LECTURES

LESSON VI

6. Nonlinear Equations and Systems

6.1. Nonlinear Equations

6.1.0. Introduction

We consider that most basic of tasks, solving equations numerically. While most equations are born with both a right-hand side and a left-hand side, one traditionally moves all terms to the left, leaving

$$f(x) = 0 \tag{6.1.0.1}$$

whose solution or solutions are desired. When there is only one independent variable, the problem is one-dimensional, namely to find the root or roots of a function. With more than one independent variable, more than one equation can be satisfied simultaneously. You likely once learned the implicit function theorem which (in this context) gives us the hope of satisfying n equations in n unknowns simultaneously. Note that we have only hope, not certainty. A nonlinear set of equations may have no (real) solutions at all. Contrariwise, it may have more than one solution. The implicit function theorem tells us that generically the solutions will be distinct, pointlike, and separated from each other. But, because of nongeneric, i.e., degenerate, case, one can get a continuous family of solutions. In vector notation, we want to find one or more n-dimensional solution vectors $\vec{x}$ such that

$$\vec{f}(\vec{x}) = \vec{0} \tag{6.1.0.2}$$

where $\vec{f}$ is the n-dimensional vector-valued function whose components are the individual equations to be satisfied simultaneously. Simultaneous solution of equations in n dimensions is much more difficult than finding roots in the one-dimensional case. The principal difference between one and many dimensions is that, in one dimension, it is possible to bracket or "trap" a root between bracketing values, and then find it out directly. In multidimensions, you can never be sure that the root is there at all until you have found it. Except in linear problems, root finding invariably proceeds by iteration, and this is equally true in one or in many dimensions. Starting from some approximate trial solution, a useful algorithm will improve the solution until some predetermined convergence criterion is satisfied. For smoothly varying functions, good algorithms will always converge, provided that the initial guess is good enough. Indeed one can even determine in advance the rate of convergence of most algorithms. It cannot be overemphasized, however, how crucially success depends on having a good first guess for the

solution, especially for multidimensional problems. This crucial beginning usually depends on analysis rather than numerics. Carefully crafted initial estimates reward you not only with reduced computational effort, but also with understanding and increased self-esteem. Hammings motto, "the purpose of computing is insight, not numbers," is particularly apt in the area of finding roots. One should repeat this motto aloud whenever program converges, with ten-digit accuracy, to the wrong root of a problem, or whenever it fails to converge because there is actually no root, or because there is a root but initial estimate was not sufficiently close to it. For one-dimensional root finding, it is possible to give some straightforward answers: You should try to get some idea of what your function looks like before trying to find its roots. If you need to mass-produce roots for many different functions, then you should at least know what some typical members of the ensemble look like. Next, you should always bracket a root, that is, know that the function changes sign in an identified interval, before trying to converge to the roots value. Finally, one should never let iteration method get outside of the best bracketing bounds obtained at any stage. We can see that some pedagogically important algorithms, such as secant method or Newton-Raphson, can violate this last constraint, and are thus not recommended unless certain fixups are implemented. Multiple roots, or very close roots, are a real problem, especially if the multiplicity is an even number. In that case, there may be no readily apparent sign change in the function, so the notion of bracketing a root and maintaining the bracket becomes difficult. We nevertheless insist on bracketing a root, even if it takes the minimum-searching techniques to determine whether a tantalizing dip in the function really does cross zero or not. As usual, we want to discourage the reader from using routines as black boxes without understanding them.

There are two distinct phases in finding the roots of nonlinear equation (see [2], pp. 130-135):

(1) Bounding the solution, and
(2) Refining the solution.

In general, nonlinear equations can behave in many different ways in the vicinity of a root.

(1) Bounding the solution

Bounding the solution involves finding a rough estimate of the solution that can be used as the initial approximation, or the starting point, in a systematic procedure that refines the solution to a specified tolerance in an efficient manner. If possible, it is desirable to bracket the root between two points at which the value of the nonlinear function has opposite signs. The bounding procedures can be:

1. Drafting the function,
2. Incremental search,
3. Previous experience or similar problem,
4. Solution of a simplified approximate model.

Drafting the function involves plotting the nonlinear function over the range of interest. Spreadsheets generally have graphing capabilities, as does `Mathematica`, `Matlab` and `Mathcad`. The resolution of the plots is generally not precise enough for accurate result. However, they are accurate enough to bound the solution. The plot of the nonlinear function displays the behavior of nonlinear equation and gives view of scope of problem.

An incremental search is conducted by starting at one end of the region of interest and evaluating the nonlinear function with small increments across the region. When the value of the function changes the sign, it is assumed that a root lies in that interval. Two end points of the interval containing the root can be used as initial guesses for a

refining method (second phase of solution). If multiple roots are suspected, one has to check for sigh changes in the derivative of the function between the ends of the interval.

(2) Refining the solution

Refining the solution involves determining the solution to a specified tolerance by an efficient procedure. The basic methods for refining the solution are:

2.1 Trial and error,

2.2 Closed domain methods (bracketing method),

2.3 Open domain methods.

Trial and error methods simply presume (guess) the root, $x = \alpha$, evaluate $f(\alpha)$, and compare to zero. If $f(\alpha)$ is close enough to zero, quit, if not guess another α and continue until $f(\alpha)$ is close enough to zero.

Closed domain (bracketing methods) are methods that start with two values of x which bracket the root, $x = \alpha$, and systematically reduce the interval, keeping root inside of brackets (inside of interval). Two most popular methods of that kind are:

2.2.1 Interval halving (bisection),

2.2.2 False position (Regula Falsi).

Bracketing methods are robust and reliable, since root is always inside of closed interval, but can be slow to convergence.

Open domain methods do not restrict the root to remain trapped in a closed interval. Therefore, there are not as robust as bracketing methods and can diverge. But, they use information about the nonlinear function itself to come closer with estimation of the root. Thus, they are much more efficient than bracketing methods.

Some general hints for root finding

Nonlinear equations can behave in various ways in the vicinity of a root. Algebraic and transcendental equations may have simple real roots, multiple real roots, or complex roots. Polynomials may have real or complex roots. If the polynomial coefficients are all real, complex root occur in conjugate pairs. If the polynomial coefficients are complex, single complex roots can occur.

There are numerous methods for finding the roots of a nonlinear equation. Some general philosophy of root finding is given below.

1. Bounding method should bracket a root, if possible.
2. Good initial approximations are extremely important.
3. Closed domain methods are more robust than open domain methods because they keep the root in a closed interval.
4. Open domain methods, when converge, in the general case converge faster than closed domain methods.
5. For smoothly varying functions, most algorithms will always converge if the initial approximation is close enough. The rate of convergence of most algorithms can be determined in advance.
6. Many problems in engineering and science are well behaved and straightforward. In such cases, a straightforward open domain method, such as Newton's method, or the secant method, can be applied without worrying about special cases and strange behavior. If problems arise during the solution, then the peculiarities of the nonlinear equation and the choice of solution method can be reevaluated.
7. When a problem is to be solved only once or a few times, then the efficiency of method is not of major concern. However, when a problem is to be solved many times, efficiency is of major concern.
8. Polynomials can be solved by any of the methods for solving nonlinear equations. However, the special techniques applicable to polynomials should be considered.

9. If a nonlinear equation has complex roots, that has to be anticipated when choosing a method.
10. Time for problem analysis versus computer time has to be considered during method selection.
11. Generalizations about root-finding methods are generally not possible.

The root-finding algorithms should contain the following features:

1. An upper limit on the number of iterations.
2. If the method uses the derivative $f'(x)$, it should be monitored to ensure that it does not approach zero.
3. A convergence test for the change in the magnitude of the solution, $|x_{i+1} - x_i|$, or the magnitude of the nonlinear function, $|x_{i+1}|$, has to be included.
4. When convergence is indicated, the final root estimate should be inserted into the nonlinear function $f(x)$ to guarantee that $f(x) = 0$ within the desired tolerance.

6.1.1. Newton's method

Newton's or often called Newton-Raphson method is basic method for determination of isolated zeros of nonlinear equations.

Let isolated unique simple root $x = a$ of equation (6.1.0.1) exist on segment $[\alpha, \beta]$ and let $f \in \mathcal{C}[\alpha, \beta]$. Then, using Taylor development, we get

$$f(a) = f(x_0) + f'(x_0)(a - x_0) + \mathbf{O}\big((a - x_0)^2\big), \tag{6.1.1.1}$$

where $\xi = x_0 + \theta(a - x_0) \quad (0 < \theta < 1)$. Having in mind that $f(a) = 0$, by neglecting last member on the right-hand side of (6.1.1.1), we get

$$a \cong x_0 - \frac{f(x_0)}{f'(x_0)}.$$

If we denote left-hand side of last approximative equation with x_1, we get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{6.1.1.2}$$

Here x_1 represents the abscissa of intersection of tangent on the curve $y = f(x)$ in the point $(x_0, f(x_0))$ with $x-$axis (see Figure 6.1.1.1).

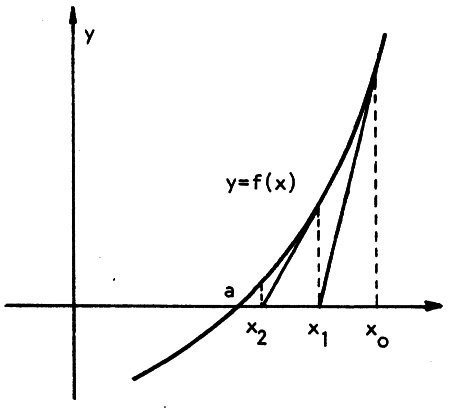

Figure 6.1.1.1

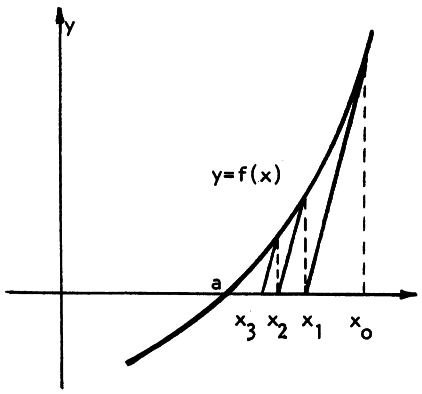

Figure 6.1.1.2

The equality (6.1.1.2) suggests the construction of iterative formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (k = 0, 1, \ldots). \tag{6.1.1.3}$$

known as Newton's method or tangent method.

We can examine the convergence of iterative process (6.1.1.3) by introducing the additional assumption for function f, namely, assume that $f \in \mathcal{C}^2[\alpha, \beta]$. Because the iterative function ϕ is at Newton's method given as

$$\phi(x) = x - \frac{f(x)}{f'(x)},$$

by differentiation we get

$$\phi'(x) = 1 - \frac{f'(x^2) - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}. \tag{6.1.1.4}$$

Note that $\phi(a) = a$ and $\phi'(a) = 0$. Being, based on accepted assumptions for f, function ϕ' continuous on $[\alpha, \beta]$, and $\phi'(a) = 0$, there exists a neighborhood of point $x = a$, denoted as $U(a)$ where it holds

$$|\phi'(x)| = \left| \frac{f(x)f''(x)}{f'(x)^2} \right| \leq q < 1. \tag{6.1.1.5}$$

Theorem 6.1.1.1. *If $x_0 \in U(a)$, series $\{x_k\}$ generated using (6.1.1.3) converges to point $x = a$, whereby*

$$\lim_{k \to +\infty} \frac{x_{k+1} - a}{(x_k - a)^2} = \frac{f''(a)}{2f'(a)}. \tag{6.1.1.6}$$

(see [1], pp. 340-341).

Example 6.1.1.1.

Find the solution of equation

$$f(x) = x - \cos x = 0$$

on segment $[0, \pi/2]$ using Newton's method

$$x_{k+1} = x_k - \frac{x_k - \cos x_k}{1 + \sin x_k} = \frac{x_k \sin x_k + \cos x_k}{1 + \sin x_k} \quad (k = 0, 1, \ldots).$$

Note that $f'(x) = 1 + \sin x > 0 (\forall x \in [0, \pi/2])$. Starting with $x_0 = 1$, we get the results given in Table 6.1.1.

Table 6.1.1

k	x_k
0	1.000000
1	0.750364
2	0.739133
3	0.739085
4	0.739085

The last two iterations give solution of equation in consideration with six exact figures.

Example 6.1.1.2.

By applying the Newton's method on solution of equation $f(x) = x^n - a = 0$ $(a > 0, n > 1)$ we obtain the iterative formula for determination of n-th root of positive number a

$$x_{k+1} = x_k - \frac{x_k^n - a}{n x_k^{n-1}} = \frac{1}{n}\left\{(n-1)x_k + \frac{a}{x_k^{n-1}}\right\} \quad (k = 0, 1, \ldots).$$

A special case of this formula, for $n = 2$ gives as a result square root.

At application of Newton method it is often problem how to chose initial value of x_0 in order series $\{x_k\}_{k\in\mathcal{N}}$ to be monotonous. One answer to this question was given by Fourier. Namely, if f'' does not change a sign on $[\alpha, \beta]$ and if x_0 is chosen in such a way that $f(x_0)f''(x_0) > 0$, the series $\{x_k\}_{k\in\mathcal{N}}$ will be monotonous. This statement follows from (6.1.1.4).

Based on Theorem 6.1.1.1 we conclude that Newton's method applied to determination of simple root $x = a$ has square convergence if $f''(a) \neq 0$. In this case factor of convergence (asymptotic constant of error) is

$$C_2 = |\frac{f''(a)}{2f'(a)}|.$$

The case $f''(a)$ is specially to be analyzed. Namely, if we suppose that $f \in C^3[\alpha, \beta]$ one can prove that

$$\lim_{k\to+\infty} \frac{x_{k+1} - a}{(x_k - a)^3} = \frac{f'''(a)}{3f'(a)}.$$

Example 6.1.1.3.

Consider the equation

$$f(x) = x^3 - 3x^2 + 4x - 2 = 0.$$

Because of $f(0) = -2$ and $f(1.5) = 0.625$ we conclude that on segment $[0, 1.5]$ this equation has a root. On the other hand, $f'(x) = 3x^2 - 6x + 4 = 3(x - 1)^2 + 1 > 0$, what means that the root is simple, enabling application of Newton's method. Starting with $x_0 = 1.5$, we get the results in Table 6.1.2.

Table 6.1.2

k	x_k
0	1.5000000
1	1.1428571
2	1.0054944
3	1.0000003

The exact value for root is $a = 1$, because $f(x) = (x - 1)^3 + (x - 1)$.

In order to reduce number of calculations, it is often used the following modification of Newton method

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)} \quad (k = 0, 1, \ldots).$$

Geometrically, x_{k+1} represents abscissa of intersection of x-axes and straight line passing through point $(x_k, f(x_k))$ and being parallel to tangent of curve $y = f(x)$ in the point $(x_0, f(x_0))$ (see Figure 6.1.1.2).

Iterative function of such modified Newton's method is

$$\phi_1(x) = x - \frac{f(x)}{f'(x_0)}.$$

Because of $\phi_1(a) = a$ and $\phi_1'(a) = 1 - \frac{f'(a)}{f'(x_0)}$, we conclude that method has order of convergence one, i.e. it holds

$$x_{k+1} - a \sim \big(1 - \frac{f'(a)}{f'(x_0)}\big)(x_k - a) \quad (k \to +\infty),$$

whereby the condition

$$\left|1-\frac{f'(a)}{f'(x_0)}\right| \le q < 1,$$

is analogous to condition (6.1.1.5)

Newton's method can be considered as special case of so known generalized Newton's method

$$x_{k+1} = x_k - \frac{\psi(x_k)f(x_k)}{\psi'(x_k)f(x_k)+\psi(x_k)f'(x_k)} \qquad (k=0,1,\ldots), \tag{6.1.1.8}$$

where ψ is given differentiable function.

For $\psi(x)=1$, (6.1.1.8) reduces to to Newton's method (6.1.1.3).

For $\psi(x)=x^p$, where p parameter, from (6.1.1.8) it follows the formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)+\frac{p}{x_k}f(x_k)} \qquad (k=0,1,\ldots),$$

i.e.

$$x_{k+1} = x_k\left\{1-\frac{f(x_k)}{x_kf'(x_k)+pf(x_k)}\right\} \quad (k=0,1,\ldots). \tag{6.1.1.9}$$

A special case of this method, for $p=1-n$ is known as method of Tihonov ([7]), for the case when f is algebraic polynomial of degree n.

The following modification of Newton's method, containing successive application of formulas

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad x_{k+1} = y_k - \frac{f(y_k)}{f'(x_k)} \qquad (k=0,1,\ldots). \tag{6.1.1.10}$$

is rather often applied.

Similar to proof of Theorem 6.1.1.1, the following standings

$$\lim_{k\to+\infty}\frac{x_{k+1}-a}{(y_k-a)(x_k-a)} = \frac{f''(a)}{f'(a)}$$

and

$$\lim_{k\to+\infty}\frac{x_{k+1}-a}{(x_k-a)^3} = \frac{1}{2}\Big(\frac{f''(a)}{f'(a)}\Big)^2,$$

where $\lim\limits_{k\to+\infty} x_k = a$ can be proved. Thus, the iterative process defined by formulas (6.1.1.10) has a cubic convergence.

6.1.2. Newton's method for multiple zeros

Consider the equation $f(x)=0$, which has in $[\alpha,\beta]$ a multiple root $x=a$ of multiplicity m (≥ 2). Suppose that $f\in C^{m+1}[\alpha,\beta]$, so that

$$f(a)=f'(a)=\ldots=f^{(m-1)}(a)=0, \quad f^{(m)}(a)\ne 0.$$

Namely, in this case f can be given in form

$$f(x)=(x-a)^m g(x) \tag{6.1.2.1}$$

where $g\in C^{m+1}[\alpha,\beta]$ and $g(a)\ne 0$.

From (6.1.2.1) it follows

$$f'(x)=m(x-a)^{m-1}g(x)+(x-a)^m g'(x)$$

and

$$\Delta(x) = \frac{f(x)}{f'(x)} = \frac{(x-a)g(x)}{mg(x) + (x-a)g'(x)} \qquad (x \neq a).$$

If we put $\Delta(a) \stackrel{\text{def}}{=} \lim_{x \to a} \Delta(x)$, then $\Delta(a) = 0$.

The iteration function of Newton's method, applied to determination of multiple root, based on previous, becomes

$$\Phi(x) = x - \frac{(x-a)g(x)}{mg(x) + (x-a)g'(x)}.$$

Because of $\Phi(a) = a,\ \Phi'(a) = 1 - \frac{1}{m},\quad \frac{1}{2} \leq \Phi'(a) < 1 \quad (m \geq 2)$ and Φ' being continuous function, it follows that there exists neighborhood of root $x = a$ in which $|\Phi'(x)| \leq q < 1$, wherefrom we conclude that Newton's method in this case is also convergent, but with degree of convergence 1.

If we know in advance order of multiplicity of root, then Newton method can be modify in such a way that it has order of convergence 2. Namely, one should

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)} \qquad (k = 0, 1, \ldots). \tag{6.1.2.2}$$

Remark 6.1.2.1.

Formally, the formula (6.1.2.2) is Newton's method applied to solving the equation

$$F(x) = \sqrt[m]{f(x)} = 0.$$

Theorem 6.1.2.1. *If x_0 is chosen enough close to root $x = a$ with order of multiplicity m, then series $\{x_k\}_{k \in N_0}$ defined by* (6.1.2.2) *converges to a, whereby*

$$\frac{x_{k+1} - a}{(x_k - a)^2} \sim \frac{1}{m(m+1)} \cdot \frac{f^{(m+1)}(a)}{f^{(m)}(a)} \qquad (k \to +\infty).$$

Proof of this theorem is to be found, for example, in [8].

In the case when the order of multiplicity m is unknown, then in spite of equation $f(x) = 0$ one can solve the equation $\frac{f(x)}{f'(x)} = 0$, which all roots are simple. Newton's method applied to this equation gives the formula

$$x_{k+1} = x_k - \left[\frac{\frac{f(x)}{f'(x)}}{\left(\frac{f(x)}{f'(x)}\right)'}\right]_{x=x_k} \qquad (k = 0, 1, \ldots),$$

i.e. formula

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - f(x_k)f''(x_k)} \qquad (k = 0, 1, \ldots),$$

with order of convergence 2. Note that this function could be obtained from (6.1.1.8) by taking $\Psi(x) = 1/f'(x)$.

6.1.3. Secant method

By approximation of first derivative $f'(x_k)$ in Newton's method with divided difference $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$ one gets secant method

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k) \qquad (k = 1, 2, \ldots), \tag{6.1.3.1}$$

which belongs to open domains methods (two steps method). For starting of iterative process 6.1.3.1) two initial values x_0 and x_1 are needed. Geometrical interpretation of secant method is given in Figure 6.1.3.1.

Let in segment $[\alpha, \beta]$ exist unique root $x = a$ of equation $f(x) = 0$. For examination of convergence of iterative process (6.1.3.1) suppose that $f \in C^2[\alpha, \beta]$ and $f'(x) \neq 0 \ \ (\forall x \in [\alpha, \beta]$.

If we put $e_k = x_k - a \ \ (k = 0, 1, \ldots)$, from (6.1.3.1) it follows

$$e_{k+1} = e_k - \frac{e_k - e_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k). \tag{6.1.3.2}$$

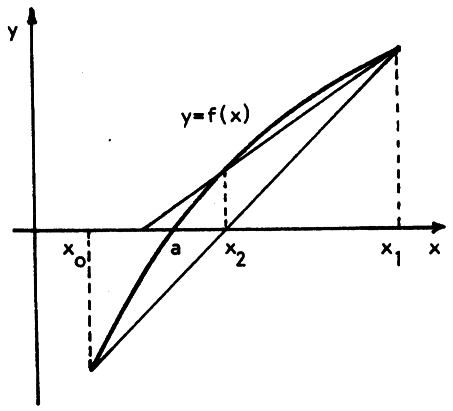

Figure 6.1.3.1

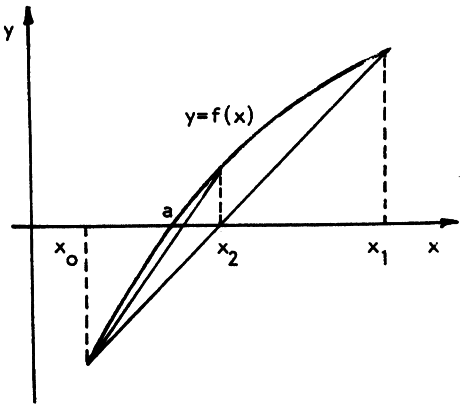

Figure 6.1.3.2

Being

$$f(x_k) = f'(a)\, e_k + \frac{1}{2} f''(a) e_k^2 + \mathbf{O}(e_k^3)$$

and

$$\frac{f(x_k) - f(x_{k-1})}{e_k - e_{k-1}} = f'(a) + \frac{1}{2}(e_k + e_{k-1}) f''(a) + \mathbf{O}(e_{k-1}^2),$$

by replacing in (6.1.3.2) we get

$$e_{k+1} = e_k \left(1 - \frac{f'(a) + \frac{1}{2} e_k f''(a) + \mathbf{O}(e_k^2)}{f'(a) + \frac{1}{2}(e_k + e_{k-1}) f''(a) + \mathbf{O}(e_{k-1}^2)} \right),$$

wherefrom it follows

$$e_{k+1} = e_k \left[1 - \left(1 + \frac{1}{2} e_k \frac{f''(a)}{f'(a)} + \mathbf{O}(e_k^2) \right) \left((1 - \frac{1}{2}(e_k + e_{k-1}) \frac{f''(a)}{f'(a)} + \mathbf{O}(e_{k-1}^2) \right) \right],$$

i.e.

$$e_{k+1} = e_k\, e_{k-1} \frac{f''(a)}{2f'(a)} (1 + \mathbf{O}(e_{k-1})). \tag{6.1.3.3}$$

In order to determine order of convergence and convergence factor, we put

$$|e_{k+1}| = C_r |e_k|^r \, |1 + \mathbf{O}(e_k)|. \tag{6.1.3.4}$$

Then, based on (6.1.3.3) and (6.1.3.4) we get

$$\begin{aligned} |e_{k+1}| &= C_r \, |e_k|^r \, |1 + \mathbf{O}(e_k)| = C_r (C_r |e_{k-1}|^r)^r \, |1 + \mathbf{O}(e_k)| \\ &= C_r \, |e_{k-1}|^r \, |e_{k-1}| \cdot \left| \frac{f''(a)}{2f'(a)} \right| \cdot |1 + \mathbf{O}(e_{k-1})|, \end{aligned}$$

wherefrom it follows

$$r^2 - r - 1 = 0 \quad \text{and} \quad C_r = \left| \frac{f''(a)}{2f'(a)} \right|^{1/r}.$$

Order of convergence r we get as a positive solution of obtained quadratic equation, i.e. $r = \frac{1}{2}(1 + \sqrt{5}) \cong 1.62$. Factor of convergence is

$$C_r = \left| \frac{f''(a)}{2f'(a)} \right|^{(\sqrt{5}-1)/2}.$$

Remark 6.1.3.1.

For the solution of equation of form

$$x = g(x) \tag{6.1.3.5}$$

one can find in bibliography Weigstein's method ([9]), where, starting from x_0 the series $\{x_k\}_{k \in N}$ is to be generated using

$$\begin{aligned} x_1 &= g(x_0) \\ x_{k+1} &= g(x_k) - \frac{(g(x_k) - g(x_{k-1}))(g(x_k) - x_k)}{(g(x_k) - g(x_{k-1})) - (x_k - x_{k-1})} \quad (k = 1, 2, \ldots) \end{aligned} \tag{6.1.3.6}$$

It can be shown that this method is actually secant method with initial values x_0 and $x_1 = g(x_0)$. Namely, if we present equation (6.1.3.5) in form

$$f(x) = g(x) - x = 0, \tag{6.1.3.7}$$

by replacing (6.1.3.7) into (6.1.3.6) we get (6.1.3.1).

The secant method can be modified in such a way that

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_0)} f(x_k) \qquad (k = 1, 2, \ldots). \tag{6.1.3.8}$$

This method is often called **regula falsi** or false position method. Differently from secant method, where is enough to take $x_1 \neq x_0$, at this method one needs to take x_1 and x_0 on different sides of root $x = a$. Geometric interpretation of false position method is given in Figure 6.1.3.2.

Iterative function at modified secant method is

$$\Phi(x) = x - \frac{x - x_0}{f(x) - f(x_0)} f(x) = \frac{x_0 f(x) - x f(x_0)}{f(x) - f(x_0)}.$$

Supposing $f \in C^1[\alpha, \beta]$, then

$$\Phi'(x) = \frac{f(x_0)}{f(x) - f(x_0)} \left[\frac{x - x_0}{f(x) - f(x_0)} f'(x) - 1 \right].$$

Because $\Phi(a) = a$ and $\Phi'(a) \neq 0$, we conclude that iterative process (6.1.3.8), if converges, has order of convergence 1. Condition of convergence is, in this case,

$$|\Phi'(x)| \leq q \leq 1 \qquad (f(x) \neq f(x_0)),$$

for every $x \in [\alpha, \beta] \setminus \{x_0\}$.

Example 6.1.3.4.

Using convergence acceleration (see [1, Theorem 2.4.1, p. 197]) on the iterative process 6.1.3.8, we get the iterative process of second order

$$x_{k+1} = \frac{x_0 g(x_k) - x_k h(x_k)}{g(x_k) - h(x_k)} \qquad (k = 1, 2, \ldots),$$

where $g(x) = \dfrac{f(x) - f(x_0)}{x - x_0}$ and $h(x) = \dfrac{f'(x) f(x_0)}{f(x)}$.

Remark 6.1.3.2.

By replacing $f'(x_k)$ in Newton's method by finite difference in point x_k, with step $h = f(x_k)$, i.e.

$$f'(x_k) \cong \frac{f(x_k + f(x_k)) - f(x_k)}{f(x_k)}$$

one gets Steffensen's method

$$x_{k+1} = x_k - \frac{f(x_k)^2}{f(x_k + f(x_k)) - f(x_k)} \qquad (k = 0, 1, \ldots). \tag{6.1.3.9}$$

Steffensen's method is interesting because it has order of convergence 2, and thereby iterative function

$$x \to \Phi(x) = x - \frac{f(x)^2}{f(x + f(x)) - f(x)}$$

does not contain derivative f'.

6.1.4. Bisection method

Let on segment α, β exist isolated simple root $x = a$ of equation

$$f(x) = 0, \tag{6.1.4.1}$$

where $f \in C[\alpha, \beta]$. Method of interval bisection for solution of equation (6.1.4.1) consists in construction of series' of intervals $\{(x_k, y_k)\}_{k \in N}$ such that

$$y_{k+1} - x_{k+1} = \frac{1}{2}(y_k - x_k), \qquad (k = 1, 2, \ldots)$$

having thereby $\lim\limits_{k \to +\infty} x_k = \lim\limits_{k \to +\infty} y_k = a$. The noted process of construction of intervals is interrupted when, for example, interval length becomes lesser than in advance given small positive number ε. This method can be described with four steps:

I. $k := 0,\ x_1 = \alpha,\ y_1 = \beta$;

II. $k := k + 1, z_k := \frac{1}{2}(x_k + y_k)$;

III. If

$$\begin{aligned} f(z_k) f(x_k) < 0 &\text{ take } x_{k+1} := x_k,\ y_{k+1} := z_k, \\ > 0 &\text{ take } x_{k+1} := z_k,\ y_{k+1} := y_k, \\ = 0 &\text{ take } a := z_k; \text{ end of calculation} \end{aligned}$$

IV. If

$$\begin{aligned} |y_{k+1} - x_{k+1}| \geq \varepsilon &\quad \text{go to II,} \\ < \varepsilon &\quad z_{k+1} := \frac{1}{2}(x_{k+1} + y_{k+1}) \\ &\quad \text{end of calculation.} \end{aligned}$$

Note that error estimation for approximation of z_{k+1} is

$$|z_{k+1} - a| \leq \frac{1}{2^{k+1}}(\beta - \alpha).$$

6.1.5. Schröder Development

Let function $f : [\alpha, \beta] \to R$ be differentiable and $f'(x) \neq 0 \quad (\forall x \in [\alpha, \beta])$. Consequently, f is then strictly monotonous on $[\alpha, \beta]$ and there exists its inverse function F which is also differentiable. Then

$$F'(y) == \frac{dx}{dy} = \frac{1}{f'(x)} \qquad (y = f(x)).$$

If function f is two times differentiable on $[\alpha, \beta]$ then

$$F''(y) = -\frac{f''(x)}{f'(x)^3}.$$

Finding the higher derivatives of function F, supposing that it is enough times differentiable, could be very complicated. Being necessary for Schröder development, a recursive procedure is suggested (see [1], pp. 353).

Suppose that function f is $(n+1)$ times differentiable on $[\alpha, \beta]$, and that

$$F^{(k)}(y) = \frac{X_k}{(f')^{2k-1}} \qquad (k = 1, \ldots, n+1), \tag{6.1.5.1}$$

where X_k is polynomial in $f', f'', \ldots, f^{(k)}$ and $f^{(i)} \equiv f^{(i)}(x)$ for $i = 1, \ldots, n+1$. By induction one can prove the formula (6.1.5.1), where polynomial X is determined recursively

$$X_{k+1} = f' X_k{}' - (2k-1) X_k f'', \tag{6.1.5.2}$$

starting with $X_1 = 1$ and $X_2 = -f''$.

First five members of series $\{X_k\}$ are

$$\begin{aligned}
X_1 &= 1, \\
X_2 &= -f'', \\
X_3 &= -f' f''' + 3 f''^2 \\
X_4 &= -f'^2 f^{IV} + 10 f' f'' f''' - 15 f''^3, \\
X_5 &= -f'^3 f^V + 15 f'^2 f'' f^{IV} + 10 f'^2 f'''^2 - 105 f' f''^2 f''' + 105 f''^4.
\end{aligned}$$

Suppose that function f has on segment $[\alpha, \beta]$ a simple zero $x = a$ whose surrounding denote with $U(a)$. If we put $h = -\dfrac{f(x)}{f'(x)} \quad (x \in U(a))$, then $f(x) + h f'(x) = 0$, wherefrom we have

$$a = F(0) = F(f + h f').$$

If $f \in C^{n+1}[\alpha, \beta]$, based on Taylor's formula we have

$$a = \sum_{k=0}^{n} \frac{1}{k!} F^{(k)}(y)(h f')^k + \frac{F^{(n+1)}(\overline{y})}{(n+1)!}(h f')^{n+1},$$

where $\overline{y} = f + t h f' = (1-t) f = \theta f \quad (t, \theta \in (0,1))$. Finally, using (6.1.5.1) we get Schröder development

$$a - x = \sum_{k=1}^{n} \frac{1}{k!} X_k \left(\frac{f'}{f'}, \frac{f''}{f'}, \ldots, \frac{f^{(k)}}{f'} \right) h^k + \mathbf{O}(f(x)^{n+1}$$

i.e.

$$\begin{aligned}
a - x = h - \frac{f''}{2f'} h^2 + \frac{3f''^2 - f' f'''}{6 f'^2} h^3 \\
+ \frac{10 f' f'' f''' - f'^2 f^{IV} - 15 f''^3}{24 f'^3} h^4 + \ldots
\end{aligned} \tag{6.1.5.3}$$

6.1.6. Methods of higher order

In this section we will present some ways for obtaining iterative processes with order of convergence greater than 2, supposing that equation

$$f(x) = 0, \tag{6.1.6.1}$$

has on segment $[\alpha, \beta]$ unique simple root $x = a$, and that function f is enough times differentiable on $[\alpha, \beta]$.

1. Using Schröder development, by taking finite number of first members on right hand site of (6.1.5.3), we can get a number of iterative formulas.

Let

$$\begin{aligned}
\Phi_2(x) &= x + h = x - \frac{f(x)}{f'(x)},\\
\Phi_3(x) &= \Phi_2(x) - \frac{f(x)''}{2f'(x)}h^2 = x - \frac{f(x)}{f'(x)} - \frac{f''(x)f(x)^2}{2f'(x)^3},\\
\Phi_4(x) &= \Phi_3(x) + \frac{3f''^2 - f'f'''}{6f'^2}h^3\\
&= x - \frac{f(x)}{f'(x)} - \frac{f''(x)f(x)^2}{2f'(x)^3} - \frac{f(x)^3}{6f'(x)^4}\left(3\frac{f''(x)^2}{f'(x)} - f'''(x)\right),
\end{aligned}$$

etc.

Note that $\Phi_2(x)$ is iteration function of Newton's method.

Because h being in first iteration $a - x \quad (x \to a)$, based on (6.1.5.3) we have

$$\Phi_m(x) - a = \mathbf{O}(h^m) = \mathbf{O}((x-a)^m) \qquad (m = 2, 3, \ldots),$$

when $x \to a$, meaning that

$$x_{k+1} = \Phi_m(x_k) \qquad (k = 0, 1, \ldots), \tag{6.1.6.2}$$

applied to finding root of equation (6.1.6.2), has order of convergence at least m.

Formulas (6.1.6.2) are often called Chebyshev iterative formulas.

Using Hermite's interpolation formulas (see Chapter 7) for function f in points $x = x_{k-1}$ and $x = x_k$ one can get the iterative formula of form (see [10])

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f(x_k)^2}{2f'(x_k)}\overline{f}''(x_k) \qquad (k = 1, 2, \ldots), \tag{6.1.6.3}$$

where $\overline{f}''(x_k) = -\dfrac{6}{\varepsilon_k^2}(f(x_k) - f(x_{k-1})) + \dfrac{2}{\varepsilon_k}(2f'(x_k) + f'(x_{k-1}))$ and $\varepsilon = x_k - x_{k-1}$. Order of convergence of this process is $r = 1+\sqrt{3}$. Iterative function of this process is one modification of Chebyshev function Φ_3.

In paper [11], Milovanović and Petković considered one modification of function Φ_3 using approximation

$$f''(x_k) \approx \frac{f'(x_k + \varepsilon_k) - f'(x_k)}{\varepsilon_k}$$

whereby $\varepsilon \to 0$ when $k \to +\infty$. The corresponding iterative process is

$$x_{k+1} = x_k - \frac{f(x)}{f'(x)} - \frac{f(x_k)^2}{2f'(x_k)^3} \cdot \frac{f'(x_k + \varepsilon_k) - f'(x_k)}{\varepsilon_k}. \tag{6.1.6.4}$$

From asymptotic equation (see [1], pp. 357-358)

$$|e_{k+1}| \sim \left|\frac{f'''(a)}{4f'(a)}\right||e_k|^2|e_{k-1}|,$$

similar as at secant method, one finds order of convergence from equation $r^2-2r-1=0$, i.e. $r=1+\sqrt{2}$. Asymptotic error constant is $C_r=\left|\frac{f'''(a)}{4f'(a)^{1/\sqrt{2}}}\right|$.

Introducing approximations for $f'(x)$ and $f''(x)$ of forms

$$\begin{aligned} f'(x) &\cong \bar{f}'(x)=\frac{f(x+f(x))-f(x-f(x))}{2f(x)} \\ f''(x) &\cong \bar{f}''(x)=\frac{f(x+f(x))-2f(x)+f(x-f(x))}{f(x)^2} \end{aligned} \tag{6.1.6.5}$$

in paper [12] is presented the iterative function

$$\Phi_3^*(x)=x-\frac{f(x)}{\bar{f}'(x)}-\frac{f(x)^2\bar{f}''(x)}{f(x)^2},$$

and proven that order of convergence of this process is 3.

2. By using method for acceleration of iterative processes

$$x_{k+1}=x_k-\frac{x_k-\Phi(x_k)}{1-\frac{1}{r}\Phi'(x_k)} \qquad (k=0,1,\ldots)$$

for Newton's method $\Phi(x)=x-\frac{f(x)}{f'(x)}$ we get the method

$$x_{k+1}=x_k-\frac{2f(x_k)f'(x_k)}{2f'(x_k)^2-f(x_k)f''(x_k)} \qquad (k=0,1,\ldots),$$

with order of convergence at least 3, known as Saleh method of tangent hyperbolas or Halley's method.

Using approximation (6.1.6.5), Milovanović and Kovačević in paper [12] considered iterative function

$$\Phi_3(x)=x-\frac{2f(x)\bar{f}'(x)}{2\bar{f}'(x)^2-f(x)\bar{f}''(x)},$$

which is one modification of Halley's method (6.1.6.7) with cubic convergence.

Some more complicated iterative formulas of higher order can be found in ([1], pp. 360-365).

6.2. Systems of nonlinear equations

6.2.0. Introduction

System of nonlinear equations

$$f_i(x_1,\ldots,x_n)=0 \qquad (i=1,\ldots,n) \tag{6.2.0.1}$$

where $f_i: R^n \to R$ $(i=1,\ldots,n)$ are given functions, can be considered as special case of operator equation

$$Fu=\theta, \tag{6.2.0.2}$$

where F is operator which maps Banach space X to Banach space Y, and θ null-vector. Thus, $X=Y=R^n$, $u=\vec{x}=[x_1\ldots x_n]^T$, $\theta=[0\ldots 0]^T$,

$$Fu=\vec{f}(\vec{x})=\begin{bmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_n(x_1,\ldots,x_n) \end{bmatrix}. \tag{6.2.0.3}$$

Basic method for solving operator equation (6.2.0.2) and also system of equations (6.2.0.1) is Newton-Kantorowich (Newton-Raphson) method, which is generalization of Newton method (6.1.1.3) .

6.2.1. Newton-Kantorowitch (Raphson) method

Basic iterative method for solving equation (6.2.0.2) is method of Newton-Kantorowich, which generalizes classical Newton's method. Fundamental results regarding existence and uniqueness of solutions of eq. (6.2.0.2) and convergence of the method are given by L.V. Kantorowich (see [22]).

Suppose that eq. (6.2.0.2) has a solution $u = a$ and operator $F : X \to Y$ is Fréchet differentiable in convex neighbourhood D of point a. Method of Newton-Kantorowich rely on linearization of eq. (6.2.0.2). Suppose that the approximative solution u_k has been found. Then, for obtaining the next approximation u_{k+1} replace eq. (6.2.0.2) with

$$Fu_k + F'_{(u_k)}(u - u_k) = \theta. \tag{6.2.1.1}$$

If for operator $F'_{(u_k)}$ there exists inverse operator $\Gamma(u_k) = (F'_{(u_k)})^{-1}$, from (6.2.1.1) we obtain the iterative method

$$u_{k+1} = u_k - \Gamma(u_k)Fu_k \qquad (k = 0, 1, \ldots) \tag{6.2.1.2}$$

known as Newton-Kantorowich method. Starting value u_0 for generating series $\{u_k\}$ is taken from D, and its selection is very tough problem. Method (6.2.1.2) can be presented as

$$u_{k+1} = Tu_k \qquad (k = 0, 1, \ldots),$$

where

$$Tu = u - \Gamma(u)Fu.$$

For developing a methods for solution of systems of nonlinear equations, we will induce some crucial theorems without proofs (see [1], pp. 375 - 380].

Theorem 6.2.1.1. *Let operator F be two times Fréchet differentiable on D, whereby for every $u \in D$ there exists operator $\Gamma(u)$. If the operators $\Gamma(u)$ and $F''(u)$ are limited, and $u_0 \in D$ is close enough to point a, the iterative process* (6.2.1.2) *has order of convergence at least two.*

For usual consideration we suppose that D is a ball in $K[u_0, R]$, where u_0 is starting value of series $\{u_k\}_{k\in N_0}$.

If Lipschitz condition

$$||F'_{(u)} - F'_{(v)}|| \le L||u - v|| \qquad (u, v \in K[u_0, R]) \tag{6.2.1.3}$$

is fulfilled, from

$$Fu - Fv - F'_{(v)}(u - v) = \int_0^1 [F'_{v+t(u-v))} - F'_{(v)}](u - v)dt$$

follows the inequality

$$||Fu - Fv - F'_{(v)}(u - v)|| \le \frac{L}{2} = ||u - v||^2.$$

Theorem 6.2.1.2. *Let operator F be Fréchet differentiable in the ball $K[u_0, R]$ and satisfies the condition* (6.2.1.3) *and let the nonequalities*

$$||\Gamma_0|| \le b_0,\ ||\Gamma_0 Fu_0|| \le \eta_0,\ h_0 = b_0 L\eta_0 \le \frac{1}{2}, \tag{6.2.1.5}$$

where $\Gamma_0 = \Gamma(u_0)$, *hold. If*

$$R \geq r_0 = \frac{1-\sqrt{1-2h_0}}{h_0}\eta_0, \tag{6.2.1.6}$$

series $\{u_k\}_{k\in N_0}$, *generating by means of* (6.2.1.2) *converges to solution* $a \in K[u_0, r_0]$ *of equation* (6.2.0.2).

For given series' $\{b_k\}, \{\eta_k\}, \{h_k\}, \{r_k\}$ defined with

$$b_{k+1} = \frac{b_k}{1-h_k}, \quad \eta_{k+1} = \frac{h_k}{2(1-h_k)}\eta_k,$$
$$h_{k+1} = b_{k+1}L\eta_{k+1}, \ r_k = \frac{1-\sqrt{1-2h_{k+1}}}{h_{k+1}}$$

the existence of series $\{u_k\}_{k\in N_0}$ is proven and the following relations

$$||\Gamma(u_k)|| \leq b_k, \ ||\Gamma(u_k)Fu_k|| \leq \eta_k, \ h_k \leq \frac{1}{2} \tag{6.2.1.7}$$

and

$$K[u_k, r_k] \subset K[u_{k-1}, r_{k-1}]. \tag{6.2.1.8}$$

hold.

Theorem 6.2.1.3. *When conditions of previous theorem are fulfilled, then it holds*

$$||u_k - a|| \leq \frac{1}{2^{k-1}}(2h_0)^{2^k-1}\eta_0 \quad (k \in N). \tag{6.2.1.9}$$

In order to avoid determination of inverse operator $\Gamma(u) = [F'_{(u)}]^{-1}$ at every step, method of Newton-Kantorowich can be modified in the following way

$$u_{k+1} = u_k - \Gamma_0 F u_k \quad (k = 0, 1, 2, \ldots), \tag{6.2.1.10}$$

where $\Gamma_0 = \Gamma(u_0)$. By introducing the operator T

$$Tu = u - \Gamma_0 F u, \tag{6.2.1.11}$$

the modified method (6.2.1.10) can be given in the form

$$u_{k+1} = Tu_k \quad (k = 0, 1, 2, \ldots).$$

Suppose that the following conditions are met:
a. Operator F is Fréchet-differentiable in the ball $K[u_0, R]$,
b. $F'_{(u)}$ satisfies the condition (6.2.1.3),
c. Operator Γ_0 exists and

$$||\Gamma_0|| \leq b_0, \quad ||\Gamma_0 F u_0|| \leq \eta_0.$$

Then, it holds the following

Theorem 6.2.1.4. *If the following condition hold,*

$$h_0 = b_0 L\eta_0 < \frac{1}{2}, \ \text{and} \quad r_0 = \frac{1-\sqrt{1-2h_0}}{h_0}\eta_0 \leq R$$

the series generated using (6.2.1.10) *converges to solution* $a \in K[u_0, r_0]$ *of equation* (6.2.0.2).

It can be shown that iterative process (6.2.1.10) is of first order.

Consider now system of nonlinear equations

$$f_i(x_1,\dots,x_n)=0 \qquad (i=1,2,\dots,n). \tag{6.2.1.12}$$

Here is $X=Y=R^n$, $u=\vec{x}=[x_1\dots x_n]^T$ and F defined by (6.2.0.3). If F is Fréchet differentiable operator, the

$$F'_{(u)}=\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & & \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}=W(\vec{x}),$$

i.e. $W(\vec{x})$ is Jacobian matrix for $\vec{f}$. If $\det(W(\vec{x}))\neq 0$, i.e. if matrix $W(\vec{x})$ is regular, method of Newton-Kantorowich for solving system of equations (6.2.1.12) is given as

$$\vec{x}^{(k+1)}=\vec{x}^{(k)}-W^{-1}(\vec{x}^{(k)}) \qquad (k=0,1,\dots), \tag{6.2.1.13}$$

where $\vec{x}^{(k)}=[x_1^{(k)}\ \dots\ x_n^{(k)}]^T$. This method is often called Newton-Raphson method.

Method (6.2.1.13) can be obtained in a simpler way, by linearization of system of equation (6.2.0.1) in neighborhood of approximation $x^{(k)}$. Let $\vec{a}=[a_1\ \dots a_n]^T$ be exact solution of this system. Using Taylor development for functions which appear in (6.2.0.1), we get

$$f_i(a_1,\dots,a_n)=f_i(x_1^{(k)},\dots,x_n^{(k)})+\frac{\partial f_i}{\partial x_1}(a_1-x_1^{(k)})+\dots$$
$$+\frac{\partial f_i}{\partial x_n}(a_n-x_n^{(k)})+r_i^{(k)} \qquad (i=1,\dots,n),$$

where partial derivatives on the right-hand side of given equations are calculated in point $\vec{x}^{(k)}$. $r_i^{(k)}$ represents corresponding residual in Taylor's formula.

Because of $f_i(a_1,\dots,a_n)=0$ $(i=1,\dots,n)$, previous system of equations can be represented in matrix form

$$\vec{0}=\vec{f}(\vec{x}^{(k)})+W(\vec{x}^{(k)})(\vec{a}-(\vec{x}^{(k)}))+\vec{r}^{(k)},$$

where $\vec{r}^{(k)}=[r_1^{(k)}\dots r_n^{(k)}]^T$. If Jacobian matrix for $\vec{f}$ is regular, then we have

$$\vec{a}=\vec{x}^{(k)}-W^{-1}(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)})\vec{r}^{(k)}.$$

By neglecting the very last member on the right-hand size, in spite of vector $\vec{a}$, we get its new approximation, denoted with $\vec{x}^{(k+1)}$. In this way, one gets (6.2.1.13).

As already noted, method (6.2.1.13) can be modified in the sense that inverse matrix of $W(\vec{x})$ is not evaluated at every step, but only at first. Thus,

$$\vec{x}^{(k+1)}=\vec{x}^{(k)}-W^{-1}(\vec{x}^{(0)}) \qquad (k=0,1,\dots). \tag{6.2.1.14}$$

Remark 6.2.1.1. *Modified method* (6.2.1.14) *can be considered as simple iterative method*

$$\vec{x}^{(k+1)}=T\vec{x}^{(k)}+\mathbf{\Lambda}\vec{f}(\vec{x}^{(k)}) \qquad (k=0,1,\dots).$$

with matrix $\mathbf{\Lambda}$ *obtained from condition that derivative of* T *is null-operator, i.e.* $\vec{I}+\mathbf{\Lambda}\mathbf{W}(\vec{x}^{(0)})$ *is null-matrix. If* $\mathbf{W}(\vec{x}^{(0)})$ *is regular matrix, then we have* $\mathbf{\Lambda}=\mathbf{W}^{-1}(\vec{x}^{(0)})$.

Previous introductory theorems can be adapted for the case of system of nonlinear equations, whereby the conditions for convergence of processes (6.2.1.13) and (6.2.1.14) can be expressed in different ways, depending on introduced norms in $\mathbf{X}$. For example, taking for norm in R^n

$$||\vec{x}||=||\vec{x}||_\infty=\max_i|x_i|$$

and supposing that $f\in C^2(D)$, where D is ball $K[x^{(0)},R]$ from theorem 6.2.1.2 it follows

Corollary 6.2.1.1. *If in D are fulfilled the following conditions:*

$$s_{ij} = \sum_{k=1}^{n} \left| \frac{\partial^2 f_i}{\partial x_j \partial x_k} \right| \qquad (i, j = 1, \ldots, n); \tag{6.2.1.15}$$

$$||\vec{f}(\vec{x}^{(0)})|| \le Q, \quad ||W^{-1}(\vec{x}^{(0)})|| \le b; \tag{6.2.1.16}$$

$$\Delta_0 = \det \mathbf{W}(\vec{x}^{(0)}) \neq 0, \quad h = nNQb^2 \le \frac{1}{2}. \tag{6.2.1.17}$$

Then, if $R \ge r = \dfrac{1-\sqrt{1-2h}}{h} Qb$, *method Newton-Kantorowich* (6.2.1.13) *converges to solution* $a \in K[\vec{x}^{(0)}, r]$.

Because for $0 < h \le 1/2$ it holds $(1-\sqrt{1-2h})/h \le 2$, so that for r in Corollary 6.2.1.1 we can take $r = 2Qb$.

Modified Newton-Kantorowich method (6.2.1.14) converges also under conditions given in Corollary 6.2.1.1.

In [1, pp. 384-386] the Newton-Kantorowich method is illustrated with system of nonlinear equation in two unknowns. It is suggested to reader to write a program code in `Mathematica` and `Fortran`.

Example 6.2.1.1. *Solve the system of nonlinear equation*

$$f_1(x_1, x_2) = 9x_1^2 x_2 + 4x_2^2 - 36 = 0$$
$$f_2(x_1, x_2) = 16x_2^2 - x_1^4 + x_2 + 1 = 0,$$

which has a solution in first quadrant $(x_1, x_2 > 0)$.

Using graphic presentation of implicit functions f_1 and f_2 in first quadrant, one can see that solution $\vec{a}$ is located in the neighborhood of point $(2, 1)$, so that we take for initial vector $\vec{x}^{(0)} = [2\ 1]^T$, i.e. $x_1^{(0)} = 2$ and $x_2^{(0)} = 1$.

By partial derivation of f_1 and f_2 one gets the Jacobian

$$W(x) = \begin{bmatrix} 18x_1x_2 & 9x_1^2 + 8x_2 \\ -4x_1^3 & 32x_2 + 1 \end{bmatrix},$$

and its inverse

$$W^{-1}(\vec{x}) = \frac{1}{\Delta(\vec{x})} \begin{bmatrix} 32x_2 + 1 & -(9x_1^2 + 8x_2) \\ 4x_1^3 & 18x_1x_2 \end{bmatrix},$$

where

$$\Delta(\vec{x}) = 18x_1x_2(32x_2 + 1) + 4x_1^3(9x_1^2 + 8x_2).$$

By putting $f_i^{(k)} \equiv f_i(x_1^{(k)}, x_2^{(k)})$ and $\Delta_k \equiv \Delta(x^{(k)})$ $(i = 1, 2;\ k = 0, 1 \ldots)$ in the scalar form of Newton-Kantorowich formula (6.2.1.13), we get the iteration formula

$$x_1^{(k+1)} = x_1^{(k)} - \frac{1}{\Delta_k}\{(32x_2^{(k)} + 1)f_1^{(k)} - (9x_1^{(k)^2} + 8x_2^{(k)})f_2^{(k)}\},$$
$$x_2^{(k+1)} = x_2^{(k)} - \frac{1}{\Delta_k}\{4x_1^{(k)^3} f_1^{(k)} + 18x_1^{(k)}x_2^{(k)} f_2^{(k)}\}.$$

The appropriate `Fortran` code for solving given nonlinear equation is

```
Double precision x1,x2,x10,x11,x20,x21,f1,f2,Delta,EPS
F1(x1,x2)=9*x1**2*x2 + 4*x2**2-36
F2(x1,x2)=16*x2**2 - x1**4 + x2 + 1
Delta(x1,x2)=18*x1*x2*(32*x2+1)+4*x1**3*(9*x1**2+8*x2)
```

```
      Open(1, File='Newt-Kant.out')
      x10=2.d0
      x20=1.d0
      EPS=1.d-6
      Iter=0
      write(1,5)
 5    format(1h ,// 3x, 'i',7x,'x1(i)',9x,'x2(i)',
     * 9x,'f1(i)', 9x,'f2(i)'/)
      write(1,10)Iter, x10,x20,F1(x10,x20),F2(x10,x20)
 1    x11=x10-((32*x20+1)*f1(x10,x20)-(9*x10**2+8*x20)*
     * f2(x10,x20)) /Delta(x10,x20)
      x21=x20-(4*x10**3*f1(x10,x20)+18*x10*x20*f2(x10,x20))
     * /Delta(x10,x20)
      Iter=Iter+1
      write(1,10)Iter, x11,x21,F1(x11,x21),F2(x11,x21)
10    Format(1x,i3, 4D14.8,2x)
      If(Dabs(x10-x11).lt.EPS.and.Dabs(x20-x21).lt.EPS)stop
      If(Iter.gt.100)Stop
      x10=x11
      x20=x21
      go to 1
      End
```

and the output list of results is

```
 i       x1(i)          x2(i)          f1(i)          f2(i)
 0 .20000000D+01 .10000000D+01 .40000000D+01 .20000000D+01
 1 .19830508D+01 .92295840D+00 .73136345D-01 .88110835D-01
 2 .19837071D+01 .92074322D+00-.28694053D-04 .68348441D-04
 3 .19837087D+01 .92074264D+00-.10324186D-10-.56994853D-10
 4 .19837087D+01 .92074264D+00 .00000000D+00-.15543122D-14
```

6.2.2. Gradient method

Because Newton-Kantorowich method demands obtaining inverse operator $F'^{-1}_{(u)}$, what can be very complicated, and even impossible, it have been developed a whole class of quasi-Newton methods, which use some approximations of noted operator (see [21], [22], [23], [24]). One of this methods is gradient method.

For given system of nonlinear equations (6.2.1.12), with matrix form (see (6.2.0.3))

$$\vec{f}(\vec{x}) = \vec{0}. \tag{6.2.2.1}$$

The gradient method for solving a given system of equations is based on minimization of functional

$$U(\vec{x}) = \sum_{i=1}^{n} f_i(x_1, \ldots, x_n)^2 = (\vec{f}(\vec{x}), \vec{f}(\vec{x})).$$

It is easy to show that the equivalence $U(\vec{x}) = 0 \iff \vec{f}(\vec{x}) = \vec{0}$ holds.

Suppose that equation (6.2.2.1) has unique solution $\vec{x} = \vec{a}$, for which functional U riches minimum value. Let $\vec{x}^{(0)}$ be initial approximation of this solution. Let us construct series $\{\vec{x}^{(k)}\}$ such that $U(\vec{x}^{(0)}) > U(\vec{x}^{(1)}) > U(\vec{x}^{(2)}) > \cdots$. In a same way as at linear equations (see Chap. IV), we take

$$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} - \lambda_k \nabla U(\vec{x}^{\,(k)}) \qquad (k = 0, 1, \ldots), \tag{6.2.2.2}$$

where $\nabla U(\vec{x}) = grad(\vec{x}) = \left[\dfrac{\partial U}{\partial x_1} \cdots \dfrac{\partial U}{\partial x_n}\right]^T$. Parameter λ_k is to be determined from condition that scalar function S, defined with $S(t) = U(\vec{x}^{\,(k)} - t\nabla U(\vec{x}^{\,(k)}))$ has a minimum value in point $t = \lambda_k$. Having in mind that equation $S'(t) = 0$ is non-linear, proceed its linearization around $t = 0$. In this case we have

$$L_i^{(k)} = f_i(\vec{x}^{\,(k)} - t\nabla U(\vec{x}^{\,(k)})) = f_i(\vec{x}^{\,(k)}) - t(\nabla f_i(\vec{x}^{\,(k)}), \nabla U(\vec{x}^{\,(k)}))$$

so that linearized equation is

$$\sum_{i=1}^{n} L_i^{(k)} \frac{d}{dt} L_i^{(k)} = -\sum_{i=1}^{n} L_i^{(k)} (\nabla f_i \vec{x}^{(k)}), \nabla U(\vec{x}^{(k)})) = 0,$$

wherefrom we obtain

$$\lambda_k = t = \frac{\sum\limits_{i=1}^{n} H_i f_i(\vec{x}^{(k)})}{\sum\limits_{i=1}^{n} H_i^2}, \tag{6.2.2.3}$$

where we put $H_i = (\nabla f_i(\vec{x}^{(k)}),\ \nabla U(\vec{x}^{(k)}))\ \ (i = 1, \ldots, n)$. Because of

$$\frac{\partial U}{\partial x_j} = \frac{\partial}{\partial x_j} \left\{ \sum_{i=1}^{n} f_i(\vec{x})^2 \right\} = 2 \sum_{i=1}^{n} f_i(\vec{x}) \frac{\partial f_i(\vec{x})}{\partial x_j},$$

we have

$$\nabla U(\vec{x}) = 2\mathbf{W}^T(\vec{x}) \vec{f}(\vec{x}), \tag{6.2.2.4}$$

where $\mathbf{W}(\vec{x})$ is Jacobian matrix.

According to previous, (6.2.2.3) reduces to

$$\lambda_k = \frac{1}{2} \cdot \frac{(\vec{f}^{(k)}, \mathbf{W}_k \mathbf{W}_k^T \vec{f}^{(k)})}{(\mathbf{W}_k \mathbf{W}_k^T \vec{f}^{(k)}, \mathbf{W}_k \mathbf{W}_k^T \vec{f}^{(k)})},$$

where $\vec{f}^{(k)} = \vec{f}(\vec{x}^{(k)})$ and $\mathbf{W}_k = \mathbf{W}(\vec{x}^{(k)})$. Finally, gradient method can be represented in the form

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - 2\lambda_k W_k^T \vec{f}(\vec{x}^{(k)}) \qquad (k = 0, 1, \ldots).$$

As we see, in spite of matrix $\mathbf{W}^{-1}(\vec{x}^{(k)})$ which appears in Newton-Kantorowich method, we have now matrix $2\lambda_k \mathbf{W}_k^T$.

Example 6.2.1.2. *System of nonlinear equations given in example 6.2.1.1 will be solved using gradient method, starting with the same initial vector* $\vec{x}^{(0)} = [2\ 1]^T$*, giving the following list of results*

```
i       x1(i)          x2(i)           2lam_k
0 .2000000000D+01 .1000000000D+01  .305787395D-03
1 .1975537008D+01 .9259994504D+00  .538747689D-03
2 .1983210179D+01 .9201871306D+00  .339553623D-03
3 .1983643559D+01 .9207840032D+00  .535596539D-03
4 .1983705230D+01 .9207387845D+00  .339328604D-03
5 .1983708270D+01 .9207429317D+00  .535573354D-03
6 .1983708709D+01 .9207426096D+00  .393325162D-03
7 .1983708731D+01 .9207426391D+00  .535990624D-03
8 .1983708234D+01 .9207429368D+00  .337793301D-03
9 .1983708734D+01 .9207426370D+00
```

Note that the convergence here is much slower than at Newton-Kantorowich method, due to fact that gradient method is of first order.

Gradient method is successfully used in many optimization problems of nonlinear programming, with large number of methods, especially of gradient type, which are basis for number of programming packages for solving nonlinear programming problems.

6.2.3. Globally convergent methods

We have seen that Newtons method and Newton-like methods (quasi-Newton methods) for solving nonlinear equations has an unfortunate tendency not to converge if the initial guess is not sufficiently close to the root. A global method is one that converges

to a solution from almost any starting point. Therefore, it is our goal to develop an algorithm that combines the rapid local convergence of Newtons method with a globally convergent strategy that will guarantee some progress towards the solution at each iteration. The algorithm is closely related to the quasi-Newton method of minimization (see [5], p. 376).

From (6.2.1.13), Newton-Raphson method, we have so known Newton step in iteration formula

$$\vec{x}^{(k+1)} - \vec{x}^{(k)} = \delta\vec{x} = -\mathbf{W}^{-1}(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)}), \qquad (k = 0, 1, \ldots) \tag{6.2.3.1}$$

where $\mathbf{W}$ is Jacobian matrix. The question is how one should decide to accept the Newton step δx? If we denote $\mathbf{F} = \vec{f}(\vec{x}^{(k)})$, a reasonable strategy for step acceptance is that $|\mathbf{F}|^2 = \mathbf{F} \cdot \mathbf{F}$ decreases, what is the same requirement one would impose if trying to minimize

$$f = \frac{1}{2}\mathbf{F} \cdot \mathbf{F}. \tag{6.2.3.2}$$

Every solution of (6.2.1.12) minimizes (6.2.3.2), but there may be some local minima of (6.2.3.2) that are not solution of (6.2.1.12). Thus, simply applying some minimum finding algorithms can be wrong.

To develop a better strategy, note that Newton step (6.2.3.1) is a descent direction for f:

$$\nabla f \cdot \delta\vec{x} = (\mathbf{F} \cdot \mathbf{W}) \cdot (-\mathbf{W}^{-1} \cdot \mathbf{F}) = -\mathbf{F} \cdot \mathbf{F} < 0. \tag{6.2.3.3}$$

Thus, the strategy is quite simple. One should first try the full Newton step, because once we are close enough to the solution, we will get quadratic convergence. However, we should check at each iteration that the proposed step reduces f. If not, we go back (backtrack) along the Newton direction until we get acceptable step. Because the Newton direction is descent direction for f, we fill find for sure an acceptable step by backtracking.

It is to mention that this strategy essentially minimizes f by by taking Newton steps determined in such a way that bring ∇f to zero. In spite of fact that this method can occasionally lead to local minimum of f, this is rather rare in practice. In such a case, one should try a new starting point.

Line Searches and Backtracking

When we are not close enough to the minimum of f, taking the full Newton step $\vec{p} = \delta\vec{x}$ need not decrease the function; we may move too far for the quadratic approximation to be valid. All we are guaranteed is that initially f decreases as we move in the Newton direction. So the goal is to move to a new point x_{new} along the direction of the Newton step $\vec{p}$, but not necessarily all the way (see [5], pp. 377-378):

$$\vec{x}_{new} = \vec{x}_{old} + \lambda\vec{p} \qquad (0 < \lambda \leq 1) \tag{6.2.3.4}$$

The aim is to find λ so that $f(\vec{x}_{old} + \lambda\vec{p})$ has decreased sufficiently. Until the early 1970s, standard practice was to choose λ so that $\vec{x}_{new}$ exactly minimizes f in the direction $\vec{p}$. However, we now know that it is extremely wasteful of function evaluations to do so. A better strategy is as follows: Since $\vec{p}$ is always the Newton direction in our algorithms, we first try $\lambda = 1$, the full Newton step. This will lead to quadratic convergence when $\vec{x}$ is sufficiently close to the solution. However, if $f(\vec{x}_{new})$ does not meet our acceptance criteria, we backtrack along the Newton direction, trying a smaller value of λ, until

we find a suitable point. Since the Newton direction is a descent direction, we are guaranteed to decrease f for sufficiently small λ. What should the criterion for accepting a step be? It is not sufficient to require merely that $f(\vec{x}_{new}) < f(\vec{x}_{old})$. This criterion can fail to converge to a minimum of f in one of two ways. First, it is possible to construct a sequence of steps satisfying this criterion with f decreasing too slowly relative to the step lengths. Second, one can have a sequence where the step lengths are too small relative to the initial rate of decrease of f. A simple way to fix the first problem is to require the average rate of decrease of f to be at least some fraction α of the initial rate of decrease $\nabla f \cdot \vec{p}$

$$f(\vec{x}_{new}) \leq f(\vec{x}_{old}) + \alpha \nabla f \cdot (\vec{x}_{new} - \vec{x}_{old}). \tag{6.2.3.5}$$

Here the parameter α satisfies $0 < \alpha < 1$. We can get away with quite small values of α; $\alpha = 10^{-4}$ is a good choice. The second problem can be fixed by requiring the rate of decrease of f at $\vec{x}_{new}$ to be greater than some fraction β of the rate of decrease of f at $\vec{x}_{old}$. In practice, we will not need to impose this second constraint because our backtracking algorithm will have a built-in cutoff to avoid taking steps that are too small.

Here is the strategy for a practical backtracking routine. Define

$$g(\lambda) \equiv f(\vec{x}_{old} + \lambda \vec{p}) \tag{6.2.3.6}$$

so that

$$g'(\lambda) = \nabla f \cdot \vec{p} \tag{6.2.3.7}$$

If we need to backtrack, then we model g with the most current information we have and choose λ to minimize the model. We start with $g(0)$ and $g'(0)$ available. The first step is always the Newton step, $\lambda = 1$. If this step is not acceptable, we have available $g(1)$ as well. We can therefore model $g(\lambda)$ as a quadratic:

$$g(\lambda) \approx [g(1) - g(0) - g'(0)]\lambda^2 + g(0). \tag{6.2.3.8}$$

By first derivative of this function we find the minimum condition

$$\lambda = -\frac{g'(0)}{2[g(1) - g(0) - g'(0)]}. \tag{6.2.3.9}$$

Since the Newton step failed, we can show that $\lambda \lesssim \frac{1}{2}$ for small α. We need to guard against too small a value of λ, however. We set $\lambda_{min} = 0.1$.

On second and subsequent backtracks, we model g as a cubic in λ, using the previous value $g(\lambda_1)$ and the second most recent value $g(\lambda_2)$.

$$g(\lambda) = a\lambda^3 + b\lambda^2 + g'(0)\lambda + g(0) \tag{6.2.3.10}$$

Requiring this expression to give the correct values of g at λ_1 and λ_2 gives two equations that can be solved for the coefficients a and b.

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1/\lambda_1^2 & -1/\lambda_2^2 \\ -\lambda_2/\lambda_1^2 & \lambda_1/\lambda_2^2 \end{bmatrix} \cdot \begin{bmatrix} g(\lambda_1) - g'(0)\lambda_1 - g(0) \\ g(\lambda_2) - g'(0)\lambda_2 - g(0) \end{bmatrix}. \tag{6.2.3.11}$$

The minimum of the cubic (6.2.3.10) is at

$$\lambda = \frac{-b + \sqrt{b^2 - 3ag'(0)}}{3a}. \tag{6.2.3.12}$$

One should enforce that λ lie between $\lambda_{max} = 0.5\lambda_1$ and $\lambda_{min} = 0.1\lambda_1$. The corresponding code in FORTRAN is given in [5], pp. 378-381. It it suggested to reader to write the corresponding code in `Mathematica`.

Multidimensional Secant Methods: Broyden's Method

Newton's method as used previously is rather efficient, but it still have several disadvantages. One of the most important is that it needs Jacobian matrix. In many problems the Jacobian matrix is not available, i.e. there do not exist analytic derivatives. If the function evaluation is complicated, the finite-difference determination of Jacobian can be prohibitive. There are quasi-Newton methods which provide cheap approximation to the Jacobian for the purpose of zero finding. The methods are often called *secant methods*, because they reduce in one dimension to the secant method. One of the best of those methods is Broyden's method (see [21]).

If one denotes approximate Jacobian by $\mathbf{B}$, then the i-th quasi-Newton step $\delta\vec{x}_i$ is the solution of

$$\mathbf{B}_i \cdot \delta\vec{x}_i = -\mathbf{F}_i, \tag{6.2.3.13}$$

where $\delta\vec{x}_i = \vec{x}_{i+1} - \vec{x}_i$. Quasi-Newton or secant condition is that $\mathbf{B}_{i+1}$ satisfy

$$\mathbf{B}_{i+1} \cdot \delta\vec{x}_i = \delta\mathbf{F}_i, \tag{6.2.3.14}$$

where $\delta\mathbf{F}_i = \mathbf{F}_{i+1} - \mathbf{F}_i$. This is generalization of the one-dimensional secant approximation to the derivative, $\delta F/\delta x$. However, equation (6.2.3.14) does not determine $\mathbf{B}_{i+1}$ uniquely in more than one dimension. Many different auxiliary conditions to determine $\mathbf{B}_{i+1}$ have been examined, but the best one results from the Broyden's formula. This formula is based on idea of getting $\mathbf{B}_{i+1}$ by making a least change to $\mathbf{B}_i$ in accordance to the secant equation (6.2.3.14). Broyden gave the formula

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \frac{(\delta\mathbf{F}_i - \mathbf{B}_i \cdot \delta\vec{x}_i) \otimes \delta\vec{x}_i}{\delta\vec{x}_i \cdot \delta\vec{x}_i}. \tag{6.2.3.15}$$

One can check that $\mathbf{B}_{i+1}$ satisfies (6.2.3.14).

Early implementations of Broyden's method used the Sherman-Morrison formula to invert equation analytically,

$$\mathbf{B}_{i+1}^{-1} = \mathbf{B}_i^{(-1)} + \frac{(\delta\vec{x}_i - \mathbf{B}_i^{-1} \cdot \delta\mathbf{F}_i) \otimes \delta\vec{x}_i \cdot \mathbf{B}_i^{-1}}{\delta\vec{x}_i \cdot \delta\mathbf{B}_i^{-1} \cdot \delta\mathbf{F}_i}. \tag{6.2.3.16}$$

Thus, instead of solving equation (6.2.3.1) by, for example, **LU** decomposition, one determined

$$\delta\vec{x}_i = -\mathbf{B}_i^{-1} \cdot \mathbf{F}_i \tag{6.2.3.17}$$

by matrix multiplication in $O(n^2)$ operations. The disadvantage of this method is that it cannot be easily embedded in a globally convergent strategy, for which the gradient of equation (6.2.3.2) requires $\mathbf{B}$, not $\mathbf{B}^{-1}$

$$\nabla(\frac{1}{2}\mathbf{F} \cdot \mathbf{F}) \simeq \mathbf{B}^T \cdot \mathbf{F} \tag{6.2.3.18}$$

Accordingly, one should implement the update formula in the form (6.2.3.15). However, we can still preserve the $\mathbf{O}(n^2)$ solution of (6.2.3.1) by using **QR** decomposition of $\mathbf{B}_{i+1}$ in $\mathbf{O}(n^2)$ operations. All needed is initial approximation $\mathbf{B}_0$ to start process. It is often accepted to take identity matrix, and then allow $\mathbf{O}(n)$ updates to produce a reasonable

approximation to the Jacobian. In [5], p. 382-383, the first n function evaluations are spent on a finite-difference approximation in order to initialize **B**. Since **B** is not exact Jacobian, it is not guaranteed that $\delta\vec{x}$ is descent direction for $f = \frac{1}{2}\mathbf{F}\cdot\mathbf{F}$ (see eq. (6.2.3.3)). That has a consequence that the line search algorithm can fail to return the suitable step if **B** is far from the true Jacobian. In this case we simply reinitialize **B**.

Like the secant method in one dimension, Broyden's method converges superlinearly once you get close enough to the root. Embedded in a global strategy, it is almost as robust as Newton's method, and often needs far fewer function evaluations to determine a zero. Note that the final value of **B** is not always close to the true Jacobian at the root, in spite of fact that method converges.

The programme code ([5], pp. 383-385) of Broyden's method differs from Newtonian methods in using **QR** decomposition instead of **LU**, and determination of Jacobian by finite-difference approximation instead of direct evaluation.

More Advanced Implementations

One of the principal ways that the methods described above can fail is if matrix **W** (Newton-Kantorowich) or **B** (Broyden's method) becomes singular or nearly singular, so that Δx cannot be determined. This situation will not occur very often in practice. Methods developed so far to deal with this problem involve the monitoring of condition number of **W** and perturbing **W** if singularity or near singularity is detected. This feature is most easily implemented if **QR** decomposition instead of **LU** decomposition in Newton (or quasi-Newton) method is applied. However, in spite of fact that this method can solve problems when **W** is exactly singular and Newton's and Newton-like methods fail, it is occasionally less robust on other problems where **LU** decomposition succeeds. Implementation details, like roundoff, underflow, etc. are to be considered and taken in account.

In [5], considering effectiveness of strategies for minimization and zero finding, the global strategies have been based on *line searches*. Other global algorithms, like *hook step* and *dogleg step* methods, are based on the *model-trust region approach*, which is related to the Levenberg-Marquardt algorithm for nonlinear least-squares. In spite being more complicated than line searches, these methods have a reputation for robustness even when starting far from desired zero or minimum.

Numerous libraries and software packages are available for solving nonlinear equations. Many workstations and mainframe computers have such libraries attached to operating systems. Many commercial software packages contain nonlinear equation solvers. Very popolar among engineers are Matlab and Matcad. More sophisticated packages like Mathematica, IMSL, Macsyma, and Maple contain programs for nonlinear equation solving. The book *Numerical recipes* [5] contains numerous programs for solving nonlinear equation.

Bibliography (Cited references and further reading)

[1] Milovanović, G.V., *Numerical Analysis I*, Naučna knjiga, Beograd, 1988 (Serbian).
[2] Hoffman, J.D., *Numerical Methods for Engineers and Scientists.* Taylor & Francis, Boca Raton-London-New York-Singapore, 2001.
[3] Milovanović, G.V. and Djordjević, Dj.R., *Programiranje numeričkih metoda na FORTRAN jeziku.* Institut za dokumentaciju zaštite na radu "Edvard Kardelj", Niš, 1981 (Serbian).
[4] Stoer, J., and Bulirsch, R., *Introduction to Numerical Analysis*, Springer, New York, 1980.
[5] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., *Numerical Recepies - The Art of Scientific Computing.* Cambridge University Press, 1989.

[6] Djordjević, L.N., *An iterative solution of algebraic equations with a parameter to accelerate convergence.* Univ. Beograd. Publ. Elektrotehn. Fak. Ser. Mat.Fiz. No. 412- No. 460(1973), 179-182.
[7] Tihonov, O.N., *O bystrom vychyslniĭ najbolshih korneĭ mnogočlena.* Zap. Leningrad. gorn. in-ta 48, 3(1968), 36-41.
[8] Ostrowski, A., *Solution of Equations and System of Equations.* Academic Press, New Yurk, 1966.
[9] Wegstein, J.H., *Accelerating convergence of iterative processes.* Comm. ACM 1 (1958), 9-13.
[10] Ralston,A., *A First Course in Numerical Analysis.* McGraw-Hill, New York, 1965.
[11] Milovanović, G.V. & Petković, M.S., *On some modifications of a third order method for solving equations.* Univ. Beograd. Publ. Elektroteh. Fak. Ser. Mat. Fiz. No. 678 - No. 715 (1980), pp. 63-67.
[12] Milovanović, Kovačević, M.A., *Two iterative processes of third order without derivatives.* IV Znanstveni skup PPPR, Stubičke Toplice, 1982, Proceedings, pp. 63-67 (Serbian).
[13] Varjuhin, V.A. and Kasjanjuk, S.A., *Ob iteracionnyh metodah utočnenija korneĭ uravneniĭ.* Ž. Vychysl. Mat. i Mat. Fiz. 9(1969), 684-687.
[14] Lika, D.K., *Ob iteracionnyh metodah vissšego porjadka.* Dokl. 2-ĭ Nauč.-tehn. respubl. konf. Moldaviĭ. Kishinev, 1965, pp.13-16.
[15] Djordjević, L.N. & Milovanović, G.V., *A combined iterative formula for solving equations.* Informatika 78, Bled 1978, 3(207).
[16] Petković, M.S., *Some iterative interval methods for solving equations.* Ph.D. thesis, University Niš, 1980.
[17] Petković, M.S. & Petković, D. Lj., *On a method for two-sided approaching for solving equations.* Freiburger Intervall-Berichte 10(1980), pp. 1-10.
[18] Hoffman, J.D., *Numerical Methods for Engineers and Scientists.* Taylor & Francis, Boca Raton-London-New York-Singapore, 2001.
[19] *IMSL Math/Library Users Manual* , IMSL Inc., 2500 City West Boulevard, Houston TX 77042
[20] *NAG Fortran Library*, Numerical Algorithms Group, 256 Banbury Road, Oxford OX27DE, U.K., Chapter F02.
[21] Broyden, C.G., *Quasi-Newton methods and their application to function minimization*, Math. Comp. 29(1967), 368-381.
[22] Kantorowich, L.V., *Funkcional'nyi analiz i prikladnaja matematika.*, Uspehi Mat. Nauk 3(1948), 89-185.
[23] Ortega, J.M. & Rheinboldt, W.C., *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.
[24] Rall, L., *Computational solution of nonlinear operator equations*, New York, 1969.
[25] Kul'čickiĭ, O.Ju. & Šimelevič, L.I., *O nahoždeniĭ načal'nogo pribiženija.*, Ž. Vyčisl. Mat. i Mat. Fiz. 14(1974), pp. 1016-1018.
[26] Dennis, J.E. and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ: Prentice Hall, 1983.