equation

The first part deals with the real zeros of a real-valued function of a single real variable $f(x)$.

There is one function: nag_zero_cont_func_bd_1 (c05sdc). It assumes that the user can determine an initial interval $[a, b]$ within which the desired zero lies, and outside which all other zeros lie. It also assumes that f changes sign within the interval, that is, $f(a) \times f(b) < 0$. Under these conditions it is guaranteed to converge to the zero.

2.2. Systems of Equations

The functions in the second part of this chapter are designed to solve a system of nonlinear equations in n unknowns:

$$f_i(x) = 0, \quad i = 1, 2, \ldots, n, \quad x = (x_1, x_2, \ldots, x_n)^T.$$

It is assumed that the functions are continuous and differentiable so that the matrix of first partial derivatives of the functions, the *Jacobian* matrix $J_{ij}(x) = \partial f_i / \partial x_j$ evaluated at the point x, exists, though it may not be possible to calculate it directly.

The functions f_i must be independent, otherwise there will be an infinity of solutions and the methods will fail. However, even when the functions are independent, the solution may not be unique. Since the methods are iterative, an initial guess at the solution has to be supplied, and the solution located will usually be the one closest to this initial guess.

Two functions are provided: nag_zero_nonlin_eqns_1 (c05tbc) requires the user to supply a function **f** for calculating the function values f_i only; nag_zero_nonlin_eqns_deriv_1 (c05ubc) requires **f** also to calculate their partial derivatives, in other words the Jacobian matrix J.

It is preferable to use nag_zero_nonlin_eqns_deriv_1 (c05ubc) if possible, since results obtained with the aid of user-supplied derivatives are generally more reliable and are obtained more efficiently; nag_zero_nonlin_eqns_1 (c05tbc) approximates the derivatives by finite differences.

It is essential to check that the user-supplied function **f** is coded correctly. It should be tested separately before being used in conjuntion with nag_zero_nonlin_eqns_1 (c05tbc) or nag_zero_nonlin_eqns_deriv_1 (c05ubc). An additional routine nag_check_deriv_1 (c05zcc) is provided to check whether the derivative values are consistent with the function values.

It is also important to ensure that **f** is coded so as to return accurate results (for example, by avoiding unnecessary cancellation), and to be efficient (for example, by arranging that common subexpressions are evaluated only once); the efficiency of the algorithm can depend strongly on the time spent in **f**.

Scaling of the variables has a considerable effect on the efficiency of the algorithm. Ideally, the problem should be designed so that the components x_i are all of similar magnitude, and all the function values f_i should be of similar magnitude.

The accuracy of the results is usually determined by the accuracy parameters of the functions, but if unreasonable accuracy is demanded, rounding errors may become important and cause a failure.

3. Available Functions

At Mark 5, we have introduced thread-safe versions of the functions requiring communication between the calling program and the user-defined function. This has been implemented by introducing a `void *` parameter both in the calling sequence of the NAG function and the user-defined function. Note that the functionality of the thread-safe version is essentially the same as the corresponding thread-unsafe version. However, it is recommended that the thread-safe version be used in preference to the thread-unsafe versions, particularly in view of the fact that the thread-unsafe version may be removed at a later mark of the C Library.

3.1. Thread-unsafe functions

Zero of a continuous function in a given interval	`c05adc`
Solution of a system of nonlinear functions	`c05nbc`
Solution of a system of nonlinear functions, using derivatives	`c05pbc`
Check user's function for calculating derivatives	`c05zbc`

3.2. Thread-safe functions

Zero of a continuous function in a given interval	`c05sdc`
Solution of a system of nonlinear functions	`c05tbc`
Solution of a system of nonlinear functions, using derivatives	`c05ubc`
Check user's function for calculating derivatives	`c05zcc`