СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc132402326)

[Практическая часть 3](#_Toc132402327)

[ЗАКЛЮЧЕНИЕ 15](#_Toc132402328)

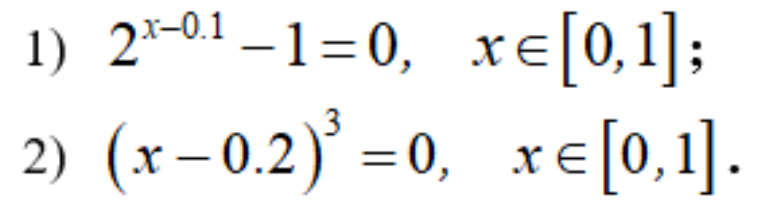
# ВВЕДЕНИЕ

**Цель работы**: изучение методов решения нелинейного уравнения f(x) = 0; сравнение точности и скорости их работы.

*Содержание работы*

1. Реализовать методы Ньютона, секущих, половинного деления в виде программ.

2. Отладить алгоритмы на тестовых примерах, решив уравнения (для всех вариантов):



3. С помощью реализованных методов найти больший нуль в следующей функции (вариант 5):



# Практическая часть

**Реализация метода Ньютона**

%Первая функция (пример)

%f=@(x)2.^(x-0.1)-1;

%Вторая функция (пример)

f=@(x)(x-0.2).^3;

%Точное значение для первой

%xt=0.1;

%Точное значение для второй

xt=0.2;

%Точность

eps=0.0001;

%Начальное приближение

xn=2;

%Настройка графика

ff=f;

xx=0:0.1:2;

plot(xx,f(xx),'r');

hold on;

%Вычисление производной

symF = sym(f);

df = matlabFunction(diff(symF));

x0=xn;

x1=xn-f(xn)/df(xn);

%счетчик итераций

m=0;

tic;

while abs(x1-x0)>eps

m=m+1;

x0 = x1;

x1=x0-f(x0)/df(x0);

xgr=[x1,x0];

ygr=[f(0),f(x0)];

plot(xgr,ygr,'b');

hold on;

end

time=toc;

ans=x1;

disp('Решение:');

disp(ans);

disp('Число операций:');

disp(m);

disp('Абсолютная погрешность:');

disp(abs(ans-xt));

disp('Время работы:');

disp(time);

Результаты выполнения для 1-ой функции

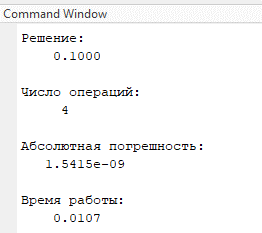


Рисунок 1 – результат выполнения программы

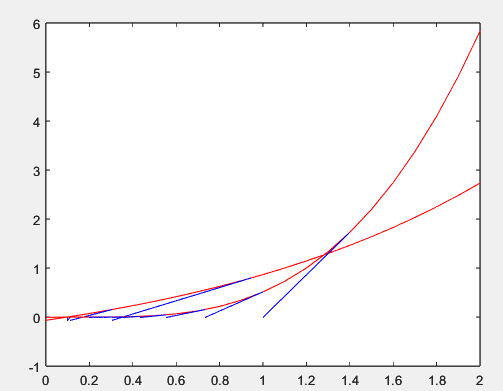


Рисунок 2 – геометрическая интерпретация

Результаты выполнения для 2-ой функции

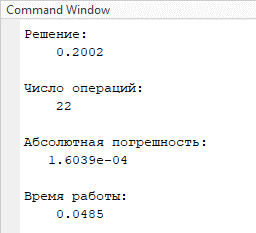


Рисунок 3 – результат выполнения программы

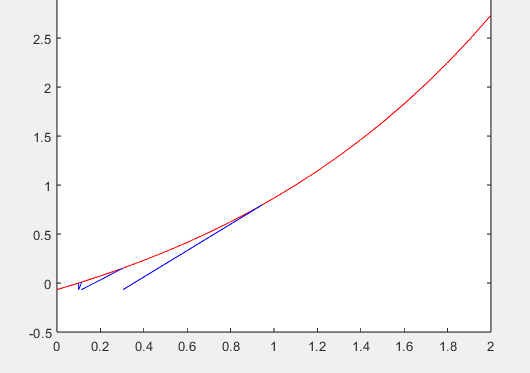


Рисунок 4 – геометрическая интерпретация

**Реализация упрощённого метода Ньютона**

%Первая функция (пример)

%f=@(x)2.^(x-0.1)-1;

%Вторая функция (пример)

f=@(x)(x-0.2).^3;

%Точное значение для первой

%xt=0.1;

%Точное значение для второй

xt=0.2;

%Точность

eps=0.0001;

%Начальное приближение

xn=2;

%Настройка графика

ff=f;

xx=0:0.1:2;

plot(xx,f(xx),'r');

hold on;

%Вычисление производной

symF = sym(f);

df = matlabFunction(diff(symF));

x0=xn;

x1=xn-f(xn)/df(xn);

%счетчик итераций

m=0;

tic;

while abs(x1-x0)>eps

m=m+1;

x0 = x1;

x1=x0-f(x0)/df(xn);

xgr=[x1,x0];

ygr=[f(0),f(x0)];

plot(xgr,ygr,'b');

hold on;

end

time=toc;

ans=x1;

disp('Решение:');

disp(ans);

disp('Число операций:');

disp(m);

disp('Абсолютная погрешность:');

disp(abs(ans-xt));

disp('Время работы:');

disp(time);

Результаты выполнения для 1-ой функции

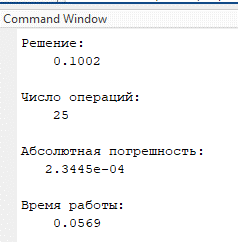


Рисунок 5 – результат выполнения программы

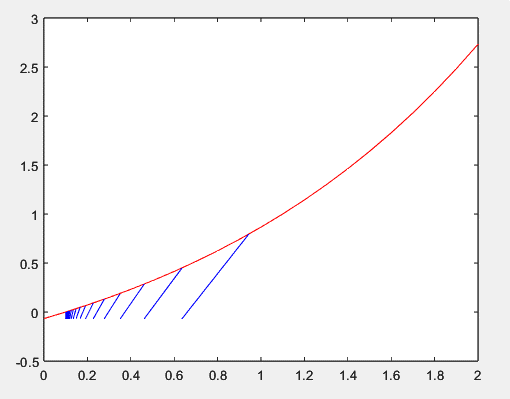


Рисунок 6 – геометрическая интерпретация

Результаты выполнения для 2-ой функции

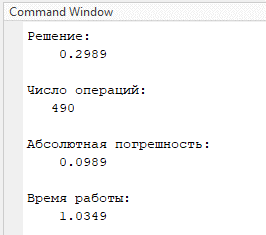


Рисунок 7 – результат выполнения программы

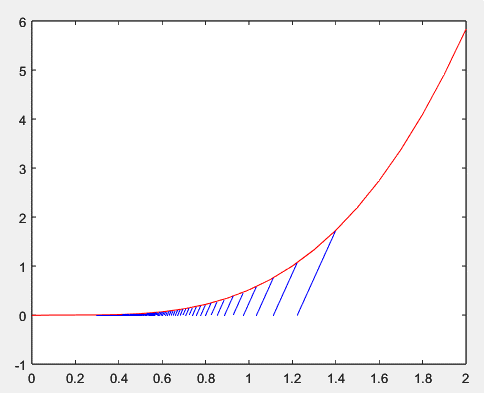


Рисунок 8 – геометрическая интерпретация

**Реализация метода секущих**

%Первая функция (пример)

%f=@(x)2.^(x-0.1)-1;

%Вторая функция (пример)

f=@(x)(x-0.2).^3;

%Точное значение для первой

%xt=0.1;

%Точное значение для второй

xt=0.2;

%Точность

eps = 0.0001;

%Начальное приближение

x1 = 0;

x0 = 1;

ff=f;

xx=0:0.01:0.5;

plot(xx,f(xx),'r');

hold on;

m=0;

tic;

while abs(x1-x0)>eps

m=m+1;

bufx1 = x1;

x1 = x1 - f(x1)\*((x0 - x1)/(f(x0)-f(x1)));

x0 = bufx1;

xgr=[x1,x0];

ygr=[f(x1),f(x0)];

plot(xgr,ygr,'b');

hold on;

end

time=toc;

ans=x1;

disp('Решение:');

disp(ans);

disp('Число операций:');

disp(m);

disp('Абсолютная погрешность:');

disp(abs(ans-xt));

disp('Время работы:');

disp(time);

Результаты выполнения для 1-ой функции

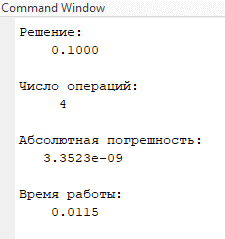


Рисунок 9 – результат выполнения программы

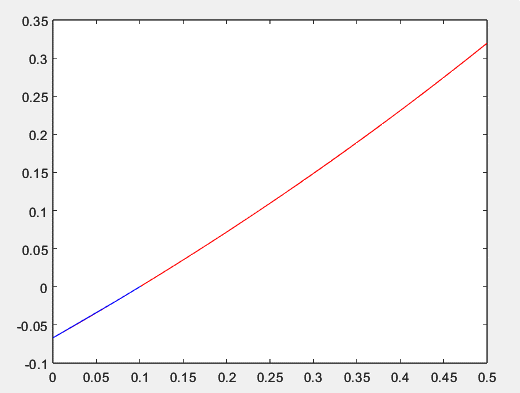


Рисунок 10 – геометрическая интерпретация

Результаты выполнения для 2-ой функции

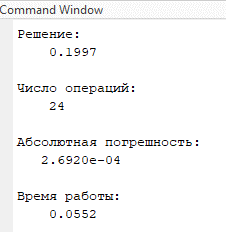


Рисунок 11 – результат выполнения программы

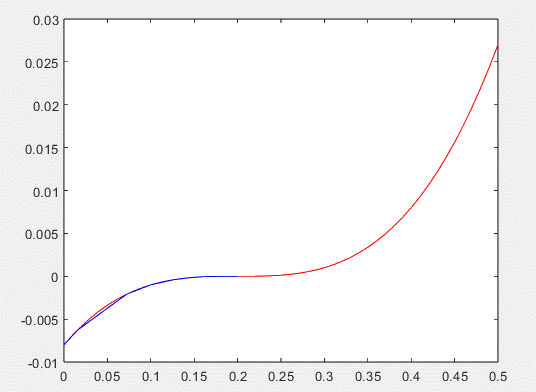


Рисунок 12 – геометрическая интерпретация

**Реализация метода половинного деления**

%Первая функция (пример)

f=@(x)2.^(x-0.1)-1;

%Вторая функция (пример)

%f=@(x)(x-0.2).^3;

eps=0.0001;

%Границы отрезка

x1=0;

x0=1;

%Точное значение для первой

xt=0.1;

%Точное значение для второй

%xt=0.2;

m=0;

tic;

if f(x1)==0

ans=f(x1);

elseif f(x0)==0

ans=f(x0);

else

dx=x0-x1;

while (x0-x1)>eps

m=m+1;

dx=dx/2;

c=x1+dx;

if sign(f(x1))~=sign(f(c))

x0=c;

else

x1=c;

end

end

end

time=toc;

ans=c;

disp('Решение:');

disp(ans);

disp('Число операций:');

disp(m);

disp('Абсолютная погрешность:');

disp(abs(ans-xt));

disp('Время работы:');

disp(time);

Результаты выполнения для 1-ой функции

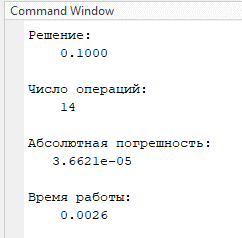


Рисунок 13 – результат выполнения программы

Результаты выполнения для 2-ой функции

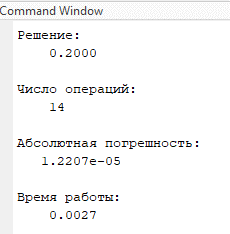


Рисунок 14 – результат выполнения программы

**Нахождение большего нуля**

а) Метод Ньютона

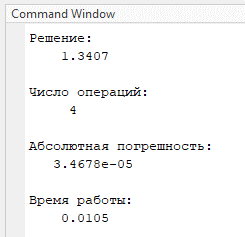


Рисунок 15 – результат выполнения программы

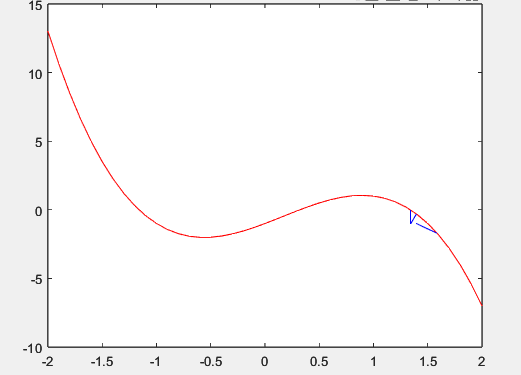


Рисунок 16 – геометрическая интерпретация

б) Упрощённый метод Ньютона

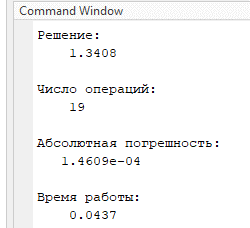


Рисунок 17 – результат выполнения программы

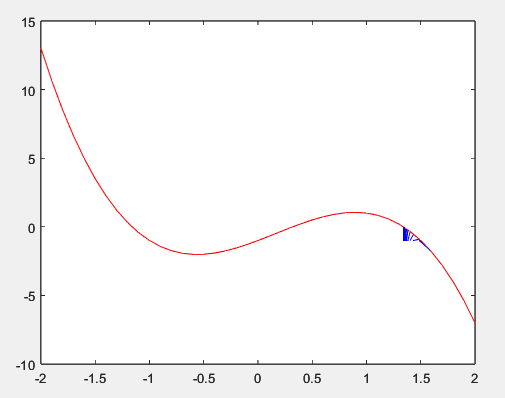


Рисунок 18 – геометрическая интерпретация

в) Метод секущих

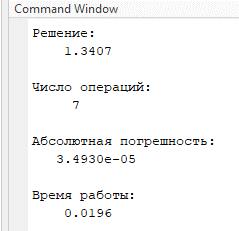


Рисунок 19 – результат выполнения программы

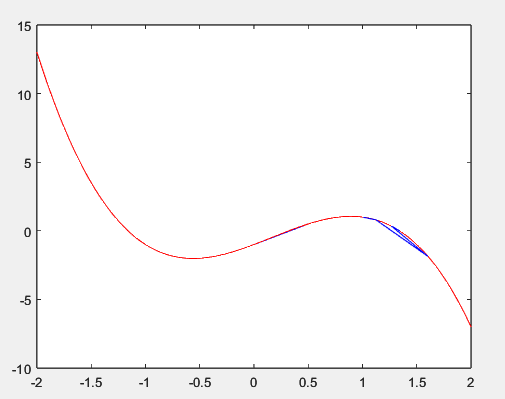


Рисунок 20 – геометрическая интерпретация

в) Метод половинного деления

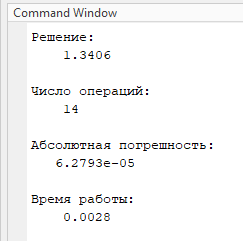


Рисунок 21 – результат выполнения программы

# ЗАКЛЮЧЕНИЕ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Приближенное решение | Абсолютная погрешность | Число итераций | Время работы |
| Ньютона | 1 – 0.1  2 – 0.2002  3 – 1.3407 | 1 – 1  2 – 1  3 – 3 | 1 – 4  2 – 22  3 – 4 | 1 – 0.0107  2 – 0.0485  3 – 0.0105 |
| Ньютона (упрощенный) | 1 – 0.1002  2 – 0.2989  3 – 1.3408 | 1 – 2  2 – 0.0989  3 – 1 | 1 – 25  2 – 490  3 – 19 | 1 – 0.0569  2 – 1.0340  3 – 0.0437 |
| секущих | 1 – 0.1  2 – 0.1997  3 – 1.3407 | 1 –  2 – 2  3 – | 1 – 4  2 – 24  3 – 7 | 1 – 0.0115  2 – 0.0552  3 – 0.0196 |
| половинного деления | 1 – 0.1  2 – 0.2  3 – 1.3406 | 1 –  2 – 1  3 – 6 | 1 – 14  2 – 14  3 – 14 | 1 – 0.0026  2 – 0.0027  3 – 0.0028 |

По результатам можно сделать вывод, что метод половинного деления наиболее быстрый. При этом, количество итераций мало меняется в зависимости от выбранного приближения. Метод Ньютона имеет наименьшее число итераций в лучшем случае, как и метод секущих. Но метод секущих в среднем делает больше итераций, поэтому работает немного медленнее. Упрощённый метод Ньютона показал наихудшие результаты как по числу операций, так и по точности.