

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»  
Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №2**  
**по курсу «Численные методы»**

Студент: Д. Д. Наумов  
Преподаватель: И. А. Иванов  
Группа: М8О-406Б-17  
Дата:  
Оценка:  
Подпись:

**Москва, 2021**

# Часть 1

## Задание

Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций.

## Вариант: 3

$$\sqrt{1-x^2} - e^x + 0,1 = 0$$

# Теория

## Метод Ньютона

При нахождении корня уравнения методом Ньютона, итерационный процесс определяется формулой

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

Для начала вычислений требуется задание начального приближения

**Теорема.** Пусть на отрезке  $[a, b]$  функция  $f(x)$  имеет первую и вторую производные постоянного знака и пусть  $f(a)f(b) < 0$ .

Тогда если точка  $x^{(0)}$  выбрана на  $[a, b]$  так, что

$$f(x^{(0)})f''(x^{(0)}) > 0$$

то начиная с неё последовательность  $\{x^{(k)}\}$ ,  $(k = 0, 1, 2, \dots)$ , определяемая методом Ньютона монотонно сходится к корню  $x^* \in [a, b]$  уравнения.

В качестве условия окончания итераций в практических вычислениях часто используется правило

$$|x^{(k+1)} - x^{(k)}| < \varepsilon \Rightarrow x^* \approx x^{(k+1)}$$

## Метод простой итерации

При использовании метода простой итерации уравнение заменяется эквивалентным уравнением с выделенным линейным членом

$$x = \varphi(x)$$

Решение ищется путем построения последовательности

$$x^{(k+1)} = \varphi(x^{(k)}), \quad k = 1, 2, \dots$$

начиная с некоторого заданного значения  $x^{(0)}$ . Если  $\varphi(x)$  - непрерывная функция, а  $x^{(k+1)} = \varphi(x^{(k)})$ ,  $k = 1, 2, \dots$  - сходящаяся последовательность, то значение  $x^* = \lim_{k \rightarrow \infty} x^{(k)}$  является решением уравнения.

**Теорема.** Пусть функция  $\varphi(x)$  - определена и дифференцируема на отрезке  $[a, b]$ . Тогда, если выполнены условия:

1.  $\varphi(x) \in [a, b], \forall x \in [a, b]$
2.  $\exists q : |\varphi'(x)| \leq q < 1, \forall x \in (a, b)$

то уравнение имеет единственный корень на  $[a, b]$ . К этому корню сходится определяемая методом простых итераций последовательность  $\{x^{(k)}\}$ ,  $(k = 0, 1, 2, \dots)$ , начиная с любого  $x^{(0)} \in [a, b]$ .

При этом справедливы оценки погрешности ( $\forall k \in N$ ):

$$|x^* - x^{(k+1)}| \leq \frac{q}{1-q} |x^{(k+1)} - x^{(k)}|$$
$$|x^* - x^{(k+1)}| \leq \frac{q^{k+1}}{1-q} |x^{(1)} - x^{(0)}|$$

# Реализация

## Метод Ньютона

```
1 double NewtonMethod (double x0, double eps, double (*f)(double), double(*difF)(double)
  ) {
2     double x = x0;
3     double next_x = x0;
4     do {
5         x = next_x;
6         next_x = x - f(x)/difF(x);
7     } while (std::abs(next_x - x) > eps);
8
9     return next_x;
10 }
```

## Метод простых итераций

```
1 double phi (double x) {
2     return std::log(std::sqrt(1 - x*x) + 0.1);
3 }
4
5 double SimpleIterMethod (double x0, double eps, double (*phi)(double)) {
6     double x = x0;
7     double next_x = x0;
8     do {
9         x = next_x;
10        next_x = phi(x);
11    } while (q / (1 - q) * std::abs(x - next_x) > eps);
12    return next_x;
13 }
```

## Результаты

### Пример работы программы

```
1 || (base) MacBook-Air-Dima:Program dandachok$ .\a.out
2 || Newton method: 0.0914902
3 || Simple iter method: 0.0915162
```

## Часть 2

### Задание

Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них, в котором значения неизвестных являются положительными); начальное приближение определить графически. Проанализировать зависимость погрешности вычислений от количества итераций.

### Вариант: 3

$$\begin{cases} (x_1^2 + 16)x_2 - 64 = 0 \\ (x_1 - 2)^2 + (x_2 - 2)^2 - 16 = 0 \end{cases}$$

# Теория

## Метод Ньютона

Если определено начальное приближение  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^T$ , итерационный процесс нахождения решения системы методом Ньютона можно представить в виде:

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \Delta x_1^{(k)} \\ x_2^{(k+1)} = x_2^{(k)} + \Delta x_2^{(k)} \\ \dots\dots\dots \\ x_n^{(k+1)} = x_n^{(k)} + \Delta x_n^{(k)} \end{cases}$$

где значения приращений  $\Delta x_1^{(k)}, \Delta x_2^{(k)}, \dots, \Delta x_n^{(k)}$  определяются из решения системы линейных алгебраических уравнений, все коэффициенты которой выражаются через известное предыдущее приближение  $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})^T$

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

В векторно-матричной форме расчетные формулы имеют вид

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

где вектор приращений  $\Delta \mathbf{x}^{(k)} = \begin{pmatrix} x^{(k)} \\ x^{(k)} \\ \vdots \\ x^{(k)} \end{pmatrix}$  находится из решения уравнения

$$\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{J}(\mathbf{x}^{(k)}) + \Delta \mathbf{x}^{(k)} = 0 \quad (2)$$



Здесь  $\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \overline{\frac{\partial x_1}{\partial x_1}} & \overline{\frac{\partial x_2}{\partial x_1}} & \cdots & \overline{\frac{\partial x_n}{\partial x_1}} \\ \overline{\frac{\partial x_1}{\partial x_2}} & \overline{\frac{\partial x_2}{\partial x_2}} & \cdots & \overline{\frac{\partial x_n}{\partial x_2}} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\frac{\partial x_1}{\partial x_n}} & \overline{\frac{\partial x_2}{\partial x_n}} & \cdots & \overline{\frac{\partial x_n}{\partial x_n}} \end{bmatrix}$  - матрица Якоби.

Выражая из (2) вектор приращений  $\Delta \mathbf{x}^{(k)}$  и подставляя его в (1), итерационный процесс нахождения решения можно записать в виде

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

В практических вычислениях в качестве условия окончания итераций обычно используется критерий

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k)}\| \leq \varepsilon$$

## Метод простой итерации

При использовании метода простой итерации система уравнений приводится к эквивалентной системе специального вида

[illegible]

Если выбрано некоторое начальное приближение  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^T$ , последующие приближения в методе простой итерации находятся по формулам

$$\left\{ \begin{array}{l} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \\ \vdots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \end{array} \right.$$

**Теорема.** Пусть вектор-функция  $\varphi(\mathbf{x})$  непрерывна, вместе со своей производной

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

в ограниченной выпуклой замкнутой области  $G$  и

$$\max_{x \in G} \|\varphi'(\mathbf{x})\| \leq q < 1$$

где  $q$  - постоянная. Если  $\mathbf{x}^{(0)} \in G$  и все последовательные приближения

$$\mathbf{x}^{(k+1)} = \varphi(\mathbf{x}^{(k)}), \quad k = 1, 2, \dots$$

содержатся в  $G$ , то процесс итерации сходится к единственному решению уравнения в области  $G$  и справедливы оценки погрешности ( $\forall k \in N$ ):

$$\|\mathbf{x}^{(*)} - \mathbf{x}^{(k+1)}\| \leq \frac{q^{(k+1)}}{1 - q} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|$$

$$\|\mathbf{x}^{(*)} - \mathbf{x}^{(k+1)}\| \leq \frac{q}{1 - q} \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$$

## Реализация

```
1 point NewtonMethod (double x, double y, double eps) {
2     double next_x = x;
3     double next_y = y;
4     do {
5         x = next_x;
6         y = next_y;
7         //std::cout << x << ' ' << y << '\n';
8         double detJ = det(f1dx1(x,y), f1dx2(x,y), f1dx2(x,y), f2dx2(x,y));
9         double detA_x = det(f1(x,y), f1dx2(x,y), f2(x,y), f2dx2(x,y));
10        next_x = x - detA_x / detJ;
11
12        double detA_y = det(f1dx1(x,y), f1(x,y), f2dx1(x,y), f2(x,y));
13        next_y = y - detA_y / detJ;
14    } while(Norma(next_x, next_y, x, y) > eps);
15
16    return point(next_x, next_y);
17 }
```

  

```
1 point SimpleIter (double x, double y, double eps, double q) {
2     double next_x = x;
3     double next_y = y;
4     do {
5         x = next_x;
6         y = next_y;
7         next_x = phi2(x, y);
8         next_y = phi1(x, y);
9     } while (q / (1 - q) * Norma(next_x, next_y, x, y) > eps);
10    return point(next_x, next_y);
11 }
```

## Результаты

```
1 || (base) MacBook-Air-Dima:Program dandachok$ ./a.out
2 || Newton method: (5.92877, 1.25097)
3 || Simple iter method: (5.90245, 1.122)
4 || (base) MacBook-Air-Dima:Program dandachok$
```