3.10. SOLVING SYSTEMS OF NONLINEAR EQUATIONS

Let it be required to find a solution to a system of *n* nonlinear equations with *n* unknowns, which can be written as follows:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases},$$

or, shorter, in the vector form:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

where **x** is the vector of unknowns, **f** is the vector function:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix}, \ \mathbf{f} = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ ... \\ f_n(\mathbf{x}) \end{pmatrix}, \ \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ ... \\ 0 \end{pmatrix}.$$

The numerical solution of this system implies finding the values $x_1, x_2, \ldots, x_n$ that satisfy each of *n* given equations with the given accuracy ε.

In rare cases, to solve such a system, it is possible to apply the method of successive elimination of unknowns and reduce solving the original problem to solving a single nonlinear equation with one unknown. The values of other unknowns are then found by appropriate substitution in certain expressions. However, in the vast majority of cases, iterative methods are used to solve systems of nonlinear equations, which consist of constructing the sequence of vectors $\mathbf{x}^{(k)}, k = 0,1,2,...$ that converges in the limit $k \to \infty$ to the desired root $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T$ of the original system.

Like in case with one nonlinear equation, it is assumed that, at first, some isolated solution to the nonlinear system is sought, which is then refined in a certain convex region $G \in R^n$ with a given accuracy ε. Localization of the solution can be carried out on the basis of the specific information about the certain problem being solved (for example, for physical reasons) or using methods of calculus. When solving a system of two equations, quite often a graphical method is convenient when the location of the roots is determined as the intersection points

of the curves $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$ on the plane (x_1, x_2). However, there are no satisfactory root separation methods for high order systems.

Example. Localize a positive solution of the system of nonlinear equations:

$$\begin{cases} f_1(x_1, x_2) = 0.1x_1^2 + x_1 + 0.2x_2^2 - 0.3 = 0 \\ f_2(x_1, x_2) = 0.2x_1^2 + x_2 - 0.1x_1x_2 - 0.7 = 0 \end{cases}.$$

To determine the convex region $G \in R^2$ containing the desired solution, let's use the graphical method. In fig. 6 we can see the graphs of the curves of the functions $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$.

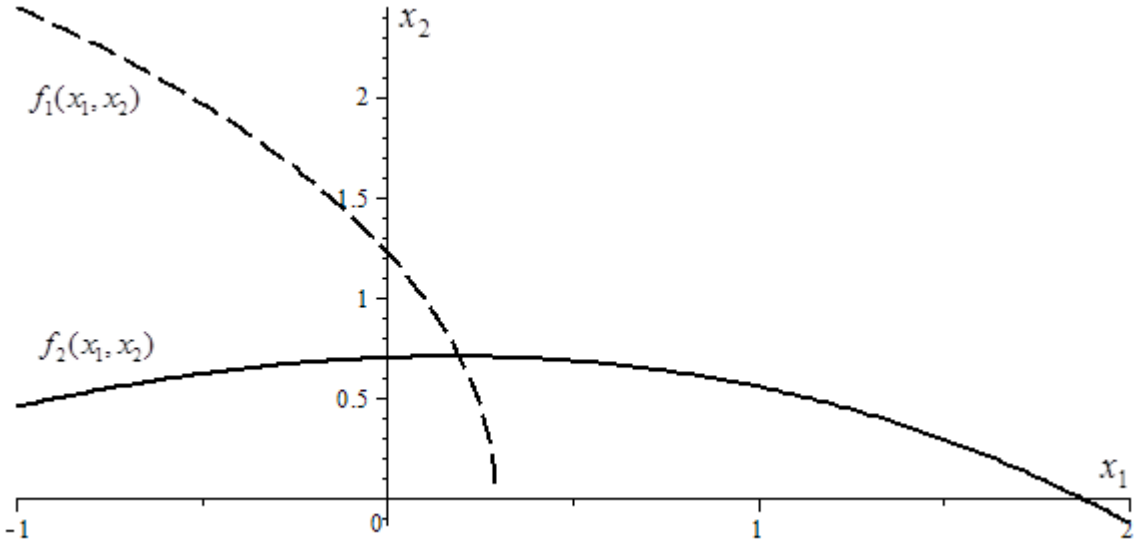

Fig. 6. The graphical method of root localization for the system of two equations

After plotting the curves $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$ on the plane (x_1, x_2) within the region of our interest, we shall find that the positive solution to the system of equations is located, for example, in a convex region that is the rectangle $G = \{0 \le x_1 \le 0.5; 0.5 \le x_2 \le 1\}$.

3.11. FIXED-POINT ITERATION METHOD (NONLINEAR SYSTEMS)

When using the generalized fixed-point iteration method, the system of equations $\mathbf{f(x)} = \mathbf{0}$ is reduced to an equivalent system of a special form:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n) \\ x_2 = \varphi_2(x_1, x_2, ..., x_n) \\ ..... \\ x_n = \varphi_n(x_1, x_2, ..., x_n) \end{cases},$$

or, in the vector form:

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{x}),\ \boldsymbol{\varphi}(\mathbf{x}) = \begin{pmatrix} \varphi_1(\mathbf{x}) \\ \varphi_2(\mathbf{x}) \\ ... \\ \varphi_n(\mathbf{x}) \end{pmatrix},$$

where the functions $\varphi_1(\mathbf{x})$, $\varphi_2(\mathbf{x})$, …, $\varphi_n(\mathbf{x})$ are defined and continuous in some neighborhood $G \in R^n$ of the desired isolated solution $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T$.

If some initial approximation $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T$ is chosen from $G \in R^n$, then subsequent approximations $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})^T$ in the fixed-point iteration method are found by the formulas:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ ..... \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \end{cases},\ k = 0, 1, 2, ...,$$

or, in the vector form:

$$\mathbf{x}^{(k+1)} = \boldsymbol{\varphi}(\mathbf{x}^{(k)}),\ k = 0, 1, 2, ...$$

If this sequence converges, then it converges to the desired solution $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T$ and the convergence is linear.

A sufficient condition for the convergence of the iteration process of the fixed-point iteration method is formulated as follows. Let the vector function $\boldsymbol{\varphi}(\mathbf{x})$ be continuous together with its derivative (the Jacobi matrix of the first derivatives of the system of functions)

$$\boldsymbol{\varphi}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial \varphi_1(\mathbf{x})}{\partial x_1} & \frac{\partial \varphi_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial \varphi_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial \varphi_2(\mathbf{x})}{\partial x_1} & \frac{\partial \varphi_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial \varphi_2(\mathbf{x})}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \varphi_n(\mathbf{x})}{\partial x_1} & \frac{\partial \varphi_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial \varphi_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

in a limited closed convex domain $G \in R^n$ and $\max_{\mathbf{x} \in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| \le q < 1$, where q is some constant. Then, if $\mathbf{x}^{(0)} \in G$ and all successive approximations $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})^T$ are also contained in $G \in R^n$, the iterative process $\mathbf{x}^{(k+1)} = \boldsymbol{\varphi}(\mathbf{x}^{(k)})$, $k = 0,1,2,...$ converges to a unique solution $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T$ of the equation $\mathbf{x} = \boldsymbol{\varphi}(\mathbf{x})$ in the domain $G \in R^n$.

In this case, the error estimates are also valid ($\forall k \in N$):

$$\left\|\mathbf{x}^{(*)} - \mathbf{x}^{(k)}\right\| \le \frac{q}{1-q}\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|, \ \left\|\mathbf{x}^{*} - \mathbf{x}^{(k)}\right\| \le \frac{q^k}{1-q}\left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\|.$$

So, the rule $\varepsilon^{(k)} = \frac{q}{1-q} \cdot \left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|$ is used as a condition for the termination of iterations in practical calculations.

It should be noted that when analyzing the convergence of a particular iterative scheme, the direct verification of the condition $\max_{\mathbf{x} \in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| \le q < 1$ may be difficult to perform; in this case we can instead calculate the norm of the "dominating" matrix $\mathbf{M}(\mathbf{x})$ with the elements $m_{ij}(\mathbf{x}) = \max_{\mathbf{x} \in G} \left|\frac{\partial \varphi_i(\mathbf{x})}{\partial x_j}\right|$, as $\max_{\mathbf{x} \in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| \le \|\mathbf{M}(\mathbf{x})\|$. Then if $\|\mathbf{M}(\mathbf{x})\| \le q < 1$, then the successive approximations $\mathbf{x}^{(k)}$ converge to the desired solution $\mathbf{x}^*$.

As before, the fixed-point iteration method may converge even if the condition $q < 1$ is violated.

The convergence of the fixed-point iteration method can again be accelerated by the Seidel method, which means that when calculating the component $x_i^{(k)}$ of the vector of unknowns at the *k*-th iteration, the already calculated values $x_1^{(k)}, x_2^{(k)}, \ldots, x_{i-1}^{(k)}$ are used at the same *k*-th iteration. The values for the remaining components $x_i^{(k-1)}, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)}$ are taken from the previous iteration. In this case, the iterative process looks as follows:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, ..., x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k+1)}, x_2^{(k)}, x_3^{(k)}, .., x_n^{(k)}) \\ x_3^{(k+1)} = \varphi_3(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k)}, ..., x_n^{(k)}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)}, ..., x_{n-1}^{(k+1)}, x_n^{(k)}) \end{cases}, \; k = 0,1,2,....$$

The convergence of this process is also linear. As well as when solving systems of linear equations, when using the Seidel method here, a problem of finding the optimal sequence of refinement of the components of the vector of unknowns $\mathbf{x}^{(k)}$ at each iteration can be stated.

Example. Find a positive solution to the system of nonlinear equations with an accuracy of $\varepsilon = 10^{-4}$ using the fixed-point iteration method:

$$\begin{cases} f_1(x_1, x_2) = 0.1x_1^2 + x_1 + 0.2x_2^2 - 0.3 = 0 \\ f_2(x_1, x_2) = 0.2x_1^2 + x_2 - 0.1x_1x_2 - 0.7 = 0 \end{cases}.$$

Let's transform the original system of equations to the following form:

$$\begin{cases} x_1 = 0.3 - 0.1x_1^2 - 0.2x_2^2 = \varphi_1(x_1, x_2) \\ x_2 = 0.7 - 0.2x_1^2 + 0.1x_1x_2 = \varphi_2(x_1, x_2) \end{cases}.$$

Let's check the fulfilment of the condition $\max\limits_{\mathbf{x}\in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| \le q < 1$ within the rectangular region $G = \{0 \le x_1 \le 0.5; 0.5 \le x_2 \le 1\}$ (i.e. $|x_1 - 0.25| \le 0.25$, $|x_2 - 0.75| \le 0.25$).

For this, let's find $\max\limits_{\mathbf{x}\in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| = \max\limits_{\mathbf{x}\in G} \left\{ \max\limits_{i=1..n} \sum\limits_{j=1}^{n} \left| \frac{\partial \varphi_i(x_1, x_2)}{\partial x_j} \right| \right\}$.

Let's determine the partial derivatives $\boldsymbol{\varphi}'(\mathbf{x})$:

$$\frac{\partial \varphi_1(x_1, x_2)}{\partial x_1} = -0.2x_1, \; \frac{\partial \varphi_1(x_1, x_2)}{\partial x_2} = -0.4x_2,$$

$$\frac{\partial \varphi_2(x_1, x_2)}{\partial x_1} = -0.4x_1 + 0.1x_2, \; \frac{\partial \varphi_2(x_1, x_2)}{\partial x_2} = 0.1x_1.$$

Obviously, in the region $G = \{0 \le x_1 \le 0.5; 0.5 \le x_2 \le 1\}$, we have:

$$\left| \frac{\partial \varphi_1(x_1, x_2)}{\partial x_1} \right| + \left| \frac{\partial \varphi_1(x_1, x_2)}{\partial x_2} \right| = |-0.2x_1| + |-0.4x_2| \le 0.1 + 0.4 = 0.5,$$

$$\left| \frac{\partial \varphi_2(x_1, x_2)}{\partial x_1} \right| + \left| \frac{\partial \varphi_2(x_1, x_2)}{\partial x_2} \right| = |-0.4x_1 + 0.1x_2| + |0.1x_1| \le 0.15 + 0.05 = 0.2.$$

So, $\max\limits_{\mathbf{x}\in G}\|\boldsymbol{\varphi}'(\mathbf{x})\| = \max(0.5, 0.2) = 0.5 = q < 1$.

Therefore, if successive approximations $(x_1^{(k)}, x_2^{(k)})$ do not leave the domain $G = \{0 \le x_1 \le 0.5; 0.5 \le x_2 \le 1\}$ (which is easy to detect in the process of calculations), then the iterative process will converge.

Let's take $x_1^{(0)} = 0.25$, $x_2^{(0)} = 0.75$ as the initial approximation.

Subsequent approximations are determined as

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}) \end{cases}, \; k = 0,1,2,...,$$

where $\varphi_1(x_1^{(k)}, x_2^{(k)}) = 0.3 - 0.1x_1^{(k)2} - 0.2x_2^{(k)2}$, $\varphi_2(x_1^{(k)}, x_2^{(k)}) = 0.7 - 0.2x_1^{(k)2} + 0.1x_1^{(k)}x_2^{(k)}$.

Iterations stop when the following condition is met:

$\varepsilon^{(k+1)} = \dfrac{q}{1-q}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \le \varepsilon$, where $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| = \max\limits_i \left|x_i^{(k+1)} - x_i^{(k)}\right|$.

The calculation results are presented in table 5.

The final approximation of the root is $x_1^* \approx 0.1964$, $x_2^* \approx 0.7062$.

Table 5

k	$x_1^{(k)}$, $x_2^{(k)}$	$\varphi_1(x_1^{(k)}, x_2^{(k)})$, $\varphi_2(x_1^{(k)}, x_2^{(k)})$	$\varepsilon^{(k)}$
0	0.25000, 0.75000	0.18125, 0.70625	–
1	0.18125, 0.70625	0.19696, 0.70623	0.06875
2	0.19696, 0.70623	0.19639, 0.70615	0.01571
3	0.19637, 0.70615	0.19641, 0.70615	0.00059
4	**0.19641, 0.70615**		**0.00005**

3.12. PROGRAM #10

Below is a proposed variant of the program algorithm for refining the value of the root of the system of two equations using the fixed-point iteration method or the Seidel method.

ALGORITHM “Fixed-point iteration method or Seidel method”

INPUT f(,), g(,), q, a, b, c, d, e

```
OUTPUT     r, s, u, v, x, y, k
BEGIN
    IF (q>=1)OR(q<=0) END
    x:=(a+b)/2, y:=(c+d)/2
    r:=2*e
    k:=0
    CYCLE “Iteration” WHILE (r>e)
        u:=x, v:=y
        #For the fixed-point iteration method
        x:=f(u,v), y:=g(u,v)
        #Alternatively for the Seidel method
        x:=f(u,v), y:=g(x,v)
        IF (x<a)OR(x>b)OR(y<c)OR(y>d) END
        IF (x>u) r:=x-u ELSE r:=u-x
        IF (y>v) s:=y-v ELSE s:=v-y
        IF (s>r) r:=s
        r:=r*q/(1-q)
        k:=k+1
    PRINT x, y, k
END
```

3.13. NEWTON METHOD (GENERALIZED)

The solution to the system of nonlinear equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ by the generalized Newton method is reduced to the solution of a sequence of linear problems that give a limit solution to the original problem. The linear problem is obtained by separating the main linear part from nonlinear equations.

If the initial approximation $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T \in G$ is determined, then the iterative process of finding the solution to the system of equations using the Newton method can be presented as follows:

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \Delta x_1^{(k)} \\ x_2^{(k+1)} = x_2^{(k)} + \Delta x_2^{(k)} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = x_n^{(k)} + \Delta x_n^{(k)} \end{cases}, \; k = 0,1,2,\ldots.$$

Here, the values of the increments $\Delta x_1^{(k)}$, $\Delta x_2^{(k)}$,…, $\Delta x_n^{(k)}$ are determined from the solution of the system of linear algebraic equations, all coefficients of which are expressed through the known previous approximation $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$:

$$\begin{cases} f_1(\mathbf{x}^{(k)}) + \dfrac{\partial f_1(\mathbf{x}^{(k)})}{\partial x_1}\Delta x_1^{(k)} + \dfrac{\partial f_1(\mathbf{x}^{(k)})}{\partial x_2}\Delta x_2^{(k)} + \ldots + \dfrac{\partial f_1(\mathbf{x}^{(k)})}{\partial x_n}\Delta x_n^{(k)} = 0 \\ f_2(\mathbf{x}^{(k)}) + \dfrac{\partial f_2(\mathbf{x}^{(k)})}{\partial x_1}\Delta x_1^{(k)} + \dfrac{\partial f_2(\mathbf{x}^{(k)})}{\partial x_2}\Delta x_2^{(k)} + \ldots + \dfrac{\partial f_2(\mathbf{x}^{(k)})}{\partial x_n}\Delta x_n^{(k)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(\mathbf{x}^{(k)}) + \dfrac{\partial f_n(\mathbf{x}^{(k)})}{\partial x_1}\Delta x_1^{(k)} + \dfrac{\partial f_n(\mathbf{x}^{(k)})}{\partial x_2}\Delta x_2^{(k)} + \ldots + \dfrac{\partial f_n(\mathbf{x}^{(k)})}{\partial x_n}\Delta x_n^{(k)} = 0 \end{cases}.$$

This linear system is obtained as a result of the expansion of the vector function $\mathbf{f}(\mathbf{x})$ according to the multidimensional Taylor formula at point $x^{(k+1)}$ after discarding the summands of the order higher than the first and based on the requirement of $\mathbf{f}(\mathbf{x}^{(k+1)}) = \mathbf{0}$ when $k \to \infty$.

In the vector-matrix form, the calculation formulas look as

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{\Delta x}^{(k)}$, $k = 0,1,2,\ldots,$

where the increment vector $\mathbf{\Delta x}^{(k)} = \begin{pmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \\ \ldots \\ \Delta x_n^{(k)} \end{pmatrix}$ is found from the solution to the equation

$\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{J}(\mathbf{x}^{(k)})\mathbf{\Delta x}^{(k)} = \mathbf{0}$.

Here, $\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \dfrac{\partial f_2(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial f_n(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$

is the Jacobi matrix of the first derivatives of the vector function $\mathbf{f}(\mathbf{x})$.

Expressing the increment vector $\Delta\mathbf{x}^{(k)}$ from this equation, the iterative process of finding the solution can ultimately be written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}^{-1}(\mathbf{x}^{(k)})\mathbf{f}(\mathbf{x}^{(k)}),\ k = 0,1,2,...,$$

where $\mathbf{J}^{-1}(\mathbf{x})$ is the inverse matrix of the Jacobi matrix. This formula is the generalization of the formula of the Newton method for nonlinear equations $x^{(k+1)} = x^{(k)} - \dfrac{f(x^{(k)})}{f'(x^{(k)})}$ to the case of systems of nonlinear equations.

However, when implementing the algorithm of the Newton method, in most cases, it is preferable not to calculate the inverse matrix $\mathbf{J}^{-1}(\mathbf{x}^{(k)})$, but to find directly the values of the increments $\Delta x_1^{(k)}$, $\Delta x_2^{(k)}$,…, $\Delta x_n^{(k)}$ from the system of linear algebraic equations and calculate a new approximation through them. To solve such SLAEs, we can use a variety of numerical methods, both direct and iterative, if necessary, taking into account the dimension *n* of the problem being solved and the specifics of the emerging Jacobi matrices $\mathbf{J}(\mathbf{x})$ (for example, symmetry, sparseness, etc.).

The use of the Newton method for nonlinear systems implies differentiability of the functions $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, …, $f_n(\mathbf{x})$ and nondegeneracy of the Jacobi matrix ($\det \mathbf{J}(\mathbf{x}^{(k)}) \neq 0$) on every iteration. Then, if the initial approximation $\mathbf{x}^{(0)} \in G$ is chosen in a sufficiently close neighborhood of the desired root, the iterations of the Newton method converge to the exact solution $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T$, and the convergence is quadratic.

In practical calculations, the criterion $\varepsilon^{(k)} = \left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\| \leq \varepsilon$ is usually used as a condition for the iterations end, where ε is the accuracy specified by the calculator.

Example. Find a positive solution to the system of nonlinear equations with an accuracy of $\varepsilon = 10^{-4}$ using the Newton method:

$$\begin{cases} f_1(x_1, x_2) = 0.1x_1^2 + x_1 + 0.2x_2^2 - 0.3 = 0 \\ f_2(x_1, x_2) = 0.2x_1^2 + x_2 - 0.1x_1x_2 - 0.7 = 0 \end{cases}.$$

The solution will be sought for in the domain $G = \{0 \leq x_1 \leq 0.5;\ 0.5 \leq x_2 \leq 1\}$.

Let's take $x_1^{(0)} = 0.25,\ x_2^{(0)} = 0.75$ as the initial approximation.

For a system of two equations, the calculation formulas of the Newton method are conveniently written in the form solved relatively to $x_1^{(k+1)}$, $x_2^{(k+1)}$, according to the Cramer's rule for solving SLAE:

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} - \dfrac{\det \mathbf{A}_1^{(k)}}{\det \mathbf{J}^{(k)}} \\ x_2^{(k+1)} = x_2^{(k)} - \dfrac{\det \mathbf{A}_2^{(k)}}{\det \mathbf{J}^{(k)}} \end{cases}, \; k = 0,1,2,..., \text{ where the Jacobi matrix}$$

$$\mathbf{J}^{(k)} = \begin{bmatrix} \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \\ \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \end{bmatrix} \text{ and matrices}$$

$$\mathbf{A}_1^{(k)} = \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) & \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \\ f_2(x_1^{(k)}, x_2^{(k)}) & \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \end{bmatrix}, \; \mathbf{A}_2^{(k)} = \begin{bmatrix} \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & f_1(x_1^{(k)}, x_2^{(k)}) \\ \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix}.$$

As a result, using the determinant formula for matrices of 2×2 dimension, the subsequent approximations are determined as follows:

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} - \dfrac{f_1(x_1^{(k)}, x_2^{(k)}) \cdot \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} - f_2(x_1^{(k)}, x_2^{(k)}) \cdot \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2}}{\dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} \cdot \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} - \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} \cdot \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2}} \\ x_2^{(k+1)} = x_2^{(k)} - \dfrac{f_2(x_1^{(k)}, x_2^{(k)}) \cdot \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} - f_1(x_1^{(k)}, x_2^{(k)}) \cdot \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1}}{\dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} \cdot \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} - \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} \cdot \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2}} \end{cases}.$$

In the example under consideration:

$$f_1(x_1^{(k)}, x_2^{(k)}) = 0.1x_1^{(k)2} + x_1^{(k)} + 0.2x_2^{(k)2} - 0.3,$$

$$f_2(x_1^{(k)}, x_2^{(k)}) = 0.2x_1^{(k)2} + x_2^{(k)} - 0.1x_1^{(k)}x_2^{(k)} - 0.7,$$

$$\frac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} = 0.2x_1^{(k)} + 1, \; \frac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} = 0.4x_2^{(k)},$$

$$\frac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} = 0.4x_1^{(k)} - 0.1x_2^{(k)}, \; \frac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} = 1 - 0.1x_1^{(k)}.$$

Substituting the chosen values $x_1^{(0)}, x_2^{(0)}$ into the right-hand sides of the calculation formulas, we obtain the approximation $x_1^{(1)}, x_2^{(1)}$, which, in turn, is used to find $x_1^{(2)}, x_2^{(2)}$, etc.

Iterations stop when the following condition is met:

$\varepsilon^{(k+1)} = \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \le \varepsilon$, где $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| = \max\limits_i \left|x_i^{(k+1)} - x_i^{(k)}\right|$.

The calculation results are presented in table 6.

The final approximation of the root is $x_1^* \approx 0.1964, x_2^* \approx 0.7062$.

Table 6

k	$x_1^{(k)}$ $x_2^{(k)}$	$f_1(x_1^{(k)}, x_2^{(k)})$ $f_2(x_1^{(k)}, x_2^{(k)})$	$\frac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1}$ $\frac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1}$	$\frac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2}$ $\frac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2}$	$\varepsilon^{(k)}$
0	0.25000 0.75000	0.06875 0.04375	1.05000 0.02500	0.30000 0.97500	–
1	0.19696 0.70649	0.00066 0.00033	1.03939 0.00813	0.28260 0.98030	0.05304
2	0.19641 0.70615	0.00000 0.00000	1.03928 0.00795	0.28246 0.98036	0.00054
3	**0.19641** **0.70615**				**0.00000**

3.14. PROGRAM #11

Below is a proposed variant of the program algorithm for refining the value of the root of the system of two equations using the Newton method.

```
ALGORITHM “Newton method”
INPUT      f(,), g(,), f1(,), g1(,), f2(,), g2(,), a, b, c, d, e
OUTPUT     r, s, u, v, x, y, k
BEGIN
```

```
        x:=(a+b)/2, y:=(c+d)/2
        r:=2*e
        k:=0
        CYCLE “Iteration” (WHILE r>e)
                u:=x, v:=y
                s:=f1(u,v)*g2(u,v)-f2(u,v)*g1(u,v)
                IF (s=0) END
                x:=u-(f(u,v)*g2(u,v)-g(u,v)*f2(u,v))/s
                y:=v-(f1(u,v)*g(u,v)-f(u,v)*g1(u,v))/s
                IF (x>u) r:=x-u ELSE r:=u-x
                IF (y>v) s:=y-v ELSE s:=v-y
                IF (s>r) r:=s
                k:=k+1
        PRINT x, y, k
END
```