NAG Library Chapter Introduction

c05 – Roots of One or More Transcendental Equations

Contents

1 Scope of the Chapter 2

2 Background to the Problems 2

2.1 A Single Equation 2
2.2 Systems of Equations 2

3 Recommendations on Choice and Use of Available Functions 3

3.1 Zeros of Functions of One Variable 3
3.2 Solution of Sets of Nonlinear Equations 3
3.3 Values of Lambert's W function 5

4 Decision Trees 5

5 Functionality Index 6

6 Auxiliary Functions Associated with Library Function Arguments 6

7 Functions Withdrawn or Scheduled for Withdrawal 7

8 References 7

1 Scope of the Chapter

This chapter is concerned with the calculation of zeros of continuous functions of one or more variables. The majority of problems considered are for real-valued functions of real variables, in which case complex equations must be expressed in terms of the equivalent larger system of real equations.

2 Background to the Problems

The chapter divides naturally into two parts.

2.1 A Single Equation

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

There are three functions with simple calling sequences. The first assumes that you can determine an initial interval $[a, b]$ within which the desired zero lies, (that is, where $f(a) \times f(b) < 0$), and outside which all other zeros lie. The function then systematically subdivides the interval to produce a final interval containing the zero. This final interval has a length bounded by your specified error requirements; the end of the interval where the function has smallest magnitude is returned as the zero. This function is guaranteed to converge to a **simple** zero of the function. (Here we define a simple zero as a zero corresponding to a sign-change of the function; none of the available functions are capable of making any finer distinction.) However, as with the other functions described below, a non-simple zero might be determined and it is left to you to check for this. The algorithm used is due to Brent (1973).

The two other functions are both designed for the case where you are unable to specify an interval containing the simple zero. One starts from an initial point and performs a search for an interval containing a simple zero. If such an interval is computed then the method described above is used next to determine the zero accurately. The other method uses a 'continuation' method based on a secant iteration. A sequence of subproblems is solved; the first of these is trivial and the last is the actual problem of finding a zero of $f(x)$. The intermediate problems employ the solutions of earlier problems to provide initial guesses for the secant iterations used to calculate their solutions.

Three other functions are also supplied. They employ reverse communication and use the same core algorithms as the functions described above.

Finally, two functions are provided to return values of Lambert's W function (sometimes known as the 'product log' or 'Omega' function), which is the inverse function of

$$f(w) = we^w \quad \text{for} \quad w \in C;$$

that is, if Lambert's W function $W(x) = a$ for $x, a \in C$, then a is a zero of the function $F(w) = we^w - x$. One function uses the iterative method described in Barry *et al.* (1995) to return values from the real branches of W (restricting $x, a \in R$). The second function enforces no such restriction, and uses the approach described in Corless *et al.* (1996).

2.2 Systems of Equations

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

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

where T stands for transpose.

It is assumed that the functions are continuous and differentiable so that the matrix of first partial derivatives of the functions, the **Jacobian** matrix $J_{ij}(x) = \left(\dfrac{\partial f_i}{\partial x_j}\right)$ 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 solutions 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.

3 Recommendations on Choice and Use of Available Functions

3.1 Zeros of Functions of One Variable

The functions can be divided into two classes. There are three functions (**nag_interval_zero_cont_func (c05avc)**, **nag_zero_cont_func_cntin_rcomm (c05axc)** and **nag_zero_cont_func_brent_rcomm (c05azc)**) all written in reverse communication form and three (**nag_zero_cont_func_brent_binsrch (c05auc)**, **nag_zero_cont_func_cntin (c05awc)** and **nag_zero_cont_func_brent (c05ayc)**) written in direct communication form (see Section 3.3.2 in How to Use the NAG Library and its Documentation for a description of the difference between these two conventions). The direct communication functions are designed for inexperienced users and, in particular, for solving problems where the function $f(x)$ whose zero is to be calculated, can be coded as a user-supplied (sub)program. These functions find the zero by using the same core algorithms as the reverse communication functions. Experienced users are recommended to use the reverse communication functions directly as they permit you more control of the calculation. Indeed, if the zero-finding process is embedded in a much larger program then the reverse communication functions should always be used.

The recommendation as to which function should be used depends mainly on whether you can supply an interval $[a, b]$ containing the zero; that is, where $f(a) \times f(b) < 0$. If the interval can be supplied, then **nag_zero_cont_func_brent (c05ayc)** (or, in reverse communication, **nag_zero_cont_func_brent_rcomm (c05azc)**) should be used, in general. This recommendation should be qualified in the case when the only interval which can be supplied is very long relative to your error requirements **and** you can also supply a good approximation to the zero. In this case **nag_zero_cont_func_cntin (c05awc)** (or, in reverse communication, **nag_zero_cont_func_cntin_rcomm (c05axc)**) **may** prove more efficient (though these latter functions will not provide the error bound available from **nag_zero_cont_func_brent_rcomm (c05azc)**).

If an interval containing the zero cannot be supplied then you must choose between **nag_zero_cont_func_brent_binsrch (c05auc)** (or, in reverse communication, **nag_interval_zero_cont_func (c05avc)** followed by **nag_zero_cont_func_brent_rcomm (c05azc)**) and **nag_zero_cont_func_cntin (c05awc)** (or, in reverse communication, **nag_zero_cont_func_cntin_rcomm (c05axc)**). **nag_zero_cont_func_brent_binsrch (c05auc)** first determines an interval containing the zero, and then proceeds as in **nag_zero_cont_func_brent (c05ayc)**; it is particularly recommended when you do not have a good initial approximation to the zero. If a good initial approximation to the zero is available then **nag_zero_cont_func_cntin (c05awc)** is to be preferred. Since neither of these latter functions has guaranteed convergence to the zero, you are recommended to experiment with both in case of difficulty.

3.2 Solution of Sets of Nonlinear Equations

The solution of a set of nonlinear equations

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

can be regarded as a special case of the problem of finding a minimum of a sum of squares

$$s(x) = \sum_{i=1}^{m} [f_i(x_1, x_2, \ldots, x_n)]^2, \quad (m \geq n). \tag{3}$$

So the functions in Chapter e04 are relevant as well as the special nonlinear equations functions.

The functions for solving a set of nonlinear equations can also be divided into classes. There are five functions (**nag_zero_nonlin_eqns_easy (c05qbc)**, **nag_zero_nonlin_eqns_expert (c05qcc)**, **nag_zero_sparse_nonlin_eqns_easy (c05qsc)**, **nag_zero_nonlin_eqns_deriv_easy (c05rbc)** and **nag_zero_nonlin_eqns_deriv_expert (c05rcc)**) all written in direct communication form and three (**nag_zero_nonlin_eqns_aa_rcomm (c05mdc)**, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)**) written in reverse communication form. The direct communication functions are designed for inexperienced users and, in particular, these functions require the f_i (and possibly their derivatives) to be calculated in user-supplied functions. These should be set up carefully so the Library functions can work as efficiently as possible. Experienced users are recommended to use the reverse communication functions as they permit you more control of the calculation. Indeed, if the zero-finding process is embedded in a much larger program then the reverse communication functions should always be used.

The main decision you have to make is whether to supply the derivatives $\dfrac{\partial f_i}{\partial x_j}$. It is advisable to do so if possible, since the results obtained by algorithms which use derivatives are generally more reliable than those obtained by algorithms which do not use derivatives.

nag_zero_nonlin_eqns_deriv_easy (c05rbc), **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)** require you to provide the derivatives, whilst **nag_zero_nonlin_eqns_aa_rcomm (c05mdc)**, **nag_zero_nonlin_eqns_easy (c05qbc)**, **nag_zero_nonlin_eqns_expert (c05qcc)**, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_sparse_nonlin_eqns_easy (c05qsc)** do not. **nag_zero_nonlin_eqns_easy (c05qbc)**, **nag_zero_sparse_nonlin_eqns_easy (c05qsc)** and **nag_zero_nonlin_eqns_deriv_easy (c05rbc)** are easy-to-use functions; greater flexibility may be obtained using **nag_zero_nonlin_eqns_expert (c05qcc)** and **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** (or, in reverse communication, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)**), but these have longer argument lists. **nag_zero_nonlin_eqns_easy (c05qbc)**, **nag_zero_nonlin_eqns_expert (c05qcc)**, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_sparse_nonlin_eqns_easy (c05qsc)** approximate the derivatives internally using finite differences. **nag_zero_nonlin_eqns_aa_rcomm (c05mdc)** does not use derivatives at all, and may be useful when the cost of evaluating $f(x)$ is high.

nag_zero_sparse_nonlin_eqns_easy (c05qsc) is an easy-to-use function specially adapted for sparse problems, that is, problems where each function depends on a small subset of the n variables so that the Jacobian matrix has many zeros. It employs sparse linear algebra methods and consequently is expected to take significantly less time to complete than the other functions, especially if n is large.

nag_check_derivs (c05zdc) is provided for use in conjunction with **nag_zero_nonlin_eqns_deriv_easy (c05rbc)**, **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)** to check the user-supplied derivatives for consistency with the functions themselves. You are strongly advised to make use of this function whenever **nag_zero_nonlin_eqns_deriv_easy (c05rbc)**, **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** or **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)** is used.

Firstly, the calculation of the functions and their derivatives should be ordered so that **cancellation errors** are avoided. This is particularly important in a function that uses these quantities to build up estimates of higher derivatives.

Secondly, **scaling** of the variables has a considerable effect on the efficiency of a function. The problem should be designed so that the elements of x are of similar magnitude. The same comment applies to the functions, i.e., all the f_i should be of comparable size.

The accuracy is usually determined by the accuracy arguments of the functions, but the following points may be useful.

(i) Greater accuracy in the solution may be requested by choosing smaller input values for the accuracy arguments. However, if unreasonable accuracy is demanded, rounding errors may become important and cause a failure.

(ii) Some idea of the accuracies of the x_i may be obtained by monitoring the progress of the function to see how many figures remain unchanged during the last few iterations.

(iii) An approximation to the error in the solution x is given by e where e is the solution to the set of linear equations

$$J(x)e = -f(x)$$

where $f(x) = (f_1(x), f_2(x), \ldots, f_n(x))^{\mathrm{T}}$.

Note that the QR decomposition of J is available from **nag_zero_nonlin_eqns_expert (c05qcc)** and **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** (or, in reverse communication, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)**) so that

$$Re = -Q^{\mathrm{T}} f$$

and $Q^{\mathrm{T}} f$ is also provided by these functions.

(iv) If the functions $f_i(x)$ are changed by small amounts ϵ_i, for $i = 1, 2, \ldots, n$, then the corresponding change in the solution x is given approximately by σ, where σ is the solution of the set of linear equations

$$J(x)\sigma = -\epsilon.$$

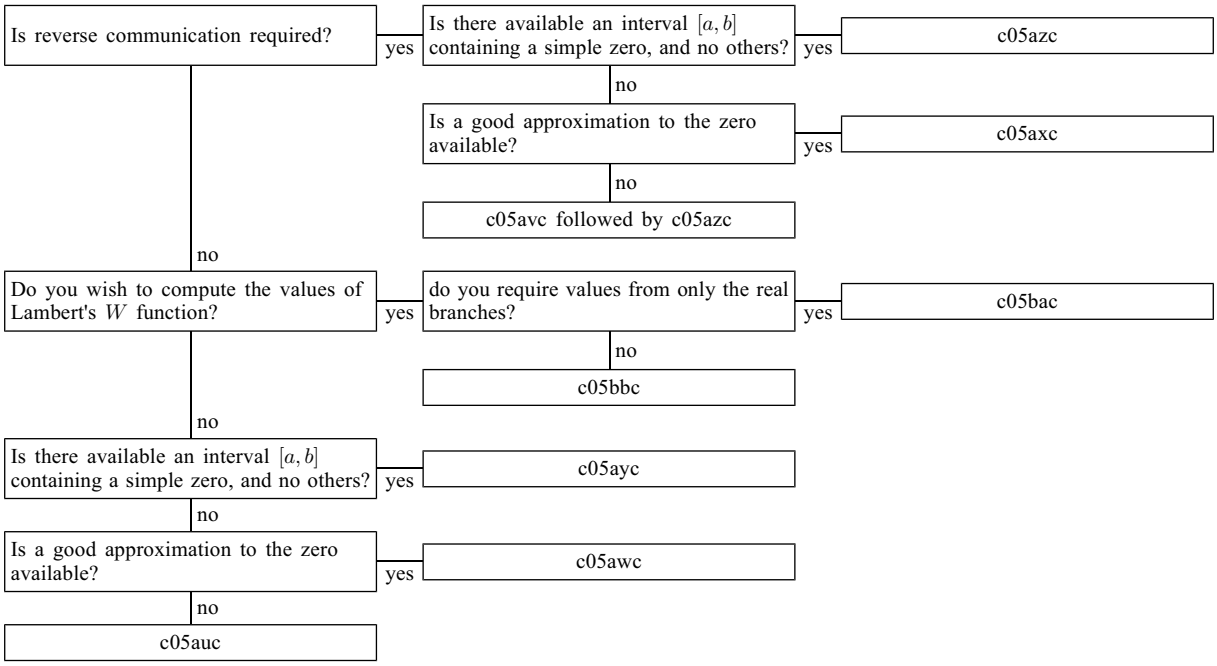
Thus one can estimate the sensitivity of x to any uncertainties in the specification of $f_i(x)$, for $i = 1, 2, \ldots, n$. As noted above, the sophisticated functions **nag_zero_nonlin_eqns_expert (c05qcc)** and **nag_zero_nonlin_eqns_deriv_expert (c05rcc)** (or, in reverse communication, **nag_zero_nonlin_eqns_rcomm (c05qdc)** and **nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)**) provide the QR decomposition of J.

3.3 Values of Lambert's *W* function

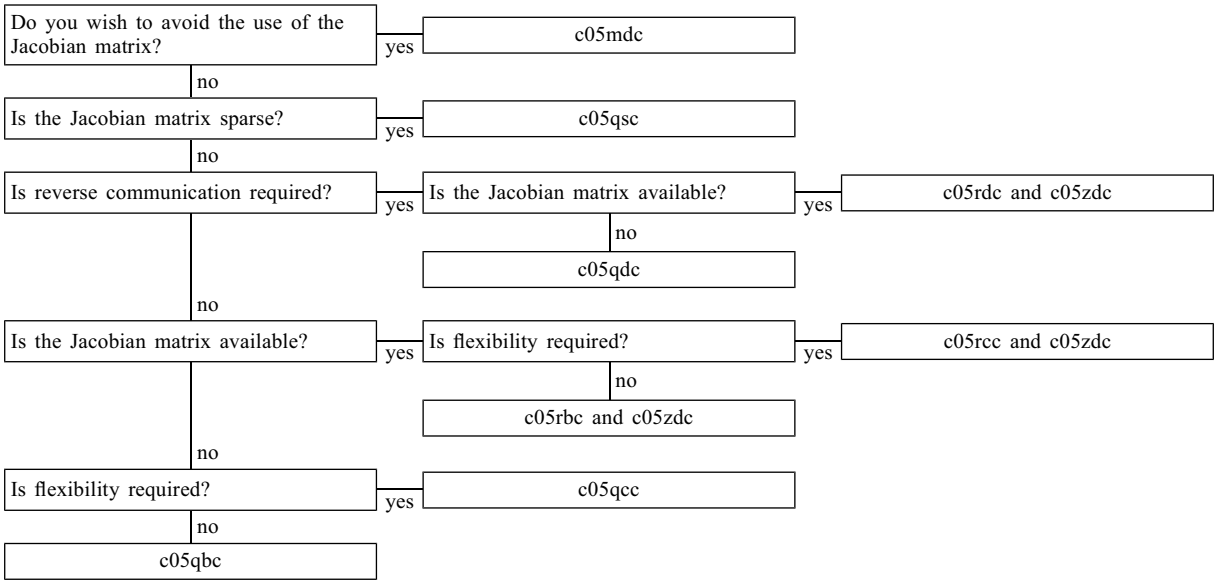
If you require purely-real values of W, these will be evaluated marginally more efficiently by **nag_lambertW (c05bac)** than **nag_lambertW_complex (c05bbc)** owing to the differing iterative procedures used by each function.

4 Decision Trees

Tree 1: Functions of One Variable



Tree 2: Functions of several variables



5 Functionality Index

Lambert's W function,
- complex values ..... nag_lambertW_complex (c05bbc)
- real values ..... nag_lambertW (c05bac)

Zeros of functions of one variable,
- direct communication,
  - binary search followed by Brent algorithm ..... nag_zero_cont_func_brent_binsrch (c05auc)
  - Brent algorithm ..... nag_zero_cont_func_brent (c05ayc)
  - continuation method ..... nag_zero_cont_func_cntin (c05awc)
- reverse communication,
  - binary search ..... nag_interval_zero_cont_func (c05avc)
  - Brent algorithm ..... nag_zero_cont_func_brent_rcomm (c05azc)
  - continuation method ..... nag_zero_cont_func_cntin_rcomm (c05axc)

Zeros of functions of several variables,
- checking function,
  - checks user-supplied Jacobian ..... nag_check_derivs (c05zdc)
- direct communication,
  - Anderson acceleration,
    - reverse communication ..... nag_zero_nonlin_eqns_aa_rcomm (c05mdc)
  - easy-to-use,
    - derivatives required ..... nag_zero_nonlin_eqns_deriv_easy (c05rbc)
    - no derivatives required ..... nag_zero_nonlin_eqns_easy (c05qbc)
    - no derivatives required, sparse ..... nag_zero_sparse_nonlin_eqns_easy (c05qsc)
  - sophisticated,
    - derivatives required ..... nag_zero_nonlin_eqns_deriv_expert (c05rcc)
    - no derivatives required ..... nag_zero_nonlin_eqns_expert (c05qcc)
- reverse communication,
  - sophisticated,
    - derivatives required ..... nag_zero_nonlin_eqns_deriv_rcomm (c05rdc)
    - no derivatives required ..... nag_zero_nonlin_eqns_rcomm (c05qdc)

6 Auxiliary Functions Associated with Library Function Arguments

None.

7 Functions Withdrawn or Scheduled for Withdrawal

The following lists all those functions that have been withdrawn since Mark 23 of the Library or are scheduled for withdrawal at one of the next two marks.

Withdrawn Function	Mark of Withdrawal	Replacement Function(s)
nag_zero_cont_func_bd (c05adc)	24	nag_zero_cont_func_brent (c05ayc)
nag_zero_cont_func_brent_bsrch (c05agc)	25	nag_zero_cont_func_brent_binsrch (c05auc)
nag_zero_nonlin_eqns (c05nbc)	24	nag_zero_nonlin_eqns_easy (c05qbc)
nag_zero_nonlin_eqns_deriv (c05pbc)	24	nag_zero_nonlin_eqns_deriv_easy (c05rbc)
nag_zero_cont_func_bd_1 (c05sdc)	25	nag_zero_cont_func_brent (c05ayc)
nag_zero_nonlin_eqns_1 (c05tbc)	24	nag_zero_nonlin_eqns_easy (c05qbc)
nag_zero_nonlin_eqns_deriv_1 (c05ubc)	25	nag_zero_nonlin_eqns_deriv_easy (c05rbc)
nag_check_deriv (c05zbc)	24	nag_check_derivs (c05zdc)
nag_check_deriv_1 (c05zcc)	24	nag_check_derivs (c05zdc)

8 References

Anderson D G (1965) Iterative Procedures for Nonlinear Integral Equations *J. Assoc. Comput. Mach.* **12** 547–560

Barry D J, Culligan–Hensley P J, and Barry S J (1995) Real values of the W-function *ACM Trans. Math. Software* **21(2)** 161–171

Brent R P (1973) *Algorithms for Minimization Without Derivatives* Prentice–Hall

Corless R M, Gonnet G H, Hare D E G, Jeffrey D J and Knuth D E (1996) On the Lambert W function *Advances in Comp. Math.* **3** 329–359

Gill P E and Murray W (1976) Algorithms for the solution of the nonlinear least squares problem *Report NAC 71* National Physical Laboratory

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

Ortega J M and Rheinboldt W C (1970) *Iterative Solution of Nonlinear Equations in Several Variables* Academic Press

Rabinowitz P (1970) *Numerical Methods for Nonlinear Algebraic Equations* Gordon and Breach