МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Вычислительная математика»

Тема: Решение нелинейных уравнений

Студент гр. 0304 _____ Алексеев Р.В.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2021

Вариант 1

Цель работы.

Целью работы является нахождение корня уравнения $f(x)=0$ методами *Ньютона* и *простых итераций* с заданной точностью *Eps*, исследование скорости сходимости и обусловленности метода.

Основные теоретические положения.

Метод Ньютона. В случае, когда известно хорошее начальное приближение решения уравнения $f(x)=0$, эффективным методом повышения точности является метод *Ньютона*. Он состоит в построении итерационной последовательности (1)

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}, \qquad (1)$$

где $f'(x_n)\neq 0$. Последовательность сходится к корню c уравнения $f(x)=0$. По теореме о сходимости метода *Ньютона* c должен быть простым корнем уравнения $f(x)=0$; в отсекающем промежутке этого корня $[a,b]$ $f(a)\cdot f(b)<0$; функция $f(x)$ – дважды непрерывно дифференцируема и $f''(x)\neq 0$.

Для оценки погрешности *n*-го приближения корня предлагается пользоваться неравенством (2)

$$|c-x_n|\le\frac{M_2}{2m_1}|x_n-x_{n-1}|^2, \qquad (2)$$

где M_2 - наибольшее значение модуля второй производной ¿$f''(x)\vee$¿ на отрезке [a,b]; m_1 - наименьшее значение модуля первой производной ¿$f'(x)\vee$¿ на отрезке [a,b]. Таким образом, если $|x_n-x_{n-1}|<\varepsilon$, то $|c-x_n|\le\frac{M_2}{2m_1}\varepsilon^2$. Это означает, что при хорошем начальном приближении корня после каждой итерации число верных десятичных знаков в очередном приближении

удваивается, т.е. процесс сходится очень быстро и имеет место квадратическая сходимость.

Если необходимо найти корень с точностью ε, то итерационный процесс можно прекращать, когда выполняется неравенство (3)

$$|x_n - x_{n-1}| < \varepsilon = \sqrt{\frac{2m_1\varepsilon}{M_2}}. \quad (3)$$

Если на *(n-1)*-м шаге очередное приближение x_{n-1} не удовлетворяет условию окончания процесса, то вычисляются величины $f(x_{n-1}), f'(x_{n-1})$ и следующие приближение корня $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$. При выполнении условия (3) величина x_n принимается за приближенное значение корня *c*, вычисленное с точностью ε.

Постановка задачи. В практической работе предлагается, использовать программы - функции NEWTON (листинг 4) и Round (листинг 1).

Листинг 4 – Функция NEWTON

```
double NEWTON(double X, double Eps, int& N) {
// extern double F1(double);
double Y, Y1, DX, Eps0;
N = 0;
double m1 = 3.29; // наименьшее значение модуля 1-ой производной
double M2 = 16.424; // наибольшее значение модуля 2-ой производной
Eps0 = sqrt(2 * m1 * Eps / M2);
do {Y = F(X);
if (Y == 0.0) {
return X;}
Y1 = F1(X);
if (Y1 == 0.0) {
puts("Производная обратилась в ноль\n");
exit(1);}
DX = Y / Y1;
X -= DX;
N++;
} while (fabs(DX) >= Eps0);
return
X;}
```

Этапы выполнения практической работы:

1. Графически или аналитически отделить корень уравнения (найти отрезки [Left, Right], на которых функция $f(x)$. удовлетворяет условиям теоремы о сходимости метода Ньютона).

2. Проверить функцию $f(x)$на выпуклость вверх или вниз, выбрать начальное приближение корня $x_0 \epsilon [a,b]$,так чтобы$f(x_0)\cdot f''(x_0)>0$.

3. Получить аналитическое выражение функций $f'(x)$ и $f''(x)$. Оценить снизу величину $m_1 = \min\limits_{x\epsilon[a,b]} |f'(x)|$, оценить сверху величину $M_2 = \max\limits_{x\epsilon[a,b]} |f''(x)|$.

4. По заданному Eps сосчитать условие окончания итерационного процесса $Eps2 = \sqrt{\frac{2 m_1 Eps}{M_2}}$.

5. Составить подпрограммы-функции вычисления $f(x)$, $f'(x)$, предусмотрев округление их значений с заданной точностью Delta.

6. Составить головную программу, вычисляющую корень уравнения $f(x)=0$ и содержащую обращение к подпрограммам $f(x)$, $f'(x)$, Round, NEWTON и индикацию результатов.

7. Провести вычисления по программе. Исследовать скорость сходимости метода и чувствительность метода к ошибкам в исходных данных.

Метод простых итераций. Метод *простых итераций* решения уравнения $f(x)=0$ заключается в замене исходного уравнения эквивалентным ему уравнением $x=\varphi(x)$ и построении последовательности $x_{n+1}=\varphi(x_n)$, сходящейся при $n \to \infty$ к точному решению

Корень уравнения $x=\varphi(x)$ является точкой пересечения двух графиков $y=x$ и $y=\varphi(x)$. Сходимость метода зависит от вида функции $\varphi(x)$. В зависимости от величины модуля первой производной ¿$\varphi'(x)\vee$¿ метод может сходиться и расходиться.

Достаточные условия сходимости метода *простых итераций* формулируются следующей теоремой:

Если функция $\varphi(x)$ определена, дифференцируема и принадлежит отрезку $[a,b]$, то существует числоq, такое что ¿$\varphi'(x)\vee\leq q<1$ на $[a,b]$, и последовательность$x_{n+1}=\varphi(x_n)$,сходится к единственному решению на $[a,b]$ уравнения $x=\varphi(x)$ при $\forall\, x_0\in[a,b]$(1)

$$\lim_{n\to\infty} x_n=\lim_{n\to\infty}\varphi(x_n¿)=c, f(c)=0, c\epsilon[a,b].¿ \qquad (1)$$

Если $\varphi'(x)>0$, то $|c-x_n|<\frac{q}{1-q}\vee x_n-x_{n-1}|$, если $\varphi'(x)<0$, то $|c-x_n|<¿ x_n-x_{n-1}|$.

Рассмотрим один шаг итерационного процесса. Исходя из найденного на предыдущем шаге значения x_{n-1}, вычисляется $y=\varphi(x_{n-1})$. Если $|y-x_{n-1}|>\varepsilon$, то полагается $x_n=y$ и выполняется очередная итерация. Если же $|y-x_{n-1}|<\varepsilon$, то вычисления заканчиваются и за приближенное значение корня принимается величина $x_n=y$. Погрешность результата вычислений зависит от знака производной $\varphi'(x)$: при $\varphi'(x)>0$: погрешность определения корня составляет $\frac{q\varepsilon}{1-q}$, а при $\varphi'(x)<0$, погрешность не превышает ε. Здесь q- число, такое, что ¿$\varphi'(x)\vee\leq q<1$ на отрезке [a,b]. Существование числа q является условием сходимости метода в соответствии с отмеченной выше теоремой.

Для применения метода *простых итераций* определяющее значение имеет выбор функции $\varphi(x)$, в уравнении $x=\varphi(x)$, эквивалентном исходному. Функцию $\varphi(x)$ необходимо подбирать так, чтобы ¿$\varphi'(x)\vee\leq q<1$. Это обусловливается тем, что если $\varphi'(x)<0$, на отрезке [a,b], то последовательные приближения $x_n=\varphi(x_{n-1})$. будут колебаться около корня c, если же $\varphi'(x)>0$, то последовательные приближения будут сходиться к корню c монотонно.

Число обусловленности метода *простых итераций* (2)

$$\nu_\Delta=\frac{1}{¿1-\varphi'(x)\vee¿.¿}$$

Постановка задачи. В практической работе предлагается использовать программы функции ITER и PHI (листинг 5).

Листинг 5 - функции ITER и PHI

```
double ITER (double X0, double Eps, int& N, double a, double b) {
if (Eps <= 0.0) {
puts("Неверное задание точности\n");
exit(1); }
double X1 = PHI(X0, a, b);
double X2 = PHI(X1, a, b);
for (N = 2; (X1 - X2) * (X1 - X2) > fabs((2 * X1 - X0 - X2) *
Eps); N++) {
X0 = X1;
X1 = X2;
X2 = PHI(X1, a, b); }
return X2; }
double PHI(double x, double a, double b) {
if (x == 0) {
printf("деление на 0!");
exit(1); }
double min = min_F1(a, b);
double max = max_F1(a, b);
double s = x - 2 / (min + max) * (F(x));
s = Round(s, delta);
return(s); }
```

Необходимо осуществить следующие шаги:

1) Графически или аналитически отделить корень уравнения $f(x)=0$ и получить отрезок [Left, Right].

2) Сосчитать $f'(x)$, найти на отрезке [Left, Right] m – минимальное значение $f'(x)$, M – максимальное значение $f'(x)$.

3) Преобразовать уравнение $f(x)=0$ к виду, удобному для итераций $\varphi(x)=x-\frac{2}{m+M}f(x)$.

3) Выбрать начальное приближение x_0, лежащее на [Left, Right].

4) Составить подпрограммы для вычисления значений $\varphi(x)$,$\varphi'(x)$, предусмотрев округление вычисленных значений с точностью Delta.

5) Составить головную программу, вычисляющую корень уравнения и содержащую обращение к программам $\varphi(x)$,$\varphi'(x)$, PHI, ITER и индикацию результатов.

6) Провести вычисления по программе. Исследовать скорость сходимости и обусловленность метода.

Выполнение работы.

Функция $f(x) = x^2+5.0\sin(x)$

1. Посторим график и локализуем корень (см. рис. 1)

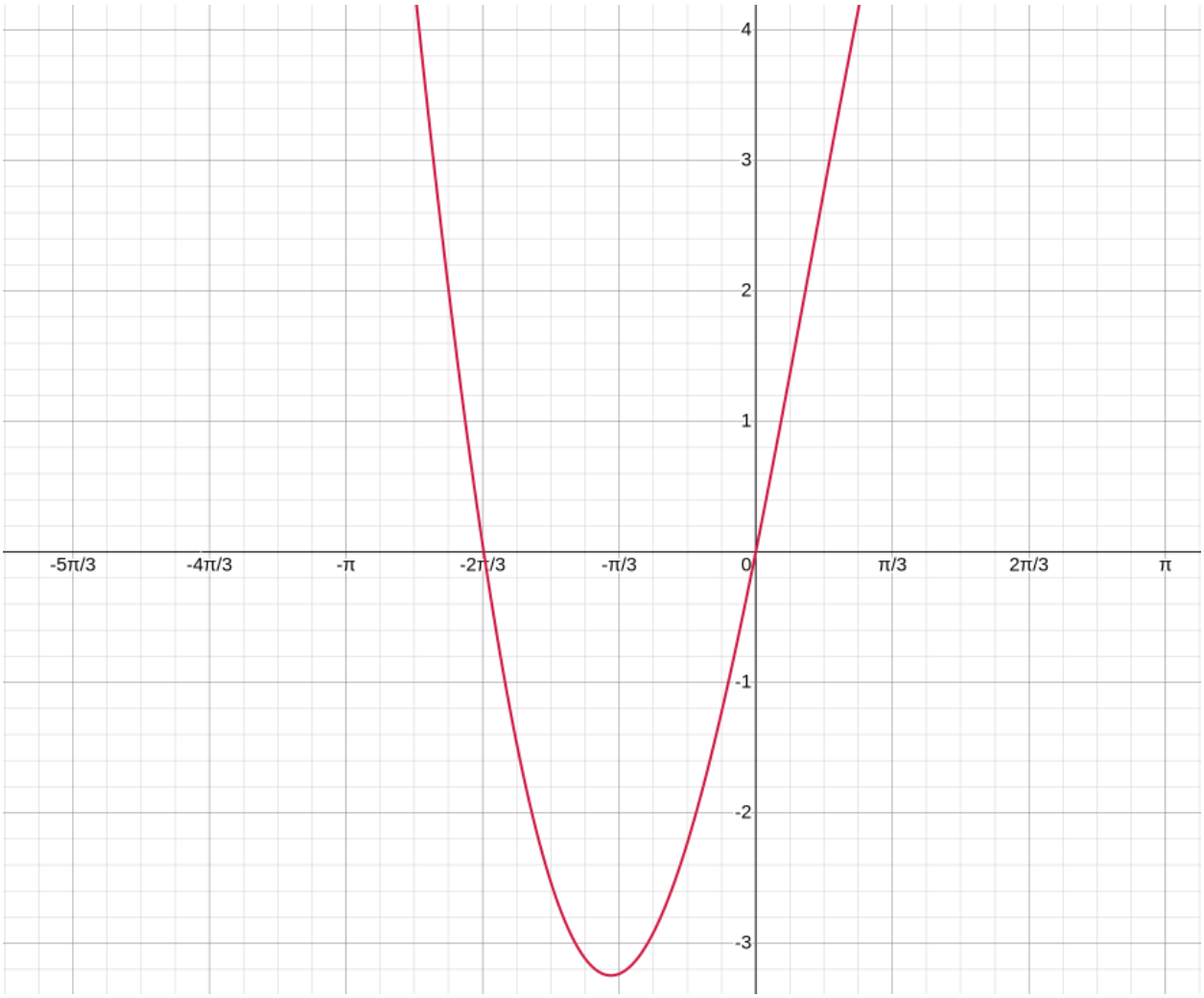

Рисунок 1 — График функции $f(x) = x^2+5.0\sin(x)$

По графику видно, что корни уравнения лежат в промежутке $[\ \frac{-7\pi}{9}\ ;\ \frac{-5\pi}{9}\ ]$ и в точке (0;0), возьмем первый отрезок.

Найдем первую производную функции:

$f'(x) = 2x+5\cos(x)$

График первой производной представлен на рис. 2.

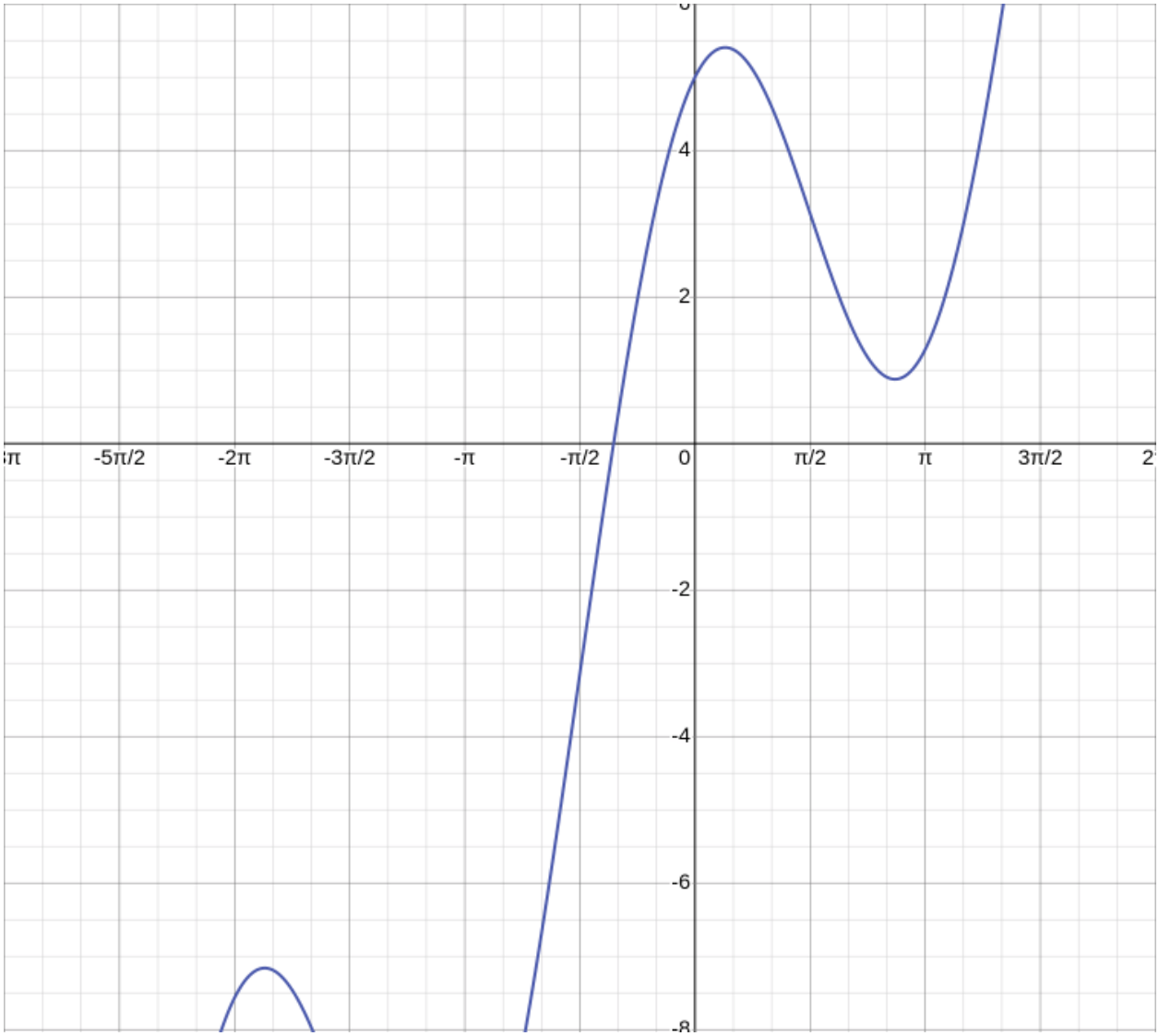

Рисунок 2 — График первой производной функции *f(x)*.

Найдем вторую производную функции:

$f''(x) = 2 - 5\sin(x)$

График второй производной представлен на рис. 3.

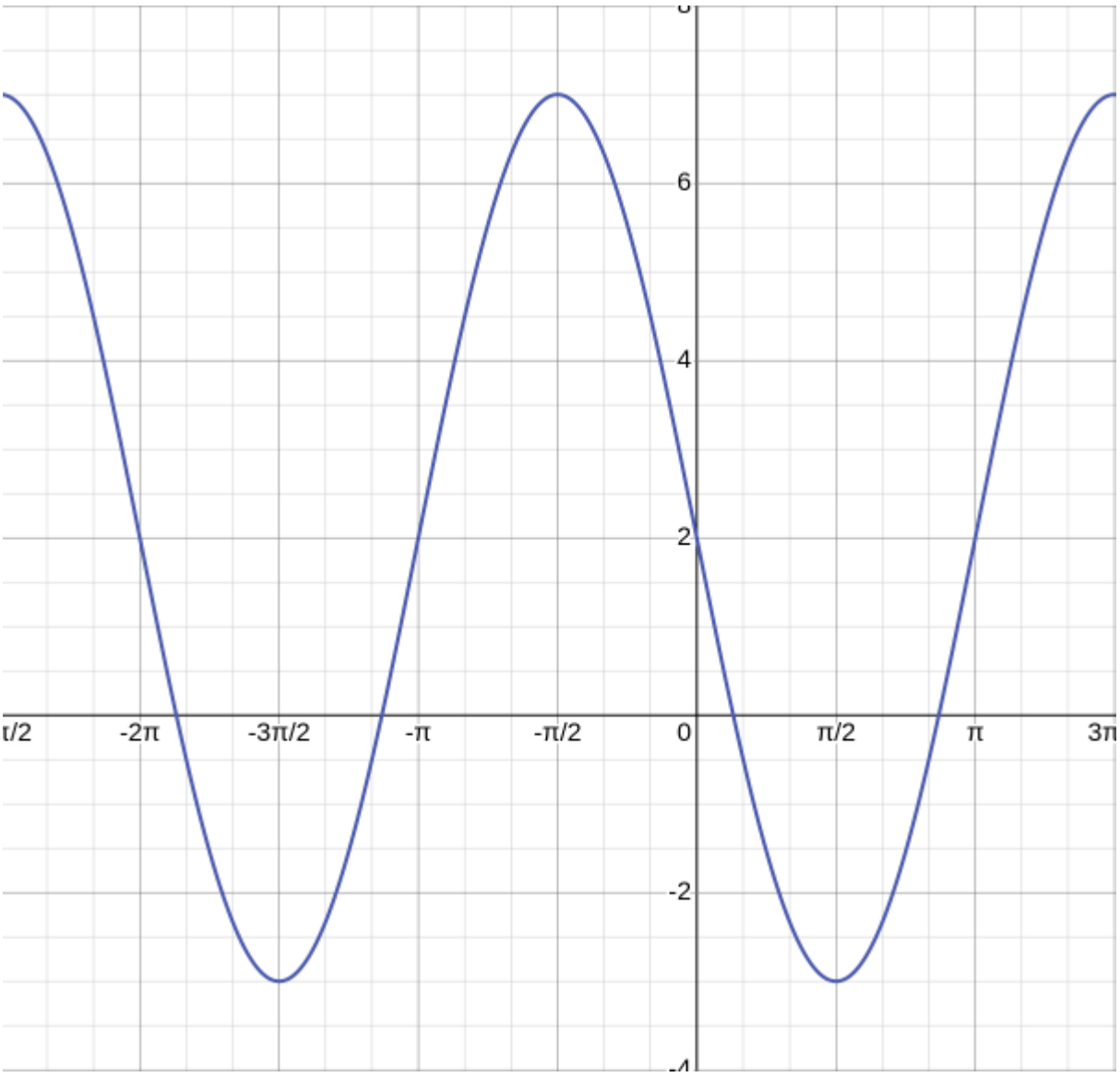

Рисунок 3 — График второй производной функции *f(x)*.

По графикам видно, чтона промежутке $[\ \frac{-7\pi}{9}\ ;\ \frac{-5\pi}{9}\ ]$ первая и вторая производные не изменяют знак, значит условия теоремы о сходимости метода Ньютона выполнены, следовательно промежуток нам подходит.

Метод Ньютона

2. Т.к. вторая производная на промежутке $[\ \frac{-7\pi}{9}\ ;\ \frac{-5\pi}{9}\ ]$ положительна, то функция на этом промежутке выпукла вниз. Выберим начальное

приближение x_0 такое, что $f(x_0)*f''(x_0)>0$. В точке $\frac{-7\pi}{9}$ значения функции и второй производной положительны, а в точке $\frac{-5\pi}{9}$ значение функции отрицательное, второй производной — положительное, значит нам подходит точка $\frac{-7\pi}{9}\approx -2{,}443$, которая может являться точкой входа для метода Ньютона.

3. Подсчитаем минимальное значение модуля первой производной и максимальное значение модуля второй производной: $m_1 = \min\limits_{x\in[a,b]} |f'(x)|$ = 4,359 и $M_2 = \max\limits_{x\in[a,b]} |f''(x)|$ = 6,924.

4. Возьмем delta = 0,000001 и eps, варьирующуюся от 0,000001 до 0,1. Точка входа в методе Ньютона: -2,443. Результаты работы метода Ньютона пресдтавлены на рис. 4.

eps	eps2	x	iter
0.000001	0.001122	-2.085935	3
0.000010	0.003548	-2.085935	3
0.000100	0.011221	-2.085935	3
0.001000	0.035484	-2.085935	3
0.010000	0.112210	-2.086708	2
0.100000	0.354838	-2.127151	1

Рисунок 4 — Разельтаты работы метода Ньютона при постоянном delta.

По рисунку видно, что чем больше eps, тем меньше количество итераций и больше значение условия окончания метода eps2.

5. Возьмем eps = 0,001 и delta от 0,000001 до 0,01. Точка входа - -2,443. Для проверки обусловеленности метода Ньютона коэффицинт обусловеленности рассчитывается по формуле $nu = \frac{1}{|1-phi'(x)|}$, где

phi(x)= $\frac{f(x)}{f'(x)}$. Тогда *phi'(x)*=1- $\frac{(f'(x))^2-f(x)f''(x)}{(f'(x))^2}$,

nu= $|\frac{(f'(x))^2}{(f'(x))^2-f(x)f''(x)}|$. Результаты работы программы представлены на рис. 5.

eps	delta	x	nu	nu_max	Вывод
0.001000	0.000001	-2.085935	1.000000	1000.000000	Хорошо
0.001000	0.000010	-2.085930	0.999996	100.000000	Хорошо
0.001000	0.000100	-2.085900	0.999967	10.000000	Хорошо
0.001000	0.001000	-2.086000	1.000063	1.000000	Плохо
0.001000	0.010000	-2.090000	1.003878	0.100000	Плохо

Рисунок 5 — Результаты работы метода Ньютона при постоянном eps.

По рисунку видно, что с увеличением delta значение обусловленности изменяется слабо, а максимальное число обусловленности уменьшается. При delta = 0,001 задача становится плохо обусловленной.

Метод простых итераций

6. Минимальное и максимальное значения первой производной функции *f(x)* на отрезке [$\frac{-7\pi}{9}$; $\frac{-5\pi}{9}$] - m=-8,717 и M=-4,359 соответственно.

7. Для метода простых итераций используется функция *phi(x)*= $x-\frac{2}{m+M}f(x)$, $\alpha=\frac{2}{m+M}$. Производная этой функции имеет вид *phi'(x)* = $x'-\alpha f'(x)=1-\alpha f'(x)=1-\frac{2'f(x)}{m+M}$. Минимальное и максимальное абсолютные значения производной составляют $|phi(x_{max})|=|1-\frac{2M}{m+M}|=|\frac{M-m}{M+m}|=q$ и $|phi(x_{min})|=|1-\frac{2m}{m+M}|=|\frac{m-M}{M+m}|=q$. Таким образом на промежутке [$\frac{-7\pi}{9}$; $\frac{-5\pi}{9}$] выполняется условие

$|phi'(x)| \le q < 1$, для проведения вычислений методом простых итераций в качестве точки входа возьмем правую границу отрезка локализации.

8. Возьмем delta = 0,000001 и eps от 0,000001 до 0,1. Точка входа в методе простых итераций - -2,443. Результаты работы программы представлены на рис. 6.

eps	x	n
0.000001	-2.085934	4
0.000010	-2.085934	4
0.000100	-2.085950	3
0.001000	-2.085950	3
0.010000	-2.084898	2
0.100000	-2.084898	2

Рисунок 6 — Результаты работы метода простых итераций при постоянном delta.

По рисунку видно, что чем больше значение eps, тем меньше требуется итераций, необходимых для нахождения корня.

9. Возьмем eps = 0,001 и delta от 0,000001 до 0,01. Проверим обусловленность метода простых итераций, коэффициент обусловленности рассчитывается по формуле nu= $\frac{1}{|1-phi'(x)|}$, где *phi(x)*= $x-\frac{2f(x)}{m+M}$, тогда nu= $|\frac{m+M}{2f'(x)}|$. Результаты работы программы представлены на рис. 7.

eps	delta	x	nu	nu_max	Вывод
0.001000	0.000001	-2.085950	0.831250	1000.000000	Хорошо
0.001000	0.000010	-2.085950	0.831250	100.000000	Хорошо
0.001000	0.000100	-2.085900	0.831290	10.000000	Хорошо
0.001000	0.001000	-2.086000	0.831210	1.000000	Хорошо
0.001000	0.010000	-2.090000	0.828042	0.100000	Плохо

Рисунок 7 — Результаты работы метода простых итераций при постоянном delta.

По рисунку видно, что с возрастанием delta уменьшается максимальное число обусловленности, при этом число обусловленности изменяется слабо. При delta>0,01 задача становится плохо обусловленной.

10. Зададим eps=0,000001 и delta=0,000001. Будем сдвигать границы отрезка локализации, а значит и точку входа. Левая граница варьируется от -2,443 до -2,193, правая граница варьируется от -1,745 до -1,995 с шагом 0,05. При изменении отрезка локализации изменяются максимальное и минимальное значения производной, следовательно и коэффициент α , с изменением α будет изменяться и функция $phi'(x)$. Результаты работы программы представлены на рис. 8.

left	right	x	q
-2.443000	-1.745000	-2.085934	0.333410
-2.393000	-1.795000	-2.085934	0.284968
-2.343000	-1.845000	-2.085935	0.236841
-2.293000	-1.895000	-2.085935	0.188974
-2.243000	-1.945000	-2.085935	0.141313
-2.193000	-1.995000	-2.085935	0.093808

Рисунок 8 — Работа метода простых итераций при изменении границ отрезка локализации корня.

По рисунку видно, что при изменении точки входа значение корня изменяется незначительно, а значение q — границы модуля $phi'(x)$, уменьшается, т.к. изменяются максимальное и минимальное значения производной функции *f(x).*

Сравнение методов Ньютона и простых итераций

10. На примере заданной функции сравним методы бисекции, хорд, Ньютона и простых итераций. Возьмем промежуток [$\frac{-7\pi}{9}$; $\frac{-5\pi}{9}$], delta = 0,000001, eps от 0,000001 до 0,1. Результаты работы программы представлены на рис. 9.

Метод	eps	x_0	x	iter
Бисекции	0.000001	-2.443000	-2.085933	19
	0.000010	-2.443000	-2.085937	16
	0.000100	-2.443000	-2.085991	12
	0.001000	-2.443000	-2.087184	9
	0.010000	-2.443000	-2.083094	6
	0.100000	-2.443000	-1.919500	2
Хорд	0.000001	-2.443000	-2.085935	8
	0.000010	-2.443000	-2.085934	7
	0.000100	-2.443000	-2.085931	6
	0.001000	-2.443000	-2.085912	5
	0.010000	-2.443000	-2.084773	3
	0.100000	-2.443000	-2.077634	2
Ньютона	0.000001	-2.443000	-2.085935	3
	0.000010	-2.443000	-2.085935	3
	0.000100	-2.443000	-2.085935	3
	0.001000	-2.443000	-2.085935	3
	0.010000	-2.443000	-2.086641	2
	0.100000	-2.443000	-2.125272	1
Простых итераций	0.000001	-2.443000	-2.085934	4
	0.000010	-2.443000	-2.085934	4
	0.000100	-2.443000	-2.085950	3
	0.001000	-2.443000	-2.085950	3
	0.010000	-2.443000	-2.084898	2
	0.100000	-2.443000	-2.084898	2

Рисунок 9 — Сравнение работы разных методов.

По рисунку видно, чтометод бисекции является наиболее медленным. Следующим по скорости является метод хорд. Методы Ньютона немного быстрее метода простых итераций, оба этих метода быстрее методов хорд и бисекции, но они не всегда сходятся.

Разработанный программный код см. в приложении А.

Выводы.

Были исследованы методы решения нелинейных уравнений, а именно методы Ньютона и простых итераций. Эти метода работают быстрее методов бисекции и хорд, но они не всегда сходятся.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ НА ЯЗЫКЕ C++

```
#include <cstdio>
#include <cmath>

using namespace std;

double F(double x) {
  return x*x + 5.0*sin(x);
}

double F1(double x) {
  return 2*x + 5*cos(x);
}

double F2(double x) {
  return 2 - 5*sin(x);
}

double nu_newton(double x){
  return pow(F1(x), 2)/(pow(F1(x), 2) - F(x)*F2(x));
}

double nu_iter(double x){
  return fabs(-11.03108758/(2*F1(x)));
}

double NEWTON(double X, double Eps, int& N) {
  double Y, Y1, DX, Eps0;
  N = 0;
  double m1 = 4.359; // наименьшее значение модуля 1-ой
производной
  double M2 = 6.924; // наибольшее значение модуля 2-ой
производной
  Eps0 = sqrt(2 * m1 * Eps / M2);
  do {
      Y = F(X);
      if (Y == 0.0) {
          return X;
      }
      Y1 = F1(X);
      if (Y1 == 0.0) {
          puts("Производная обратилась в ноль\n");
          exit(1);
      }
      DX = Y / Y1;
      X -= DX;
      N++;
  } while (fabs(DX) >= Eps0);
  return X;
}

double PHI(double x, double a, double b) {
```

```
  if (x == 0) {
      printf("деление на 0!");
      exit(1);
  }
  double min = fmin(F1(a), F1(b));
  double max = fmax(F1(a), F1(b));
  double s = x - 2 / (min + max) * (F(x));
  return(s);
}

double ITER (double X0, double Eps, int& N, double a, double b) {
  if (Eps <= 0.0) {
      puts("Неверное задание точности\n");
      exit(1);
  }
  double X1 = PHI(X0, a, b);
  double X2 = PHI(X1, a, b);
  for (N = 2; pow((X1 - X2), 2) > fabs((2 * X1 - X0 - X2) * Eps);
N++) {
      X0 = X1;
      X1 = X2;
      X2 = PHI(X1, a, b);
  }
  return X2;
}

double BISECT(double Left, double Right, double Eps, int& N) {
  double E = fabs(Eps)*2.0;
  double FLeft = F(Left);
  double FRight = F(Right);
  double X = (Left + Right) / 2.0;
  double Y;
  if (FLeft * FRight > 0.0) {
      puts("Неверное задание интервала\n");
      exit(1);
  }
  if (Eps <= 0.0) {
      puts("Неверное задание точности\n");
      exit(1);
  }
  N = 0;
  if (FLeft == 0.0)
      return Left;
  if (FRight == 0.0)
  return Right;
  while ((Right - Left) >= E) {
      X = 0.5*(Right + Left);
      Y = F(X);
      if (Y == 0.0)
          return X;
      if (Y*FLeft < 0.0)
          Right = X;
      else {
          Left = X;
          FLeft = Y;
```

```
            }
            N++;
        }
        return X;
    }

    double HORDA(double Left, double Right, double Eps, int &N) {
        double FLeft = F(Left);
        double FRight = F(Right);
        double X, Y;
        if (FLeft * FRight > 0.0) {
            puts("Неверное задание интервала\n");
            exit(1);
        }
        if (Eps <= 0.0) {
            puts("Неверное задание точности\n");
            exit(1);
        }
        N = 0;
        if (FLeft == 0.0) { return Left; }
        if (FRight == 0.0) { return Right; }
        do {
            X = Left - (Right - Left) * FLeft / (FRight - FLeft);
            Y = F(X);
            if (Y == 0.0) { return X; }
            if (Y * FLeft < 0.0) { Right = X; FRight = Y; }
            else { Left = X; FLeft = Y; }
            N++;
        } while (fabs(Y) >= Eps);
        return X;
    }

    double Round(double X, double Delta) {
        if (Delta <= 1E-9) {
            puts("Неверно задана точность округления\n");
            exit(1);
        }
        if (X>0.0)
            return (Delta*(long((X / Delta) + 0.5)));
        else
            return (Delta*(long((X / Delta) - 0.5)));
    }

    int main(){

        double eps = 0.000001;
        double delta = 0.000001;
        int n = 0;


puts("-----------------------------------------------------------------
-------------------------------");
        printf("%6s\t%14s\t%13s\t%12s\n", "eps", "eps2", "x", "iter");
```

```
puts("-------------------------------------------------------------------
------------------------------");
       for(eps = 0.000001; eps <= 0.1; eps *= 10){
           double x = Round(NEWTON(-2.443, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%2d\t\n",                          eps,
sqrt(2*4.359*eps/6.924), x, n);
       }

puts("-------------------------------------------------------------------
------------------------------");

       eps = 0.001;
       n = 0;


puts("-------------------------------------------------------------------
------------------------------");
       printf("%6s\t%14s\t%13s\t%12s\t%16s\t%10s\n",  "eps",  "delta",
"x", "nu", "nu_max", "Вывод");

puts("-------------------------------------------------------------------
------------------------------");
       for(delta = 0.000001; delta <= 0.01; delta *= 10){
           double x = Round(NEWTON(-2.443, eps, n), delta);
           printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6s\n",  eps,  delta,  x,
nu_newton(x),  eps/delta,  nu_newton(x)  <  eps/delta  ?  "Хорошо"  :
"Плохо");
       }

puts("-------------------------------------------------------------------
------------------------------");

       delta = 0.000001;
       n = 0;


puts("-------------------------------------------------------------------
------------------------------");
       printf("%6s\t%14s\t%9s\n", "eps", "x", "n");

puts("-------------------------------------------------------------------
------------------------------");
       for(eps = 0.000001; eps <= 0.1; eps *= 10){
           double  x  =  Round(ITER(-2.443,  eps,  n,  -2.443,  -1.745),
delta);
           printf("%6f\t%6f\t%d\n", eps, x, n);
       }

puts("-------------------------------------------------------------------
------------------------------");

       eps = 0.001;
       n = 0;
```

```
puts("-----------------------------------------------------------------
-------------------------------");
       printf("%6s\t%14s\t%14s\t%13s\t%16s\t%10s\n",  "eps",  "delta",
"x", "nu", "nu_max", "Вывод");

puts("-----------------------------------------------------------------
-------------------------------");
       for(delta = 0.000001; delta <= 0.01; delta *= 10){
           double  x  =  Round(ITER(-2.443,  eps,  n,  -2.443,  -1.745),
delta);
           printf("%6f\t%6f\t%6f\t%6f\t%6f\t%s\n",   eps,   delta,   x,
nu_iter(x), eps/delta, nu_iter(x) < eps/delta ? "Хорошо" : "Плохо");
       }

puts("-----------------------------------------------------------------
-------------------------------");

       eps = 0.000001;
       delta = 0.000001;
       n = 0;


puts("-----------------------------------------------------------------
-------------------------------");
       printf("%6s\t%14s\t%14s\t%13s\n", "left", "right", "x", "q");

puts("-----------------------------------------------------------------
-------------------------------");
       for(double left = -2.443, right = -1.745; left <= -2.143, right
>= -2.045; left += 0.05, right -= 0.05){
           double x = Round(ITER(-2.443, eps, n, left, right), delta);
           printf("%6f\t%6f\t%6f\t%6f\n",      left,      right,      x,
fabs((fmin(F1(left),                          F1(right))-fmax(F1(left),
F1(right)))/(fmin(F1(left), F1(right))+fmax(F1(left), F1(right)))));
       }

puts("-----------------------------------------------------------------
-------------------------------");

       delta = 0.000001;
       n = 0;
       int i = 1;


puts("-----------------------------------------------------------------
-------------------------------");
       printf("%6s\t%14s\t%14s\t%13s\t%11s\n",  "Метод",  "eps",  "x_0",
"x", "iter");

puts("-----------------------------------------------------------------
-------------------------------");
       for(eps = 0.000001, i = 1; eps <= 0.1; eps *= 10, i++){
           double x = Round(BISECT(-2.443, -1.745, eps, n), delta);
           if(i == 1)
```

```
            printf("Бисекции\t");
        else
            printf("\t\t");
        printf("%6f\t%6f\t%6f\t%2d\n", eps, -2.443, x, n);
    }

puts("-----------------------------------------------------------------
-------------------------------");

    delta = 0.000001;
    n = 0;
    i = 1;

    for(eps = 0.000001, i = 1; eps <= 0.1; eps *= 10, i++){
        double x = Round(HORDA(-2.443, -1.745, eps, n), delta);
        if(i == 1)
            printf("Хорд\t\t");
        else
            printf("\t\t");
        printf("%6f\t%6f\t%6f\t%2d\n", eps, -2.443, x, n);
    }

puts("-----------------------------------------------------------------
-------------------------------");

    delta = 0.000001;
    n = 0;
    i = 1;

    for(eps = 0.000001, i = 1; eps <= 0.1; eps *= 10, i++){
        double x = Round(NEWTON(-2.433, eps, n), delta);
        if(i == 1)
            printf("Ньютона\t\t");
        else
            printf("\t\t");
        printf("%6f\t%6f\t%6f\t%2d\n", eps, -2.443, x, n);
    }

puts("-----------------------------------------------------------------
-------------------------------");

    delta = 0.000001;
    n = 0;
    i = 1;

    for(eps = 0.000001, i = 1; eps <= 0.1; eps *= 10, i++){
        double x = Round(BISECT(-2.443, -1.745, eps, n), delta);
        if(i == 1 && i != 2)
            printf("Простых");
        if(i != 1 && i == 2)
            printf("итераций\t");
        else
            printf("\t\t");
        printf("%6f\t%6f\t%6f\t%2d\n", eps, -2.443, x, n);
    }
```

```
puts("-------------------------------------------------------------------
--------------------------------");

       return 0;
     }
```