Решение систем нелинейных уравнений

Дана система нелинейных уравнений

$$\begin{cases} f_1(x_1, x_2, x_3, \ldots, x_n) = 0, \\ f_2(x_1, x_2, x_3, \ldots, x_n) = 0, \\ \ldots \\ f_n(x_1, x_2, x_3, \ldots, x_n) = 0, \end{cases} \quad (1)$$

Или $f_i(x_1, x_2, x_3, \ldots, x_n) = 0, i = \overline{1 \ldots n}.$

Необходимо решить эту систему, т.е. найти вектор , удовлетворяющий всем уравнениям системы (1) с точностью ε.

Вектор X определяет точку в n-мерном Евклидовом пространстве, т.е. $\bar{X} \in$ этому пространству и удовлетворяет всем уравнениям системы (1).

В отличие от систем линейных уравнений для систем нелинейных уравнений неизвестны прямые методы решения. При решении систем нелинейных уравнений используются итерационные методы.

Эффективность всех итерационных методов зависит от выбора начального приближения (начальной точки), т.е. вектора $\overline{X^0}$.

.

Приступая к задаче решения системы нелинейных уравнений, необходимо убедиться в существовании решения и количестве корней, а также выбрать нулевое приближение.

- в случае системы двух уравнений с двумя неизвестными это можно сделать, построив графики функций в удобных координатах.
- В случае сложных функций можно посмотреть поведение полиномов, их аппроксимирующих.
- Для трех и более неизвестных, а также для комплексных корней, удовлетворительных способов подбора начального приближения нет.

Область, в которой начальное приближение $\overline{X^0}$ сходится к искомому решению, называется **областью сходимости G**. Если начальное приближение $\overline{X^0}$ лежит за пределами G, то решение системы получить не удается.

Выбор начальной точки $\overline{X^0}$ во многом определяется интуицией и опытом специалиста.

Метод простых итераций

Для применения этого метода исходная система (1) должна быть преобразована к виду

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, x_3, \ldots, x_n), \\ x_2 = \varphi_2(x_1, x_2, x_3, \ldots, x_n), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = \varphi_n(x_1, x_2, x_3, \ldots, x_n), \end{cases} \qquad (2)$$

Или $x_i = \varphi_i(x_1, x_2, x_3, \ldots, x_n), i = \overline{1, n}.$

Далее, выбрав начальное приближение $\overline{X^0} = [x_1^0, x_2^0, \ldots, x_n^0]$ и используя систему (2), строим итерационный процесс поиска по схеме:

$$x_i^k =_i \left(x_1^{k-1}, x_2^{k-1}, x_3^{k-1}, \ldots, x_n^{k-1}\right),$$

т.е. на каждом k-ом шаге поиска вектора переменных находим $\overline{X}$, используя значения переменных, полученные на шаге (k-1).

Итерационный процесс поиска прекращается как только выполнится условие

$$\left|x_j^k - x_j^{k-1}\right| \leq \epsilon, j = \overline{1,n}. \quad (3)$$

При этом условие (3) должно выполняться одновременно по всем переменным. Метод простых итераций используется для решения таких систем нелинейных уравнений, в которых выполняется **условие сходимости итерационного процесса поиска**, а именно:

$$\sum_{i=1}^{n} \left|\frac{\delta\varphi_i}{\delta x_j}\right| < 1, j = \overline{1,n}. \quad (4)$$

т.е. сумма абсолютных величин частных производных всех преобразованных уравнений системы (2) по j-ой переменной меньше единицы. В более общей форме **достаточное условие сходимости**: метод простых итераций сходится к решению системы (2), если какая-либо норма матрицы Якоби, построенной по правым частям системы (2), меньше 1 на каждой итерации. Метод Зейделя имеет аналогичные условия сходимости.

Для метода Зейделя алгоритм итерационных вычислений следующий:

$$\begin{cases} x_1^{(k+1)} = \Phi_1\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \\ x_2^{(k+1)} = \Phi_2\left(x_1^{(k+1)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = \Phi_n\left(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{n-1}^{(k+1)}, x_n^{(k)}\right). \end{cases}$$

т.е. на каждом (k+1)-ом шаге поиска вектора переменных находим $\overline{X}$, используя значения переменных, полученные на предыдущем шаге и на этом же шаге. Метод сходится, если на каждом итерационном шаге выполняется условие:

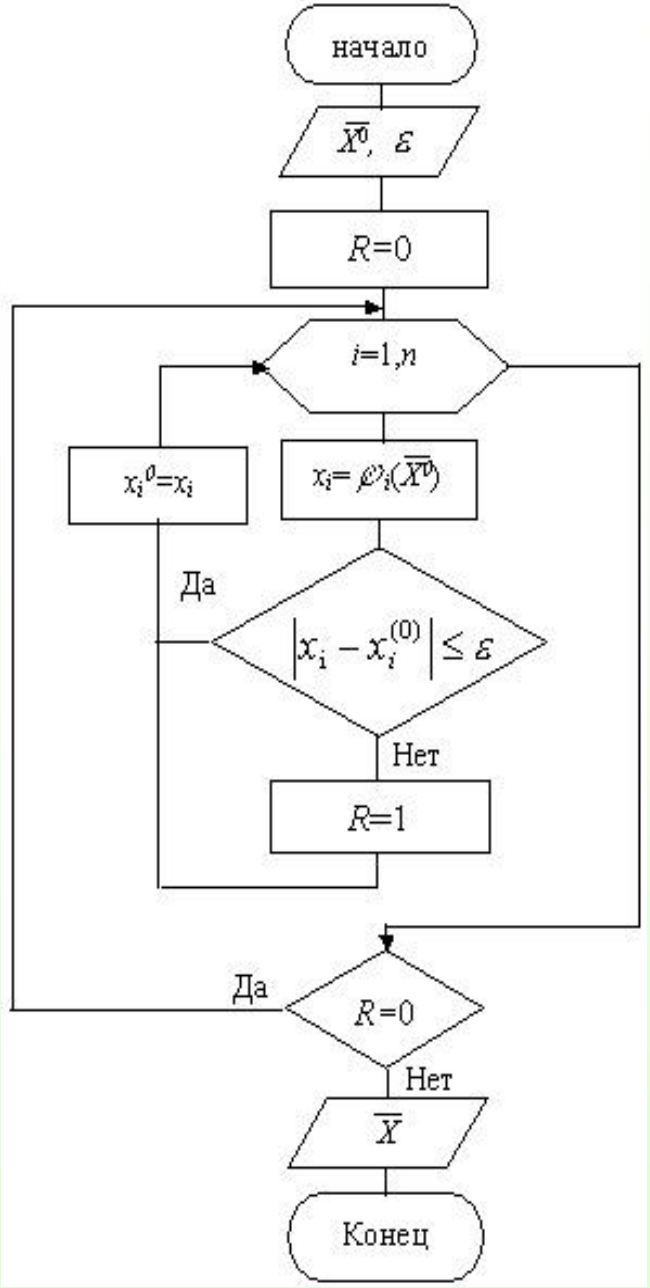
$$\left\|J\left(x^{(k)}\right)\right\| < 1, \quad J\left(x^{(k)}\right) = \left[\frac{\partial \Phi_i\left(x^{(k)}\right)}{\partial x_j}\right]_{x=x^k},$$

$$i, j = \overline{1, n}, \quad k = 0,\ 1,\ 2, \ldots$$

Вычислительный процесс останавливается при выполнении неравенства:

$$\left\|x^{(k+1)} - x^{(k)}\right\| \leqslant \varepsilon.$$

Рис. 1. Блок-схема алгоритма метода простых итераций



Рассмотрим пример.
Дана система нелинейных уравнений:

$$\begin{cases} x_1^2 + x_2^2 = 1 \\ lnx_1 + 2x_2 = -1 \end{cases}$$

Необходимо определить область сходимости системы, выбрать начальную точку и найти одно из решений системы.
1.Строим графики уравнений:

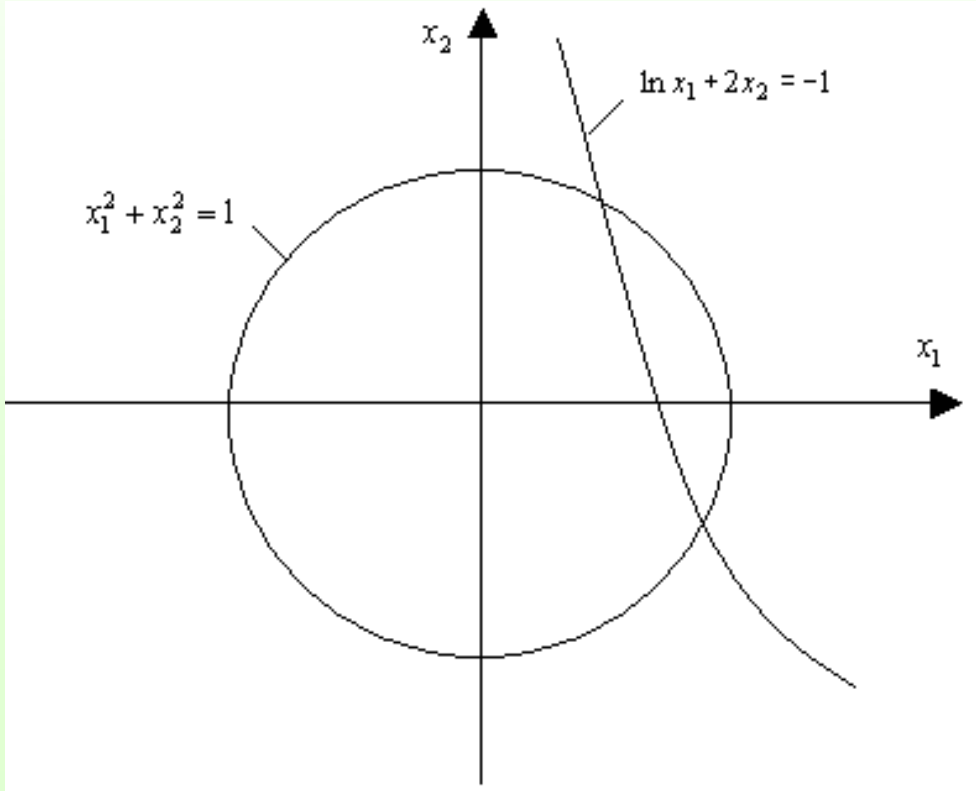


Рис. 2.

1.Преобразуем систему для решения методом итераций:

$$\begin{cases} x_1 = \sqrt{1 - x_2^2} \to \varphi_1(x_1, x_2), \\ x_2 = -0,5 - 0,5 ln x_1 \to \varphi_2(x_1, x_2). \end{cases}$$

Проверяем условие сходимости (4). Для заданной системы оно имеет вид:

$$|\delta\varphi_1/\delta x_1| + |\delta\varphi_2/\delta x_1| < 1$$
$$|\delta\varphi_1/\delta x_2| + |\delta\varphi_2/\delta x_2| < 1$$

Находим:

$$\delta\varphi_1/\delta x_1 = 0;$$
$$\delta\varphi_1/\delta x_2 = -x/\sqrt{1 - x_2^2};$$
$$\delta\varphi_2/\delta x_1 = -1/(2x_1);$$
$$\delta\varphi_2/\delta x_2 = 0$$

В результате условие (4) будет иметь вид:

$$|0| + |1/(2x_1)| < 1,$$
$$|x_2/\sqrt{1 - x_2^2}| + |0| < 1.$$

Определяем область сходимости G.

Граница области сходимости определится при решении системы,

$$\begin{cases} 1/(2x_1) = 1; \\ x_2/\sqrt{1 - x_2^2} = 1. \end{cases}$$

Отсюда x_1=0,5; $x_2 = \pm\sqrt{0,5}$

В результате область сходимости определится при $|x_1| \geq 0.5$ и $-0.707 \leq x_2 \leq 0.707.$ На графике уравнений строим область сходимости G:

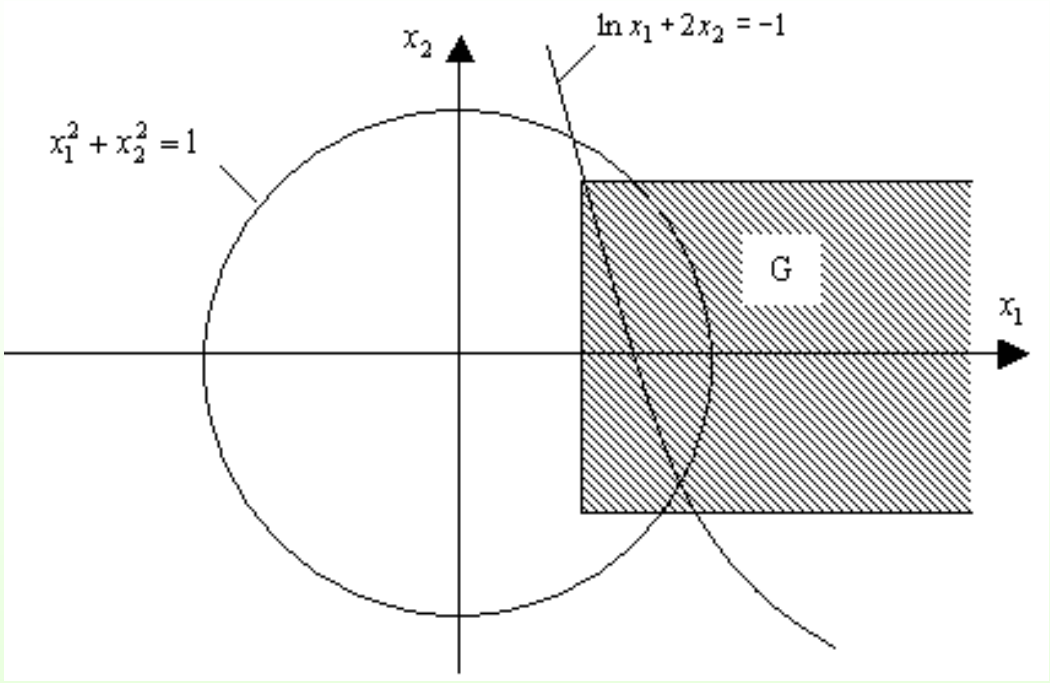


Рис. 3

Выбираем начальную точку $\overline{X^0} = [0.8; -0.6]$, принадлежащую области сходимости G. Используя выбранную начальную точку $\overline{X^0} = [0.8; -0.6]$ решаем заданную систему нелинейных уравнений.

Решение систем нелинейных уравнений методом Ньютона

Дана система нелинейных уравнений

$$\begin{cases} f_1(x_1, x_2, x_3, \ldots, x_n) = 0, \\ f_2(x_1, x_2, x_3, \ldots, x_n) = 0, \\ \ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, x_3, \ldots, x_n) = 0, \end{cases} \quad (5)$$

или $f_1(x_1, x_2, x_3, \ldots, x_n) = 0, i = \overline{1 \ldots n}.$

Необходимо решить эту систему, т.е. найти вектор $\bar{X} = [x_1, x_2, x_3, \ldots, x_n]$ удовлетворяющий всем уравнениям системы (5) с точностью ε.

Метод Ньютона - наиболее распространенный метод решения систем нелинейных уравнений. Он обеспечивает более быструю сходимость по сравнению с методом итераций.

В основе метода Ньютона лежит идея линеаризации всех нелинейных уравнений системы (5). Сообщим всей системе (5) малые приращения h_j и разложим каждое уравнение системы (5) в ряд Тейлора:

$$\begin{cases} f_1(x_1+h_1, x_2+h_2, \ldots, x_n+f_n) = f_1(x_1, x_2, \ldots, _n) + h_1\frac{\delta f_1}{\delta x_1} + \\ +h_2\frac{\delta f_1}{\delta x_2} + \ldots + h_n\frac{\delta f_1}{\delta x_n} + R_1, \\ f_2(x_1+h_1, x_2+h_2, \ldots, x_n+f_n) = f_2(x_1, x_2, \ldots, _n) + h_1\frac{\delta f_2}{\delta x_1} + \\ +h_2\frac{\delta f_2}{\delta x_2} + \ldots + h_n\frac{\delta f_2}{\delta x_n} + R_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1+h_1, x_2+h_2, \ldots, x_n+f_n) = f_n(x_1, x_2, \ldots, _n) + h_1\frac{\delta f_n}{\delta x_1} + \\ +h_2\frac{\delta f_n}{\delta x_2} + \ldots + h_n\frac{\delta f_n}{\delta x_n} + R_n, \end{cases} \quad (6)$$

где h_j- приращение по каждой x_j; R_i - остаточные нелинейные члены второго и более высоких порядков каждого ряда Тейлора. Если приращения h_j таковы, что переменные x_j принимают значения близкие к корню, то будем считать, что левые части уравнений системы (6) обращаются в нули. Тогда отбросив R_i сведем задачу решения системы нелинейных уравнений (5) к решению системы линейных уравнений, в которой неизвестными являются приращения h_j, $j = \overline{1,n}$

$$\begin{cases} h_1\frac{\delta f_1}{\delta x_1} + h_2\frac{\delta f_1}{\delta x_2} + \ldots + h_n\frac{\delta f_1}{\delta x_n} + R_1 = -f_1(x_1, x_2, \ldots, x_n), \\ h_1\frac{\delta f_2}{\delta x_1} + h_2\frac{\delta f_2}{\delta x_2} + \ldots + h_n\frac{\delta f_2}{\delta x_n} + R_2 = -f_2(x_1, x_2, \ldots, x_n), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_1\frac{\delta f_n}{\delta x_1} + h_2\frac{\delta f_n}{\delta x_2} + \ldots + h_n\frac{\delta f_n}{\delta x_n} + R_n = -f_n(x_1, x_2, \ldots, x_n). \end{cases} \quad (7)$$

Система (7) – система линейных уравнений с неизвестными h_j, $j = \overline{1, n_j}$

Запишем (7) в матричной форме: $A \cdot \bar{H} = \bar{B}$, где

$$A = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_2} & \dots & \frac{\delta f_1}{\delta x_n} \\ \frac{\delta f_2}{\delta x_1} & \frac{\delta f_2}{\delta x_2} & \dots & \frac{\delta f_2}{\delta x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\delta f_n}{\delta x_1} & \frac{\delta f_n}{\delta x_2} & \dots & \frac{\delta f_n}{\delta x_n} \end{bmatrix} \text{ – матрица коэффициентов системы,}$$

$$\bar{B} = \begin{bmatrix} -f_1 \\ -f_2 \\ \dots \\ -f_n \end{bmatrix} \text{ – вектор свободных членов,}$$

$$\bar{H} = \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_n \end{bmatrix} \text{ – вектор неизвестных системы} \quad (7)$$

Матрица A, составленная из частных производных

$$a_{ij} = \frac{\delta f_i}{\delta x_j}; i = \overline{1, n}; j = \overline{1, n}$$

называется **матрицей Якоби,** а её определитель - **Якобианом**.

Метод Ньютона состоит из двух этапов.

На первом этапе реализации метода Ньютона необходимо построить систему (7).

На втором этапе, начиная с начальной точки $\overline{X^0}$, необходимо решать систему (7) на каждом шаге итерационного процесса поиска методом Гаусса. Найденные значения приращений h_i используются как поправки к решению, полученному на предыдущем шаге поиска, т.е. $H = A^{-1}B$

$$\begin{aligned} x_1^{(k+1)} &= x_1^{(k)} + h_1^{(k)}, \\ x_2^{(k+1)} &= x_2^{(k)} + h_2^{(k)}, \\ &\cdots\cdots\cdots \\ x_n^{(k+1)} &= x_n^{(k)} + h_n^{(k)}, \end{aligned} \qquad (8)$$

или $x_j^{(k+1)} = x_j^{(k)} + h_j^{(k)} : j = \overline{1,n}.$

Итерационный процесс прекращается, как только выполнится условие

$$\begin{matrix} |h_j| \leq \varepsilon; \\ j = \overline{1,n} \end{matrix} \qquad (9)$$

по всем приращениям одновременно.

Определение матрицы Якоби

В методе Ньютона на каждом шаге итерационного процесса поиска необходимо формировать матрицу Якоби, при этом каждый элемент матрицы можно определить:

1. аналитически, как частную производную $\frac{\delta f_i}{\delta x_j}$,
2. методом численного дифференцирования, как отношение приращения функции к приращению аргумента, т.е. $\frac{\delta f_i}{\delta x_j} \approx \frac{\Delta f_i}{\Delta x_j}$

В результате частная производная $f_i(\bar{X})$ по первой координате x_1 определится как

$$\frac{\delta f_i}{\delta x_1} \approx \frac{f_i(x_1 + \Delta x_1, x_2, x_3 \dots x_n) - f_i(x_1, x_2, \dots, x_n)}{\Delta x_1},$$

а частная производная $f_i(\bar{X})$ по координате x_j определится как

$$\frac{\delta f_i}{\delta x_j} \approx \frac{f_i(x_1, x_2, \dots, x_j + \Delta x_j \dots x_n) - f_i(x_1, x_2, \dots, x_n)}{\Delta x_j},$$

где $\Delta x_j \approx \varepsilon$.

Метод Ньютона имеет **преимущества** по сравнению с другими методами. Его достоинством является высокая скорость сходимости. Но с увеличением числа неизвестных область сходимости уменьшается, а в случае больших систем, сходимость обеспечивается, если начальная точка близка к искомому решению. Сходимость метода зависит от удачности выбора начального приближения:

если $\det\left[\frac{\partial \vec{f}}{\partial \vec{x}}\right] \neq 0$ то итерации сходятся к корню.

Недостатком метода является вычислительная сложность: на каждой итерации требуется находить матрицу Якоби и решать систему линейных уравнений. Кроме того, если аналитический вид частных производных неизвестен, их надо получать по формулам приближенного дифференцирования, например, как отношение приращения функции к приращению аргумента:

$$F_{ji} = \frac{F_j(x_{10}, x_{20}, \ldots, x_{i0} + \varepsilon, \ldots, x_{n0}) - F_j(x_{10}, x_{20}, \ldots, x_{i0}, \ldots x_{n0})}{\varepsilon},$$

где ε– достаточно малое число.

Таким образом, для применения метода Ньютона необходимо выполнение следующих двух условий:

1. существование частных производных первого порядка, входящих в матрицу Якоби

2. матрица Якоби для каждой итерации должна быть невырождена

$$\det\left[\frac{\partial \vec{f}}{\partial \vec{x}}\right] \neq 0$$

Так как метод Ньютона отличается высокой скоростью сходимости при выполнении условий сходимости, на практике критерием работоспособности метода является число итераций:
если оно оказывается большим (для большинства задач >100), то начальное приближение выбрано плохо.

Методы контроля сходимости итерационных методов решения нелинейных систем

1. Норма (эвклидова или -максимум) вектора невязок $\sqrt{F_{i1}^2 + F_{i2}^2 + \ldots + F_{in}^2} \le \varepsilon$

2. Эвклидова норма вектора относительных отклонений переменных

$$\sqrt{\left(\frac{\Delta x_1}{x_1}\right)^2 + \left(\frac{\Delta x_2}{x_2}\right)^2 + \ldots + \left(\frac{\Delta x_n}{x_n}\right)^2} \le \varepsilon$$

3. Норма-максимум вектора относительных отклонений $\max \left|\frac{\Delta x_i}{x_i}\right| \le \varepsilon$

4. $| x_i^k - x_i^{k-1} | \le \varepsilon, \; i = 1,\ldots,n$

Пример.

Методом Ньютона приближенно найти положительное решение системы уравнений

$$\begin{cases} f_1(x,y,z) = x^2 + y^2 + z^2 - 1, \\ f_2(x,y,z) = 2x^2 + y^2 - 4z, \\ f_3(x,y,z) = 3x^2 - 4y + z^2. \end{cases}$$

исходя из начального приближения $x_0 = y_0 = z_0 = 0{,}5$.

Полагая: $\boldsymbol{x}^{(0)} = \begin{bmatrix} 0{,}5 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}$, $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \end{bmatrix}$, имеем: $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x^2 + y^2 + z^2 - 1 \\ 2x^2 + y^2 - 4z \\ 3x^2 - 4y + z^2 \end{bmatrix}$

Отсюда: $\boldsymbol{f}(\boldsymbol{x}^{(0)}) = \begin{bmatrix} 0{,}25 + 0{,}25 + 0{,}25 - 1 \\ 0{,}50 + 0{,}25 - 2{,}00 \\ 0{,}75 - 2{,}00 + 0{,}25 \end{bmatrix} = \begin{bmatrix} -0{,}25 \\ -1{,}25 \\ -1{,}00 \end{bmatrix}$.

Составим матрицу Якоби:

$$W(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} = \begin{bmatrix} 2x & 2y & 2z \\ 4x & 2y & -4 \\ 6x & -4 & 2z \end{bmatrix}$$

Имеем:

$$W(\boldsymbol{x}^{(0)}) = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{bmatrix}, \quad \text{причем } \Delta = \det W(\boldsymbol{x}^{(0)}) = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{bmatrix} = -40.$$

Следовательно, матрица $W(\boldsymbol{x}^{(0)})$ - неособенная. Составим обратную ей матрицу:

$$W^{-1}(\boldsymbol{x}^{(0)}) = -\frac{1}{40}\begin{bmatrix} -15 & -5 & -5 \\ -14 & -2 & 6 \\ -11 & 7 & -1 \end{bmatrix} = \begin{bmatrix} \frac{3}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{7}{20} & \frac{1}{20} & -\frac{3}{20} \\ \frac{11}{40} & -\frac{7}{40} & \frac{1}{40} \end{bmatrix}.$$

По формуле получаем первое приближение: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - W^{-1}(\boldsymbol{x}^{(k)})\, \boldsymbol{f}(\boldsymbol{x}^{(k)})$ $(k = 0, 1, 2, \ldots)$.

$$x^{(1)} = x^{(0)} - W^{-1}(x^{(0)})\, f(x^{(0)}) = \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \end{bmatrix} - \begin{bmatrix} \frac{3}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{7}{20} & \frac{1}{20} & -\frac{3}{20} \\ \frac{11}{40} & -\frac{7}{40} & \frac{1}{40} \end{bmatrix} \begin{bmatrix} -0,25 \\ -1,25 \\ -1,00 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \end{bmatrix} + \begin{bmatrix} 0,375 \\ 0 \\ -0,125 \end{bmatrix} = \begin{bmatrix} 0,875 \\ 0,500 \\ 0,375 \end{bmatrix}.$$

Аналогично находятся дальнейшие приближения. Результаты вычислений приведены в Таблице.

Таблица. Последовательные приближения корней

i	x	y	z
0	0,5	0,5	0,5
1	0,875	0,5	0,375
2	0,78981	0,49662	0,36993
3	0,78521	0,49662	0,36992

Останавливаясь на приближении $x^{(3)}$, будем иметь:

$$x = 0,7852;\ y = 0,4966;\ z = 0,3699.$$

Таким образом, алгоритм решения задачи методом Ньютона выглядит следующим образом:

1. Выбирается начальное приближение x_0, система приводится к стандартному виду f(x) = 0.
2. Рассчитывается матрица Якоби значений частных производных в точке начального приближения $W(x_0)$, вычисляется её определитель (Якобиан) $\det W(x_0)$ и, если он не равен нулю, то переходим к п.3.
3. Решается система линейных уравнений относительно приращений переменных $\Delta x^{(k)} = - W^{-1}(x^{(k)}) f(x^{(k)})$.
4. К вектору начального приближения прибавляется вектор приращений.

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

5. Проверяется условие сходимости к решению и, если оно не достигнуто, то процедура повторяется с п.2 .

$$| x_i^{k+1} - x_i^k | \le \varepsilon, \quad i = 1,\ldots,n$$

Пример.

Построить метод простых итераций и итерационный процесс Ньютона для системы уравнений

$$f(x, y) = x + 3 \lg x - y^2 = 0,$$

$$g(x, y) = 2x^2 - xy - 5x + 1 = 0.$$

Решение. **Графики функций** $f(x, y)$ **и** $g(x, y)$ **приведены на рис. 5.12.**

Метод простых итераций запишем как

$$x_{k+1} = y_k^2 - 3 \lg x_k,$$

$$y_{k+1} = 2x_k + \frac{1}{x_k} - 5,$$

где $x_0 = 3,4, y_0 = 2,2, F(x, y) = y^2 - 3 \lg x; G(x, y) = 2x + \frac{1}{x} - 5.$

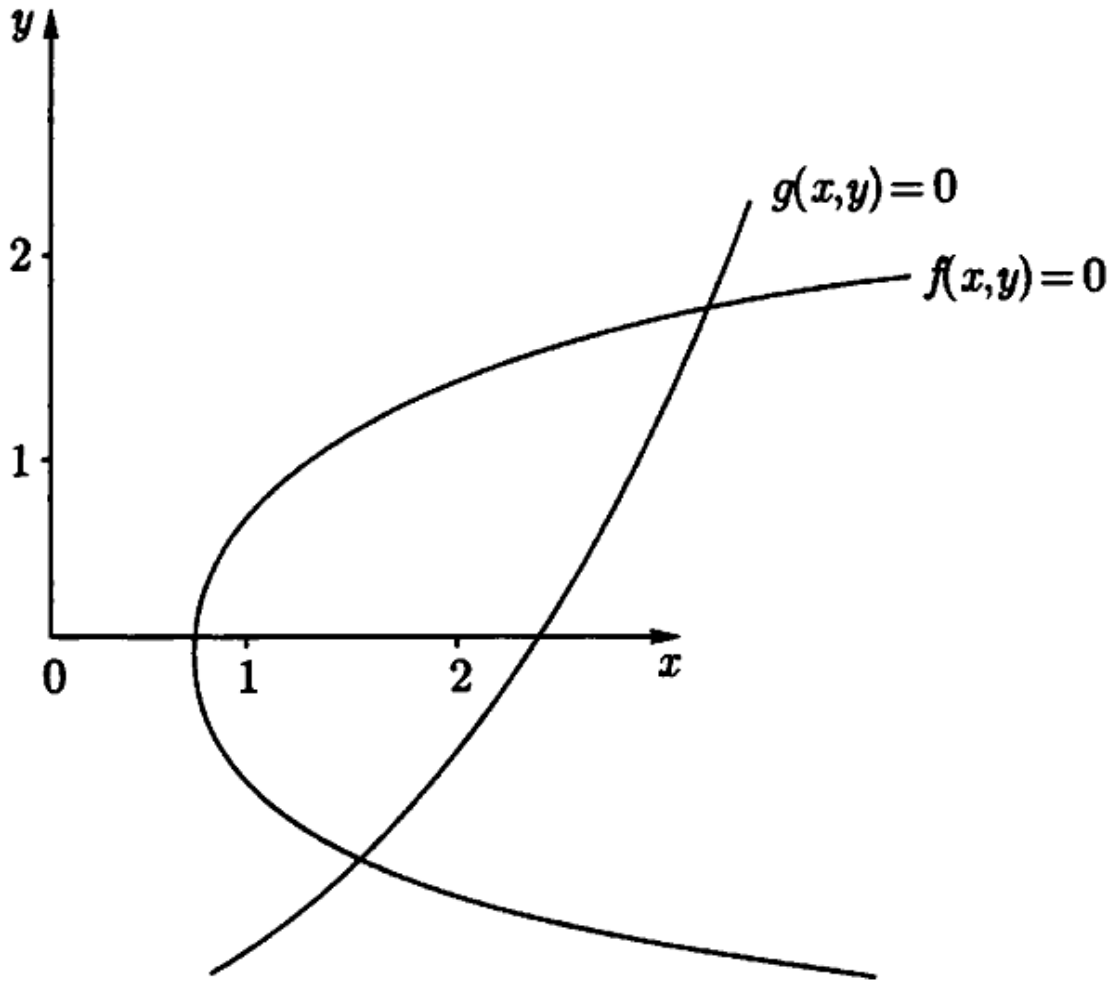


Рис. 5.12

Матрица Якоби для такого процесса запишется как

$(\lg x)' = 1/x * \lg e$

$$\mathbf{J} = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \end{pmatrix} = \begin{pmatrix} -\frac{3\lg e}{x} & 2y \\ 2 - x^{-2} & 0 \end{pmatrix}.$$

Несложно проверить, вычислив норму матрицы Якоби, что для приведенного начального приближения достаточное условие сходимости не выполняется.

Рассмотрим другой итерационный процесс:

$$x_{k+1} = \left(\frac{x_k (y_k + 5) - 1}{2} \right)^{1/2},$$

$$y_{k+1} = (x_k + 3\lg x_k)^{1/2},$$

$$F(x,y) = \left(\frac{x_k (y_k + 5) - 1}{2} \right)^{1/2}, G(x,y) = (x + 3\lg x)^{1/2}.$$

Матрица Якоби для этого итерационного метода будет

$$\mathbf{J} = \begin{pmatrix} \frac{5+y}{2\sqrt{2}\sqrt{x(y+5)-1}} & \frac{x}{2\sqrt{2}\sqrt{x(y+5)-1}} \\ \frac{1+\frac{3\lg e}{x}}{2\sqrt{x+3\lg x}} & 0 \end{pmatrix}.$$

В окрестности начального приближения условие сходимости выполнено. Таблица первых пяти приближений будет

k	0	1	2	3	4	5
x	3,4	3,426	3,451	3,466	3,475	3,480
y	2,2	2,243	2,2505	2,255	2,258	2,259

Возможности MATLAB для решения систем нелинейных уравнений

Решение нелинейных уравнений и систем выполняется с помощью функции **fsolve**.

>> fsolve(< имя функции>, <начальное значение>),

Где < имя функции> - имя М-функции, которая определяет левую часть уравнения f(x)=0 или системы уравнений F(x)=0: она должна принимать на входе вектор аргументов и возвращать вектор значений; <начальное значение> - вектор приближений, относительно которого будет осуществляться поиск решения.

>> **fsolve**(< имя функции>, <начальное значение>,<опции>),

Где <опции> - дополнительные параметры управления вычислительным процессом. Их можно определить с помощью функции

optimset (<вид контроля>, <значение>).

Установление принятых по умолчанию настроек:

>>fsolve('fun',x0,optimset('fsolve'))

Если в качестве параметра использовать функцию optimset(‘Display’,’iter’), то будет выводиться информация о каждом шаге вычислительного процесса, а функция optimset(‘Display’,’final’) выведет информацию только о завершении процесса.

Отключение вывода информации вообще: optimset(‘Display’,’off’).

Параметр optimset(‘Display’,’notify’) установлен по умолчанию.

Функция optimset(‘TolX’,’1.0e-8’) установит точность вычислительного процесса. Одновременное задание точности и вывода информации о вычислительном процессе: optimset(‘Display’,’iter’,‘TolX’,’1.0e-8’):

>> x = fsolve(fun,x0, optimset('Display','iter','TolFun',1e-8))

>>[x,fval,exitflag] =fsolve(fun,x0)

exitflag – условия окончания вычислительного процесса (1 – выч. процесс сошёлся к решению); **fval** – значение целевой функции (критерий окончания процесса поиска решения)

>>[x,fval,exitflag,output] = fsolve(...)

output- вывод информации об окончании процесса

>>[x,fval,exitflag,output,jacobian] = fsolve(...)

jacobian- вывод Якобиана в точке решения **x**.

jacobian(f,v) - расчёт Якобиана вектора функций **f** с переменными, определяемыми вектором **v**.

Пример вычисления Якобиана.

```
>> syms x y z
>> f=[x*y*z; y^2; x+z];
>> v=[x,y,z];
>> R=jacobian(f,v)
>> D=det(R)
 R =
 [ y*z, x*z, x*y]
 [   0,   2*y,    0]
 [   1,     0,      1]
 D =  2*y^2*z-2*x*y^2
```

Задав конкретные числовые значения переменным **x,y,z** , используя функцию **subs**, можно вычислить Якобиан, его детерминант или норму:

```
>> NormaR=double norm(subs(subs(subs(R,x,x0),y,y0),z,z0) .
```

Решение систем нелинейных уравнений средствами символьной математики MATLAB

```
syms u v
eqns = [2*u^2 + v^2 == 0, u - v == 1];
vars = [v u];
[solv, solu] = solve(eqns,vars)
```

```
solv =
```

$$\begin{pmatrix} -\frac{2}{3}-\frac{\sqrt{2}\,\mathrm{i}}{3} \\ -\frac{2}{3}+\frac{\sqrt{2}\,\mathrm{i}}{3} \end{pmatrix}$$

```
solu =
```

$$\begin{pmatrix} \frac{1}{3}-\frac{\sqrt{2}\,\mathrm{i}}{3} \\ \frac{1}{3}+\frac{\sqrt{2}\,\mathrm{i}}{3} \end{pmatrix}$$

Для систем двух уравнений определенную помощь в поиске начального приближения может оказать функция **contour(Z),**

где Z – двумерный массив значений, или

contour(*x,y,* Z), где *x, y* – векторы аргументов для Z:

```
>> [x,y]=meshgrid(-2:0.2:2,-2:0.2:2); % задание поля аргументов;
>> Z=x.*exp(-x.^2-y.^2); % массив значений функции;
>> [C,h]=contour(x,y,Z); % вывод контура;
>> clabel(C,h) % вывод меток для линий уровня.
```

Например, возьмем функцию F(x,y) и построим для нее линии уровня:

```
>> [X,Y] = meshgrid(-2:.2:2,-2:.2:3);
>> Z = X.*exp(-X.^2-Y.^2);
>> [C,h] = contour(X,Y,Z);
>> set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
>> colormap cool
```

Пример . Найти решение системы нелинейных уравнений: $x_1^2 + x_2^2 = 1$

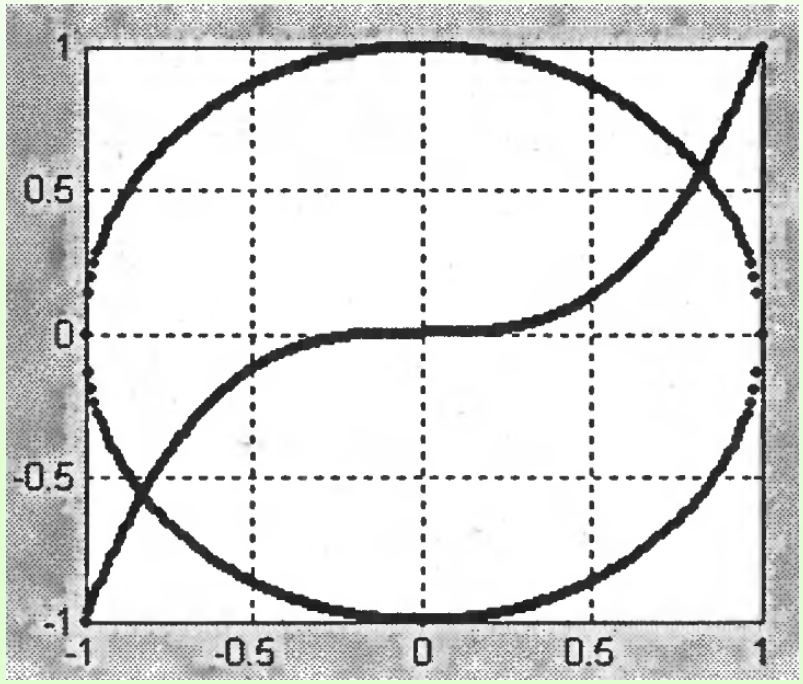
$x_1^3 - x_2 = 0$

Решим систему графически.

```
>> x=-1:0.01:1;   %Аргумент
>> y1=(1-x.^2).^(1/2);   %Первое уравнение, положительные значения
>> y2= -(1-x.^2).^(1/2);   %Первое уравнение, отрицательные значения
>> y3=x.^3;   %Второе уравнение
>> y=[y1;y2;y3];
>> plot(x,y, 'k. ')
>> grid on
```

или

```
>> ezplot('x1^2+x2^2-1')
>> hold on
>> ezplot('x1^3-x2')
>> grid on
```



Составим М-функцию **fun**.m, соответствующую левой части системы.

Важно помнить, что все уравнения должны иметь вид F(x)=0.

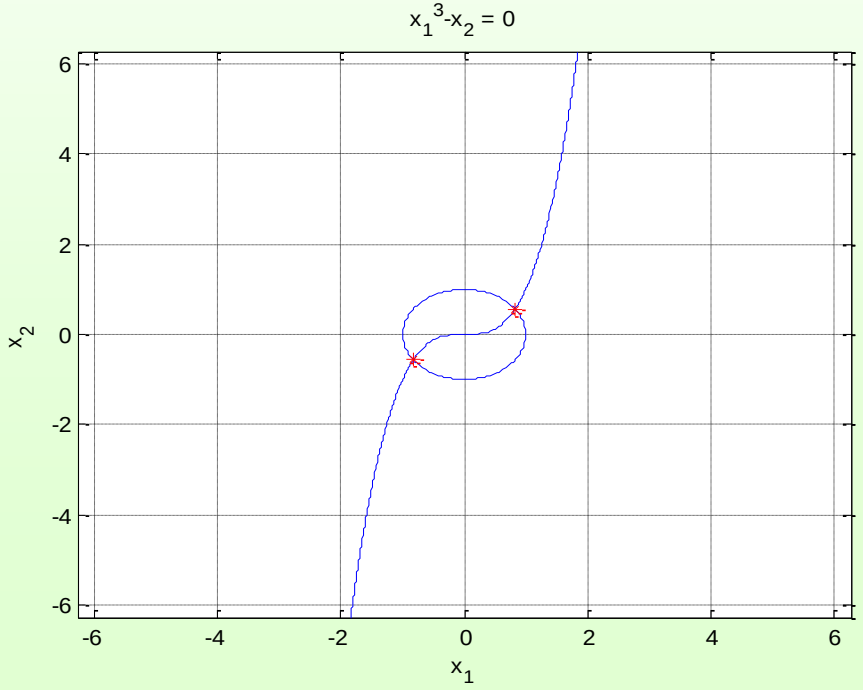
```
function f=fun(x)
f(1)=x(1)^2+x(2)^2-1;
f(2)=x(1)^3-x(2);
end
```

Теперь решим систему, указав в качестве начального приближения в начале вектор (1,1), затем (-1,-1).

```
>> [xr1,fr1,exitflag]=fsolve('fun',[1 1])
>> [xr2,fr2,exitflag]= fsolve('fun',[-1 -1])
```

Выведем на график полученные решения:

```
>> hold on
>> plot(xr1(1),xr1(2), '*r', xr2(1),xr2(2), '*r')
```



ЗАДАЧА 5.11. Решить систему нелинейных уравнений

$$x_1^2 + x_2^2 + x_3^2 = 1,$$
$$2x_1^2 + x_2^2 - 4x_3 = 0,$$
$$3x_1^2 - 4x_2 + x_3^2 = 0.$$

Листинг 5.4 содержит М-функцию заданной системы, а ее решение показано в листинге 5.5. Обратите внимание на выходные параметры функции `fsolve`. Здесь `x` – это решение системы, `f` – значение вектор-функции для найденного решения, а `ex` – признак завершения алгоритма решения нелинейной системы. Отрицательное значение параметра `ex` означает, что решение не найдено, ноль – досрочное прерывание вычислительного процесса при достижении максимально допустимого числа итераций, положительное значение подтверждает, что решение найдено с заданной точностью. В нашем случае значения функции для найденного решения близки к нулю и признак завершения положительный, значит, найдено верное решение.

Листинг 5.4

```
function f=Y(x)
f(1)=x(1)^2+x(2)^2+x(3)^2-1;
f(2)=2*x(1)^2+x(2)^2-4*x(3);
f(3)=3*x(1)^2-4*x(2)+x(3)^2;
end
```

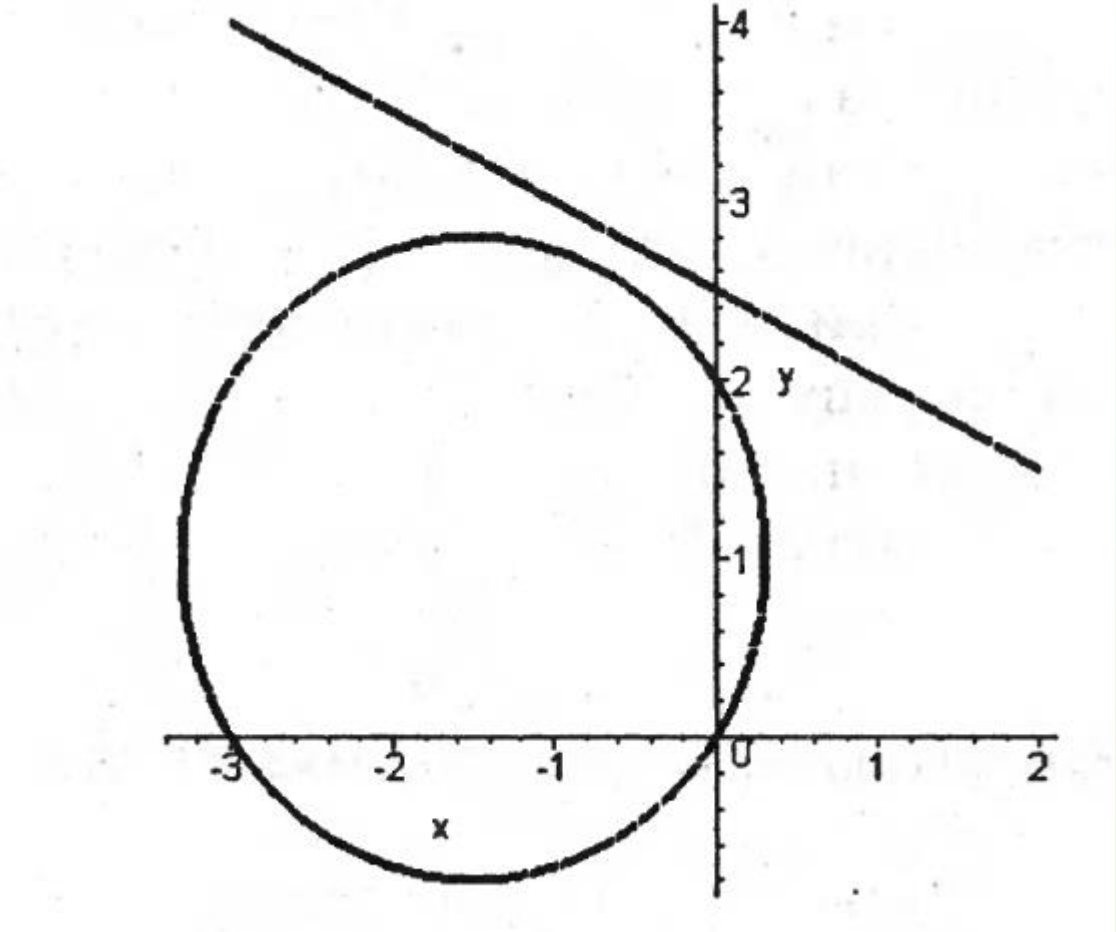
Листинг 5.5

```
>> [x,f,ex]=fsolve('Y',[0.5 0.5 0.5])
Optimization terminated: first-order optimality is less than options.TolFun.
x =
    0.7852    0.4966    0.3699
f =
  1.0e-009 *
    0.1828    0.3652    0.5480
ex =
     1
```

ЗАДАЧА 5.9. Решить систему уравнений

$$x^2 + y^2 + 3x - 2y = 0,$$
$$x + 2y = 5.$$

Система не имеет действительных решений, так как линии на графике (рис. 5.24) не пересекаются.



Графическое решение системы (рис. 5.24) показало, что она корней не имеет. Однако применение к системе функции `fsolve` дает положительный ответ, что видно из листинга 5.6. Происходит это потому, что алгоритм, реализованный в этой функции, основан на минимизации суммы квадратов компонент вектор-функции. Следовательно, наличие точки минимума не гарантирует существование корней системы в ее окрестности.

Листинг 5.6

```
function y=Fu(x)
y(1)=x(1)^2+x(2)^2-3*x(1)-2*x(2);
y(2)=x(1)+2*x(2)-5;
end
%-----------------
>> [x,f,ex]=fsolve("fu",[1 1])
Optimization terminated: first-order optimality is less than
options.TolFun.
x =
    0.3033    2.3483
f =
  1.0e-009 *
    0.2012    0.0001
ex =
     1
```

№1 Про юного купидона

Возможно вы не знали, что существует целая школа юных купидонов. Именно в такой школе купидона по имени Юра вызвали к доске на математике. Задача была следующая: «Предполагая, что профиль сердца можно описать уравнением $(x^2 + y^2 - 1)^3 - x^2 * y^3 = 0$, а стрела летит по траектории $y = -0.05x^2 - 0.5x + 0.25$, найдите точки, через которые войдёт и выйдет стрела любви». Юра сразу сообразил, что нужно составить систему уравнений:

$$\begin{cases} (x^2 + y^2 - 1)^3 = x^2 * y^3 \\ y = -0.05x^2 - 0.5x + 0.25 \end{cases}$$

Вот только решить её Юра не может, он не силён в математике, помогите же ему!

Необходимо решить заданную систему нелинейных уравнений методами простых итераций и Ньютона. Перед началом расчета необходимо построить графики, для того, чтобы определить начальную точку поиска. Делая задания, следует проверять выполнение условий сходимости методов. Фиксировать число требуемых итераций. Сделать проверку, подставив найденные корни в уравнения, а также решить уравнения, используя стандартные операторы MATLAB. Кроме того, необходимо подготовить отчёт, в котором будут отражены блок-схемы и коды алгоритмов, а так же приведены результаты работы.