Министерство науки и высшего образования
Федеральное государтсвенное бюджетное образовательное учреждение высшего образования
Югорский государственный университет

Отчет о лабораторной работе №4
по дисциплине «Методы оптимизации»

Выполнил

Студент группы 1162б
________________ Панчишин И. Р.
«___» __________ 2019 г.

Принял

Доцент ИЦЭ
________________ Самарин В. А.
«___» __________ 2019 г.

Ханты-Мансийск, 2019

Цель

Изучить численные методы приближенного нахождения корня.

Задачи

1. Рассмотреть метод половинного деления.
2. Рассмотреть метод хорд.
3. Рассмотреть метод Ньютона.

Ход работы

Вывод рекурентной формулы метода хорд:

$$\begin{cases} f(x_1) = kx_1 + b \\ f(x_2) = kx_2 + b \end{cases}$$

Вычтем из первого уравнения второе и выразим k

$$f(x_1) - f(x_2) = k(x_1 - x_2)$$
$$k = \frac{f(x_1) - f(x_2)}{x_1 - x_2}$$

Подставим k в первое уравнение системы и выразим b

$$b = f(x_1) - \frac{x_1}{x_1 - x_2}(f(x_1) - f(x_2))$$

Запишем уравнение прямой (хорды), используя полученные коэффициенты

$$f(x) = \frac{f(x_1) - f(x_2)}{x_1 - x_2}x + f(x_1) - \frac{x_1}{x_1 - x_2}(f(x_1) - f(x_2)) = \frac{f(x_1) - f(x_2)}{x_1 - x_2}(x - x_1) + f(x_1)$$

Выразим значение корня x_3, т. е. $x = x_3$, $y = 0$

$$\frac{f(x_1) - f(x_2)}{x_1 - x_2}(x_3 - x_1) + f(x_1) = 0$$
$$x_3 = -f(x_1)\frac{x_1 - x_2}{f(x_1) - f(x_2)} + x_1$$

Данная форма не требует нахождения производной.

Вывод рекурентной формулы метода Ньютона:

По определению производной

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = f'(x)$$
$$f(x) = f'(x)(x - x_0) + f(x_0)$$

Найдем точку пересечения с абсциссой или первое приближение корня — x_1

$$f'(x_1)(x_1 - x_0) + f(x_0) = 0$$
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_1)}$$

Реализация требуемых методов на языке Octave представлена в листинге ниже:

```
set(0, defaulttaxesfontsize, 12)
set(0, defaulttextfontsize, 12)


% метод половинного деления
% будет работать бесконечно, если на отрезке нет корня
function [xroot yroot n] = bisection(f, a, b, e)
    Ap = [a f(a)]; % A point
    Bp = [b f(b)];
    n = 2;

    [Ap Bp n] = bisection_step(f, Ap, Bp, e, n);

    xroot = (Ap(1) + Bp(1)) / 2;
    yroot = f(xroot);
end

function [Ap Bp n] = bisection_step(f, Ap, Bp, e, n)
    if (abs(Bp(1) - Ap(1)) <= e)
        return
    end

    c = (Ap(1) + Bp(1)) / 2;
    fc = f(c);
    ++n;

    % учтен случай, когда попадается корень
    if (fc * Ap(2) <= 0)
        Bp = [c fc];
    end

    if (fc * Bp(2) <= 0)
        Ap = [c fc];
    end

    [Ap Bp n] = bisection_step(f, Ap, Bp, e, n);
end

% первая производная
function res = der1(f, x0, h)
    res = (f(x0 + h) - f(x0)) / h;
end

% вторая
function res = der2(f, x0)
    h = 0.001;
    d1 = der1(f, x0, h);
    d2 = der1(f, x0 + h, h);
    res = (d2 - d1) / h;
end

% метод хорд
function [xroot yroot n] = chord(f, a, b, e)
    Ap = [a f(a)];
    Bp = [b f(b)];
    n = 2;

    d2 = der2(f, Ap(1));
    if (d2 * Ap(2) > 0) % выбор начального приближения корня
        [Xp n] = chord_step(f, Bp, Ap, e, n);
```

```
    else
        [Xp n] = chord_step(f, Ap, Bp, e, n);
    end

    [xroot yroot] = deal(Xp(1), Xp(2));
end

function [Xp n] = chord_step(f, Xp, Bp, e, n)
    x = Xp(1) - Xp(2) * (Xp(1) - Bp(1)) / (Xp(2) - Bp(2));

    if (abs(x - Xp(1)) <= e)
        return
    end

    fchord = @(X) (Xp(2) - Bp(2)) / (Xp(1) - Bp(1)) * (X - Xp(1)) + Xp(2);
    global X;
    % раскомментировать для визуализации
    %plot(X, fchord(X));

    Xp = [x f(x)];
    ++n;

    [Xp n] = chord_step(f, Xp, Bp, e, n);
end

% метод Ньютона
function [xroot yroot n] = newton(f, a, b, e)
    Ap = [a f(a)];
    Bp = [b f(b)];
    n = 2;

    d2 = der2(f, Ap(1));
    if (d2 * Ap(2) > 0)
        [Xp n] = newton_step(f, Ap, e, n);
    else
        [Xp n] = newton_step(f, Bp, e, n);
    end

    [xroot yroot] = deal(Xp(1), Xp(2));
end

function [Xp n] = newton_step(f, Xp, e, n)
    d1 = der1(f, Xp(1), 0.001);
    x = Xp(1) - Xp(2) / d1;

    if (abs(x - Xp(1)) <= e)
        return
    end

    ftang = @(X) d1 * (X - Xp(1)) + Xp(2);
    global X;
    %plot(X, ftang(X));

    Xp = [x f(x)];
    ++n;

    [Xp n] = newton_step(f, Xp, e, n);
end
```

```
% функции
F = {@(X) 2.^X - 2, @(X) 2.^-X - 2, @(X) -(2.^X - 2), @(X) -(2.^-X - 2)};
global X = linspace(-3, 3, 100);

Fm = {@bisection, @chord, @newton};

%% визуализация работы
%for i = 1:length(Fm)
%    for j = 1:4
%        subplot(2, 2, j);
%        box off;
%        hold on;
%        grid on;
%        set(gca, xaxislocation, origin);
%        set(gca, yaxislocation, origin);
%        plot(X, F{j}(X));
%        xlabel(x);
%        ylabel(y);
%
%        [xroot, yroot, n] = Fm{i}(F{j}, -2, 2.5, 0.2);
%        plot(xroot, yroot, bo, MarkerFaceColor, b);
%    end
%    figure;
%end

% зависимость кол-ва вычислений от точности
hold on;
E = linspace(0.00000001, 0.2, 20);
for i = 1:length(Fm)
    N = [];
    for e = E
        [xroot yroot n] = Fm{i}(F{1}, -2, 2.5, e);
        N = [N n];
    end
    plot(E, N);
end
legend(Половинного деления, Хорд, Ньютона);
xlabel(Погрешность)
ylabel(Вычислений)


pause
```

В коде можно встретить неравенство $f''(x_0)f(x_0) > 0$. Оно описывает обязательное требование к начальному приближению x_0, которым является один из концов отрезка поиска.

Результаты нахождения корня представлены на рисунках **??**, **??**, **??**. На каждом рисунке функция в различных положениях на плоскости. Благодаря правильному определению начального приближения алгоритм сходится во всех случаях.

Зависимости количества вычислений функции для каждого метода представлены на Рис. **??**.

Вывод

Выполнил все поставленные задачи, вывел основные формулы и написал программную реализацию требуемых методов, сравнил их работу. Наиболее эффективным методом приближенного нахождения корня среди рассмотренных оказался метод Ньютона.

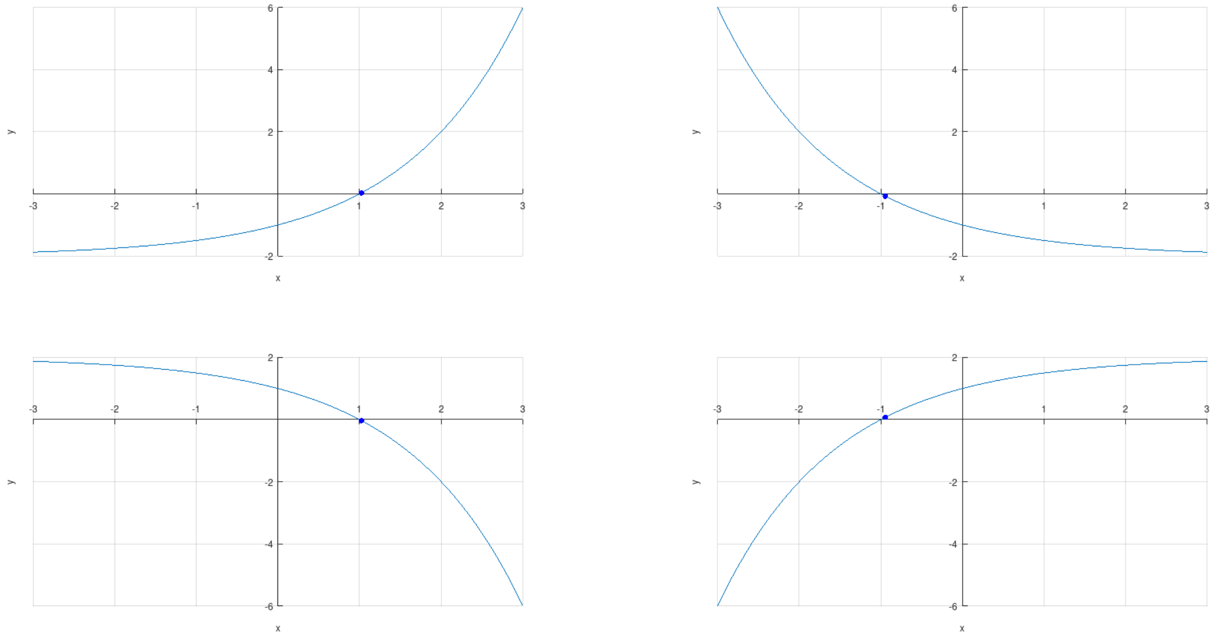

Рис. 1: Метод половинного деления

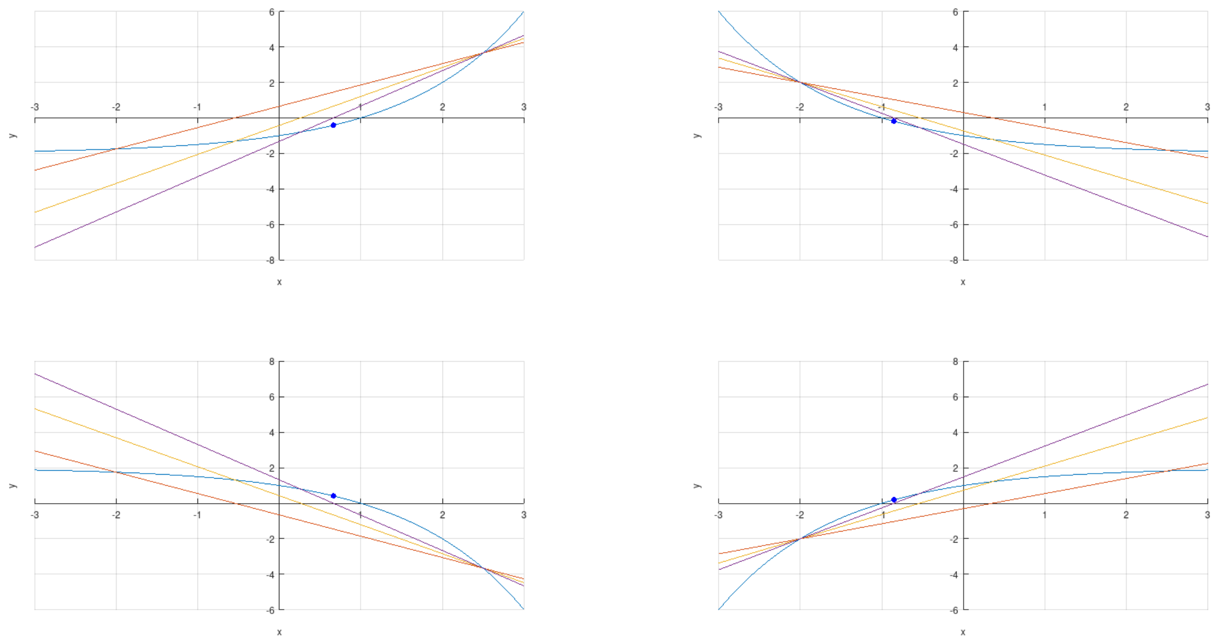

Рис. 2: Метод хорд

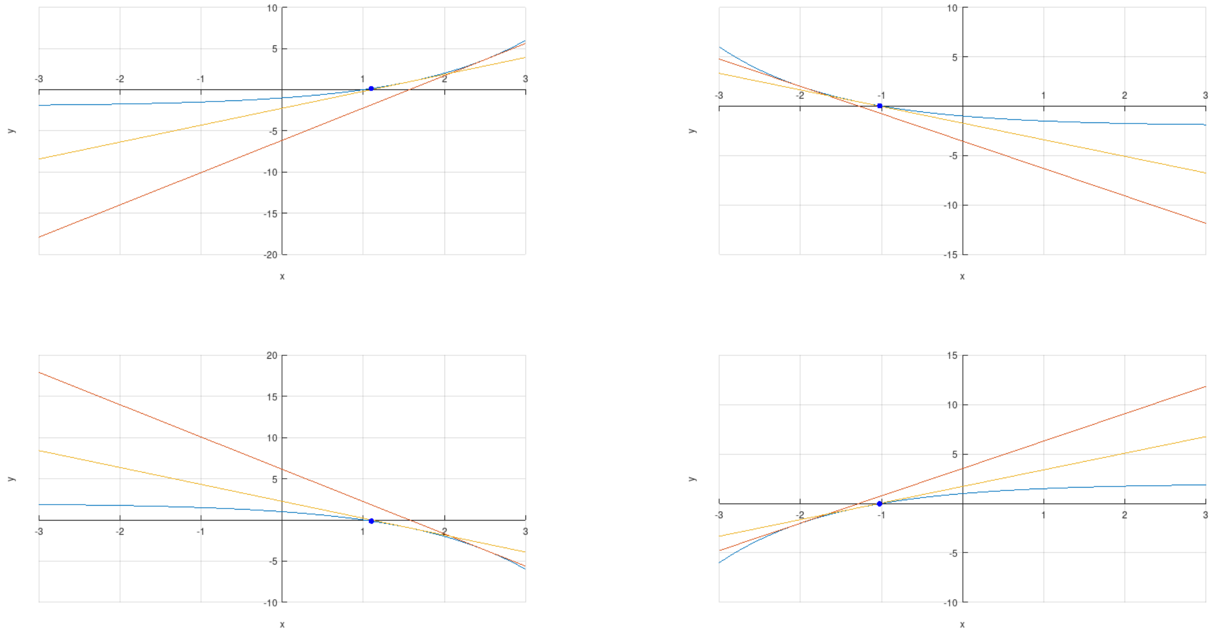

Рис. 3: Метод Ньютона

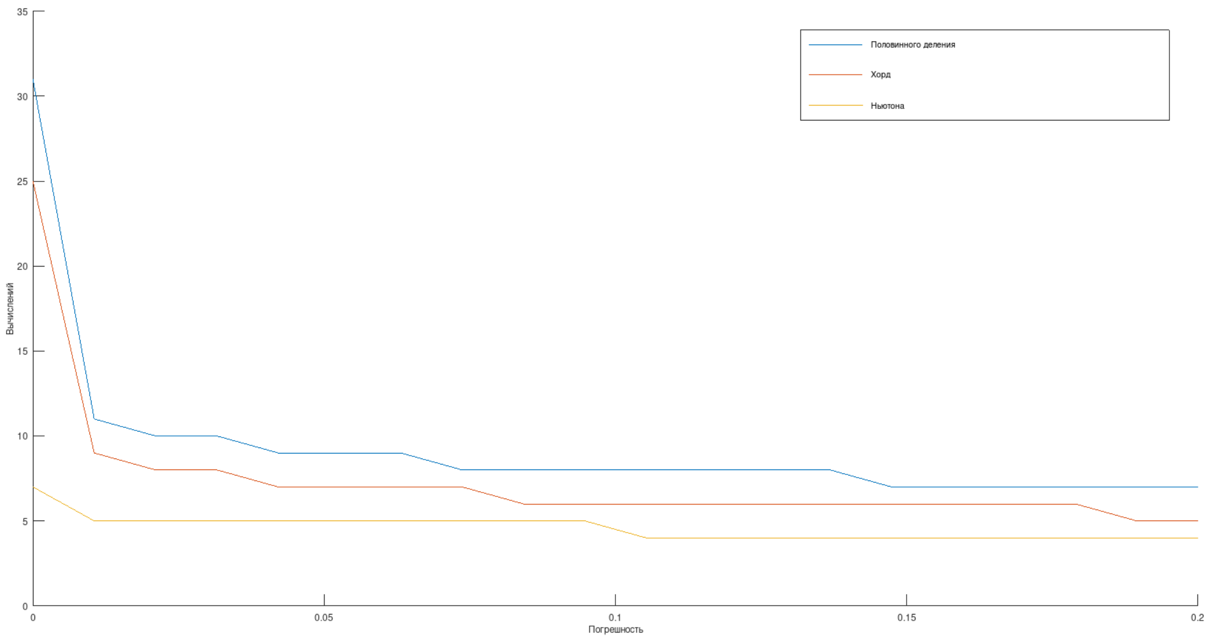

Рис. 4: Скорость работы