**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №5**

**по дисциплине «Вычислительная математика»**

Тема: Метод Ньютона

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Федоров И.А. |
| Преподаватель |  | Сучков А.И. |

Санкт-Петербург

2019

**Цель работы.**

Сформировать практические навыки нахождения корней алгебраических и трансцендентных уравнений методом Ньютона.

**Основные теоретические положения.**

В случае, когда известно хорошее начальное приближение решения уравнения , эффективным методом повышения точности является метод Ньютона (касательных). Он состоит в построении итерационной последовательности  сходящейся к корню уравнения  Достаточные условия сходимости метода формулируются следующей теоремой: Пусть  определена и дважды дифференцируема на , причём  а производные сохраняют знак на отрезке . Тогда, исходя из начального приближения  удовлетворяющего неравенству можно построить последовательность



сходящуюся к единственному на  решению *ξ* уравнения 

Метод Ньютона допускает простую геометрическую интерпретацию, представленную на рис. 1. Если через точку с координатами  провести касательную, то абсцисса точки пересечения этой касательной с осью OX будет очередным приближением  корня уравнения 

Для оценки погрешности n-го приближения корня предлагается пользоваться неравенством:



где - наибольшее значение модуля второй производной  на отрезке - наименьшее значение модуля первой производной  на

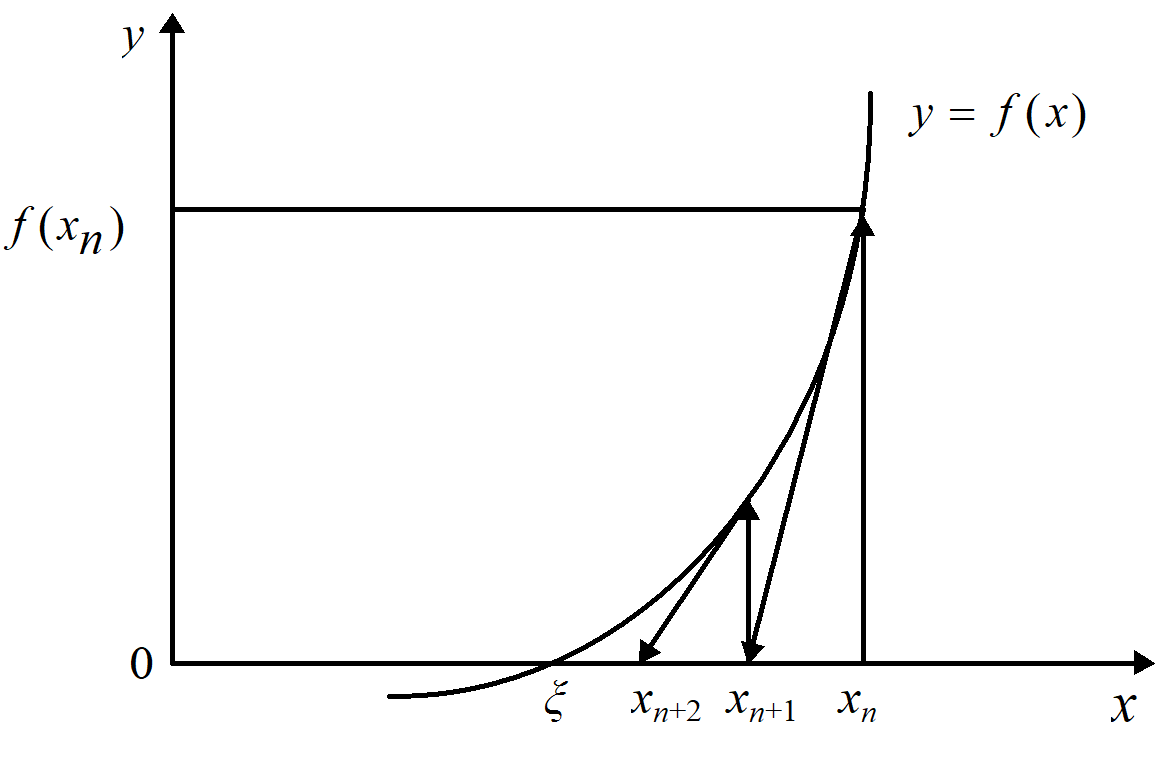


Рисунок 1 – Геометрическая интерпретация метода Ньютона

 - наименьшее значение модуля первой производной  на отрезке  Таким образом, если  то  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 (имеет место квадратическая сходимость). Из указанного следует, что при необходимости нахождения корня с точностью *ε* итерационный процесс можно прекращать, когда



Рассмотрим один шаг итераций. Если на (n−1)-м шаге очередное приближение  не удовлетворяет условию окончания процесса, то вычисляются величины  и следующие приближения корня . При выполнении условия остановки, описанного выше, величина принимается за приближенное значение корня  вычисленное с точность 

**Постановка задачи.**

Используя программы-функции NEWTON и Round из файла methods.cpp (файл заголовков methods.h), найти корень уравнения  заданной точностью *eps* методом Ньютона, исследовать скорость сходимости и обусловленность метода. Порядок выполнения работы следующий:

1. Графически отделить корень уравнения .
2. Выбрать приближение корня  так, чтобы 
3. Оценить снизу величину  оценить сверху величину 
4. По заданному  выбрать значение  для условия итерационного процесса 
5. Составить подпрограммы-функции вычисления  предусмотрев округление их значений с заданной точностью *delta*.
6. Составить головную программу, вычисляющую корень уравнения  и содержащую обращение к подпрограммам F, F1, BISECT, NEWTON и индикацию результатов.
7. Провести вычисления по программе. Исследовать скорость сходимости метода и его чувствительность к ошибкам в исходных данных, сравнить скорости сходимости методов Ньютона, бисекции и хорд.

**Выполнение работы.**

Графически отделим корень уравнения, т.е. найдем отрезок , на котором функция удовлетворяет условиям сходимости метода Ньютона. График функции  представлен на рис. 2. Первая и вторая производные данной функции вычисляются по формулам (1) и (2) соответственно:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

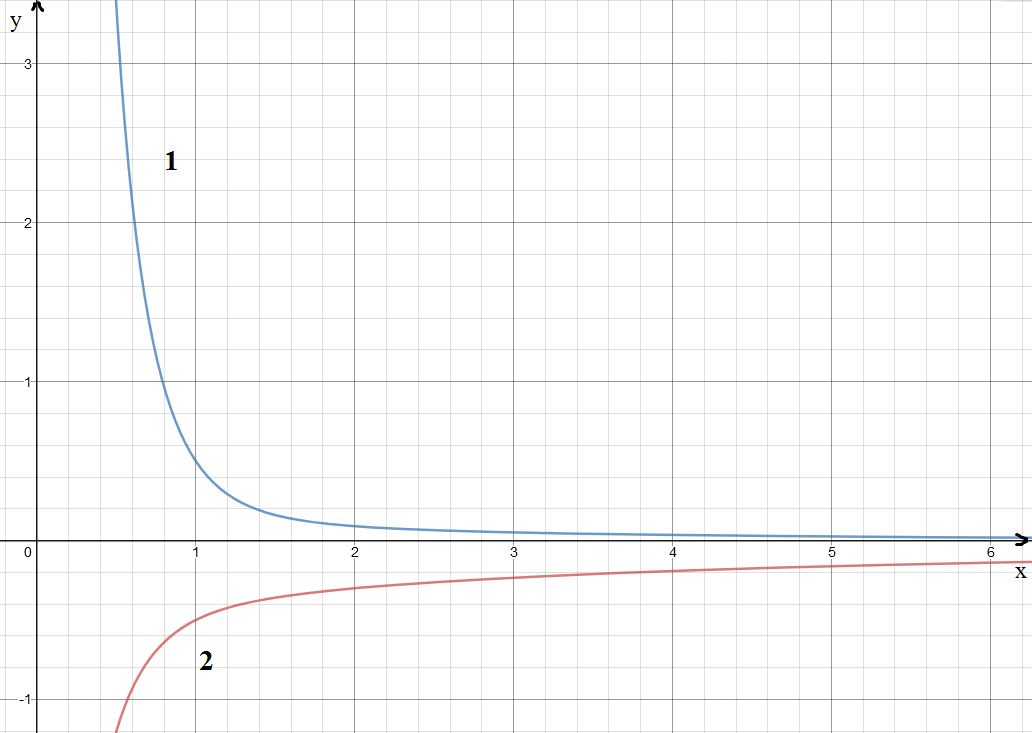


Рисунок 2 – График функции 

Из рисунка возьмем отрезок . Видно, что так как график на это отрезке является выпуклым и убывающим, то на выбранном отрезке  и  то есть непрерывны и сохраняют определенный знак при  Следовательно данный отрезок удовлетворяет условиям сходимости метода.

Выберем начальное приближение корня  Так как на выбранном отрезке  вторая производная  при  то в качестве начального приближения выберем  так как в таком случае будет выполнятся условие хорошего начального приближения 

Проведем оценку величины  снизу и оценим снизу величину  Графики функций первой и второй производных представлены на рис. 3. Из рисунка видно, что т.к. вторая производная на промежутке  убывает, то  Аналогично, т.к. первая производная возрастает и отрицательная, то получим  В итоге имеем что  и .



Подпрограмма вычисления функции, в которой предусматривается округление вычисленных значений функции  с заданной точностью *delta*,  вводимой с клавиатуры, с использованием программы-функции Round, продемонстрирована в приложении А.

Головная функция, вычисляющая корень уравнения  с заданной точностью *eps* и содержащая обращение к подпрограммам вычисления функции F, программам-функциям  HORDA и   Round   и выводящая результаты, продемонстрирована в приложении Б.

Были проведены вычисления функции по программе, с варьирующимися от 0.1 до 0.000001 значениями *eps* (точность вычисления корня). График зависимости числа итераций от параметра *eps* представлен на рис. 3*.* По полученному графику видно, что чем более высокая точность *eps* выходных данных необходима, тем большее число итераций нужно совершить.

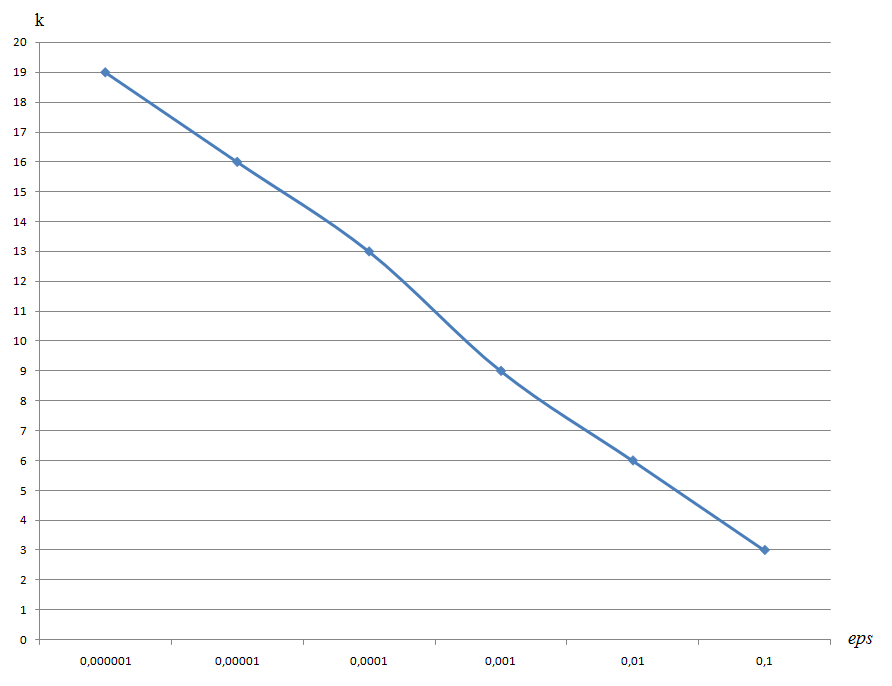


Рисунок 2 – График зависимости итераций от *eps* для метода бисекций

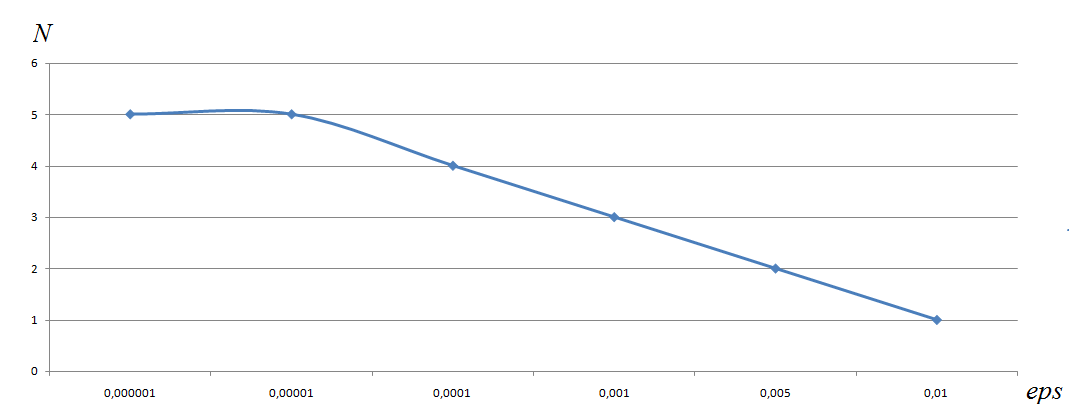


Рисунок 3 – График зависимости итераций от *eps* для метода хорд

Если сравнить полученный график с графиком зависимости для метода бисекции (который был использован в прошлой практической работе), то можно убиться, что графики имеют схожий вид, но для метода хорд нужно меньшее число итераций для вычисления корня с заданной точностью, чем для метода бисекции. Это можно объяснить тем, что в методе хорд отрезок локализации делятся не пополам, а в отношении  .

Были проведены исследования для заданного значения чувствительности метода к ошибка *delta* в исходных данных, которые моделируются с помощью программы Round. Результаты вычислений показаны в табл. 1. По результатам видно, что с ростом ошибок в исходных данных, уменьшается точность результата нахождения корня.

Таблица 1 – Значения корня при разных *delta*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Значение *delta* | Значение корня | Число итераций | Значение | Значение *delta* |
| 0.1 | 3.666667 | 1 | 0.025919 | 0.493865 |
| 0.05 | 3.750000 | 1 | 0.057414 | 0.246933 |
| 0.01 | 3.714286 | 1 | 0.021701 | 0.049387 |
| 0.005 | 3.691244 | 2 | 0.001342 | 0.024693 |
| 0.001 | 3.691125 | 3 | 0.001461 | 0.004939 |
| 0.0005 | 3.692503 | 3 | 0.000083 | 0.002469 |
| 0.0001 | 3.692379 | 4 | 0.000207 | 0.000494 |
| 0.00005 | 3.692676 | 4 | 0.000091 | 0.000247 |
| 0.00001 | 3.692577 | 4 | 0,000009 |  |
| 0.000005 | 3.692579 | 5 | 0,000007 |  |
| 0.000001 | 3.692587 | 5 | 0,000001 |  |

Были проведены вычисление по программе для заданного значения ошибки  c различными значениями точности *eps*, варьирующимися от 0.1 до 0.000001. Результаты вычислений показаны в таблице 2. По результатам видно, что чем большая точность вычисления задана, тем большее число итераций необходимо совершить программе. Это связано с тем, что работа алгоритма продолжается до тех пор, пока значение , где  - последнее вычисленное приближение корня, не станет по модулю меньше заданной точности *eps.* Теоретически метод хорд сходится со скоростью геометрической прогрессии (имеет линейную скорость сходимости). Это можно наблюдать по полученным результатам. Для одношаговых алгоритмов справедлива оценка , где  В данном случае *c* не превышает 0.2.

Таблица 2 – Значения корня при разных *eps*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Значение eps | Номер итерации | Значение | Значение | Значение | Значение |  |
| 0.100000 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 0.010000 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 0.001000 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 0.000500 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 0.000100 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 3 | 3.693987 | -0.000284 | 3.692680 | 0.001307 | 0.067308 |
| 0.000050 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 3 | 3.693987 | -0.000284 | 3.692680 | 0.001307 | 0.067308 |
| 0.000010 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 3 | 3.693987 | -0.000284 | 3.692680 | 0.001307 | 0.067308 |
| 4 | 3.692680 | -0.000019 | 3.692592 | 0.000088 | 0.067006 |
| 0.000005 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 3 | 3.693987 | -0.000284 | 3.692680 | 0.001307 | 0.067308 |
| 4 | 3.692680 | -0.000019 | 3.692592 | 0.000088 | 0.067006 |
| 0.000001 | 1 | 3.000000 | 0.150431 | 3.713257 | 0.713257 | 0.029847 |
| 2 | 3.713257 | -0.004177 | 3.693987 | 0.019271 | 0.067790 |
| 3 | 3.693987 | -0.000284 | 3.692680 | 0.001307 | 0.067308 |
| 4 | 3.692680 | -0.000019 | 3.692592 | 0.000088 | 0.067006 |
| 5 | 3.692592 | -0.000001 | 3.692587 | 0.000006 | 0.208861 |

**Выводы.**

В ходе выполнения практической работы графически отделен корень уравнения  (то есть был найден отрезок локализации ). Была рассмотрена программа нахождения приблизительного значения корня уравнения  для нелинейной функции  методом хорд. Были написаны головная функция, вычисляющая корень с заданной точностью, и функция вычисления значения заданной функции. Был построен график зависимости числа итераций от *eps.* На основе полученного графика был сделан вывод, что при увеличении заданной точности вычисления возрастает и число итераций. При сравнении метод хорд оказался несколько быстрее метода бисекции при тех же условиях (меньше число итераций при выполнении). Были вычислены результаты работы программы с фиксированным *eps* для различных значений *delta* (используя подпрограмму для округления Round). После анализа результатов сделан вывод, что при возрастании погрешности входных данных падает точность вычисления корня. Были проведены вычисление по программе для заданного значения ошибки *delta* и варьирующихся значений *eps.* Экспериментально было проверено, что метод хорд сходится со скоростью геометрической прогрессии (имеет линейную скорость).

Приложение A

Код Подпрограммы ВЫЧИСЛЕНИЯ ФУНКЦИИ

double F(double x) {

if(x < 0){

cout << "Отрицательный x"; exit(1);

}

extern double delta;

double s;

long S;

s = atan(x) - log(x);

if (s / delta < 0) {

S = s / delta - 0.5;

} else {

S = s / delta + 0.5;

}

s = S \* delta;

s = Round(s, delta);

return s;

}

Приложение Б

Код ГОЛОВНОЙ ФУНКЦИИ

int main() {

int k;

long int s;

float a1, b1, c1, d1, eps1, delta1;

double a, b, eps, x;

printf("Введите eps: ");

scanf("%f", &eps1);

eps = eps1;

printf("Введите a: ");

scanf("%f", &a1);

a = a1;

printf("Введите b: ");

scanf("%f", &b1);

b = b1;

printf("Введите delta: ");

scanf("%f", &delta1);

delta = delta1;

x = HORDA(a, b, eps, k);

printf("x = %f k = %d\n", x, k);

return 0;

}