|  |
| --- |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ |
| Факультет прикладной математики и информатики |
| Кафедра информационных систем управления |
| Лукьянович Александр Сергеевич  10 группа, 2 курс |
| Лабораторная работа |
|  |
|  |
| **Методы решения НАУ**    **Преподаватель**  *Полевиков Виктор Кузьмич* |
|  |
| Дата сдачи:  06.03.2017 |

Оглавление

[Постановка задачи 3](#_Toc476523267)

[Метод дихотомии 3](#_Toc476523268)

[Метод простых итераций 4](#_Toc476523269)

[Метод Ньютона 4](#_Toc476523270)

[Краткая теория 5](#_Toc476523271)

[Метод дихотомии 5](#_Toc476523272)

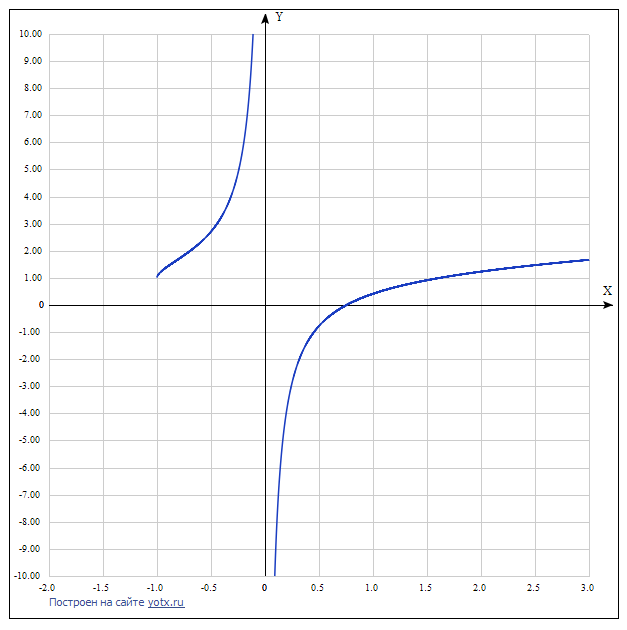
[Метод простых итераций 5](#_Toc476523273)

[Метод Ньютона 6](#_Toc476523274)

[Листинг программы 9](#_Toc476523275)

[Результаты 11](#_Toc476523276)

# Постановка задачи

Рассматриваем функцию . Построим её график:

Также находим первую и вторую производные:

## Метод дихотомии

1. Построить стандартную программу для решения уравнения с заданной точностью .
2. Решить уравнение с помощью этой программы и получить решение. Останавливать процесс при выполнении неравенства .

## Метод простых итераций

1. Построить стандартную программу для решения уравнения с заданной точностью .
2. Решить уравнение с помощью этой программы и получить решение. Останавливать процесс при выполнении неравенства .
3. Получить априорную оценку количества итераций.

## Метод Ньютона

1. Построить стандартную программу для решения уравнения с заданной точностью .
2. Решить уравнение с помощью этой программы и получить решение. Останавливать процесс при выполнении неравенства .

# Краткая теория

## Метод дихотомии

Из графика определяем промежуток [a;b] = [0.5; 3]. Функция на этом промежутке непрерывна и имеет решение. Определяем:

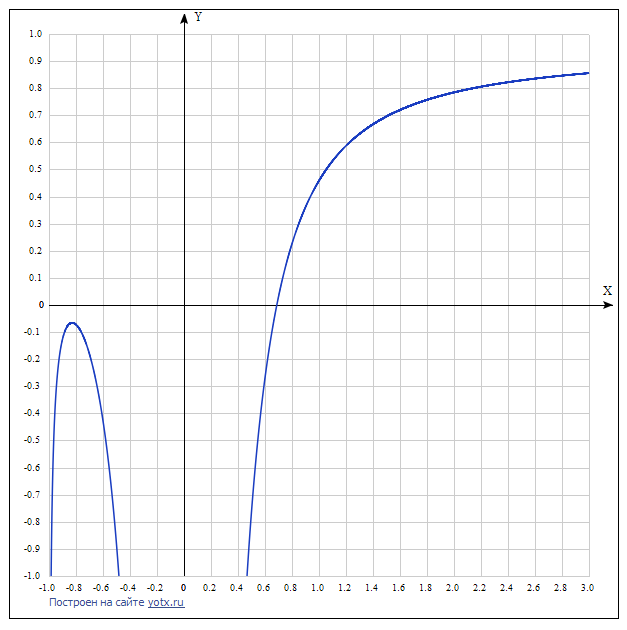
То есть, делим промежуток пополам, определяем с какой стороны от центра исходного отрезка расположен корень и повторяем дробление. При выполнении неравенства останавливаем процесс.

## Метод простых итераций

Задачу нахождения корней можно формулировать как задачу нахождения неподвижной точки .

Строим итерационную последовательность

Наша последовательность действительно сходится к решению, так как соблюдаются условия теоремы о сходимости данного метода:

1. выбирается так, чтобы была определена и непрерывно дифференцируема в .
2. Окрестность [a;b] является областью отделённого корня, потому исходя из графика возьмём [a;b] = [0.65; 0.85], то есть, . А начальное приближение выбираем .
3. . Наша функция должна иметь вид . Подбирая различные константы определяем, что может быть выбрано . То есть, . Построим для наглядности график при заданном :
4. . Действительно, .

Априорную оценку выполняем по формуле

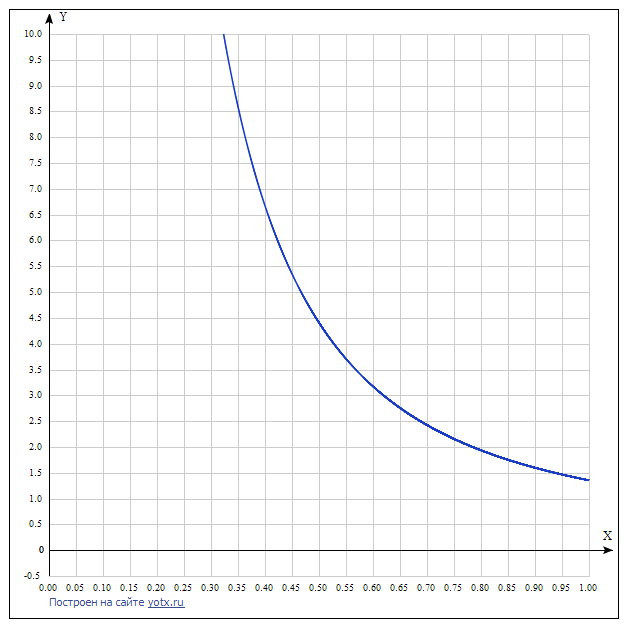
## Метод Ньютона

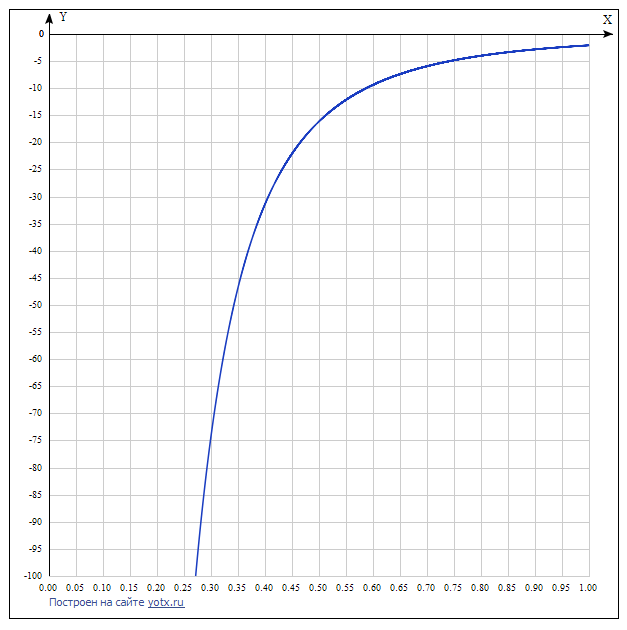
Метод Ньютона представляет собой частный случай метода простых итераций.

Строим итерационную последовательность:

выбираем так, чтобы . То есть, [a;b] = [0.1; 0.77], .

Наша последовательность сходится к решению, так как соблюдены условия теоремы о сходимости данного метода:

1. [a;b] – область отделённого корня
2. определена и дважды дифференцируема на отрезке [a;b]. Причём, сохраняет знак, то есть, . Для наглядности построим графики первой и второй производных: 
3. . Благодаря графикам очень легко заметить, что данное условие выполняется.

При выполнении неравенства останавливаем процесс.

# Листинг программы

Программа выполнена на языке программирования C++, в среде Microsoft Visual Studio Community 2015.

#include "resorces.h"

int main(int argc, char argv[])

{

setlocale(LC\_ALL, ".1251");

const double eps = pow(10, -7);

try

{

ofstream fout("output.txt", ios\_base::trunc);

if (!fout.good())

throw exception("проблема с файлом.");

fout << fixed << setprecision(15);

fout << "х = " << dichotomy(0.5, 3, eps) << endl;

fout << "х = " << mpi(0.8, eps) << endl;

fout << "x = " << newton(0.6, eps) << endl;

fout.close();

}

catch (exception& e) { cerr << e.what() << endl; }

return 0;

}

double funcF(double x) { return sqrt(x + 1) - 1.0 / x; }

double funcF\_derivative(double x) { return 0.5 / sqrt(x + 1) + 1.0 / (x\*x); }

double funcFi(double x) { return x - 0.4 \* funcF(x); }

double dichotomy(const double a, const double b, const double eps)

{

double xLeft = a, xRight = b;

while (true)

{

double x = (xLeft + xRight) / 2;

if (funcF(xLeft)\*funcF(x) < 0)

xRight = x;

else

xLeft = x;

if (fabs(funcF(x)) <= eps)

return x;

}

}

double mpi(const double xBeg, const double eps)

{

double x = funcFi(xBeg);

while (true)

{

double xNext = funcFi(x);

if (fabs(xNext - x) <= eps)

return xNext;

x = xNext;

}

}

double newton(const double xBeg, const double eps)

{

double x = xBeg;

while(true)

{

double xNext = x - funcF(x) / funcF\_derivative(x);

if (fabs(xNext - x) <= eps)

return xNext;

x = xNext;

}

}

# Результаты

Применив методы решения НАУ, мы получили следующие решения:

* метод дихотомии:

х = 0.754877656698227

* метод простых итераций:

х = 0.754877679663058

* метод Ньютона:

x = 0.754877666246693