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. 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.

```
addpath(../code)

set(0, defaultaxesfontsize, 14)
set(0, defaulttextfontsize, 14)


% целевые функции
F = {@(X) 2.^X - 2, @(X) 2.^-X - 2, @(X) -(2.^X - 2), @(X) -(2.^-X - 2)};
% область определения целевой функции
X = linspace(-3, 3, 100);
% методы приближенного поиска корня
M = {@bisection, @secant, @newton};
Name = {Половинного деления, Хорд, Ньютона};


% хорда
fchord = @(Xp, Bp, X) (Xp(2) - Bp(2)) / (Xp(1) - Bp(1)) * (X - Xp(1)) + Xp(2);
% касательная
ftang = @(Xp, d1, X) d1 * (X - Xp(1)) + Xp(2);

% поиск корня
for i = 1:length(M)
    for j = 1:length(F)
        subplot(2, 2, j);
        box off;
        hold on;
        grid on;
        set(gca, xaxislocation, origin);
        set(gca, yaxislocation, origin);
        xlabel(x);
        ylabel(y);
        plot(X, F{j}(X));

        [xroot, yroot, info] = M{i}(F{j}, -2, 2.5, 0.3)
        title([Name{i} , Шагов  num2str(info.nstep) , Вычислений  num2str(info.ncalc)]);
        plot(xroot, yroot, ro, MarkerFaceColor, r);

        if (i == 2)
            for Approx = info.Approx
                plot(X, fchord(Approx(1:2), Approx(3:4), X));
            end
        end
        if (i == 3)
            for Approx = info.Approx
                plot(X, ftang(Approx(1:2), Approx(3), X));
            end
        end
    end
```

```
    figure
end


% зависимость кол-ва вычислений от точности
hold on;
E = linspace(0.001, 0.5, 100);
for i = 1:length(M)
    N = [];
    for e = E
        [_, _, info] = M{i}(F{1}, -2, 2.5, e);
        N = [N, info.ncalc];
    end
    plot(E, N);
end
legend(Name);
xlabel(Погрешность)
ylabel(Вычислений)


pause
```

```
% Copyright © 2019 Panchishin Ivan

% метод половинного деления
% bisection method

% будет работать бесконечно, если на отрезке нет корня
function [xroot, yroot, info] = bisection(f, a, b, e)
    Ap = [a, f(a)];
    Bp = [b, f(b)];

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

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

function [Ap, Bp, info] = bisection_step(f, Ap, Bp, e)
    c = (Ap(1) + Bp(1)) / 2;
    fc = f(c);

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

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

    info.nstep = 1;
    info.ncalc = 1;

    if (abs(Bp(1) - Ap(1)) <= e)
        return
    end

    [Ap, Bp, info1] = bisection_step(f, Ap, Bp, e);
```

```
    info.nstep += info1.nstep;
    info.ncalc += info1.ncalc;
end
```

```
% Copyright © 2019 Panchishin Ivan

% метод хорд
% secant method

% Approx - позволяет восстановить хорду

function [xroot, yroot, info] = secant(f, a, b, e)
    Ap = [a, f(a)];
    Bp = [b, f(b)];

    [d2, d2ncalc] = deriv2(f, Ap(1));
    if (d2 * Ap(2) < 0)
        X0p = Ap;
    else
        X0p = Bp;
        Bp = Ap;
    end

    [X1p, info] = secant_step(f, X0p, Bp, e);

    info.ncalc += 2 + d2ncalc;
    [xroot, yroot] = deal(X1p(1), X1p(2));
end

function [X1p, info] = secant_step(f, X0p, Bp, e)
    x1 = X0p(1) - X0p(2) * (X0p(1) - Bp(1)) / (X0p(2) - Bp(2));
    X1p = [x1, f(x1)];

    info.nstep = 1;
    info.ncalc = 1;
    info.Approx = [X0p, Bp];

    if (abs(X1p(1) - X0p(1)) <= e)
        return
    end

    [X1p, info1] = secant_step(f, X1p, Bp, e);

    info.nstep += info1.nstep;
    info.ncalc += info1.ncalc;
    info.Approx = [info.Approx; info1.Approx];
end
```

```
% Copyright © 2019 Panchishin Ivan

% метод Ньютона
% Newtons method

% Approx - позволяет восстановить касательную

function [xroot, yroot, info] = newton(f, a, b, e)
    Ap = [a, f(a)];
    Bp = [b, f(b)];
```

```

    [d2, d2ncalc] = deriv2(f, Ap(1));
    if (d2 * Ap(2) < 0) % выбор начального приближения корня
        X0p = Bp;
    else
        X0p = Ap;
    end

    [X1p, info] = newton_step(f, X0p, e);

    info.ncalc += 2 + d2ncalc;
    [xroot, yroot] = deal(X1p(1), X1p(2));
end

function [X1p, info] = newton_step(f, X0p, e)
    [d1, d1ncalc] = grad(f, X0p(1));
    x1 = X0p(1) - X0p(2) / d1;
    X1p = [x1, f(x1)];

    info.nstep = 1;
    info.ncalc = 1 + d1ncalc;
    info.Approx = [X0p, d1];

    if (abs(X1p(1) - X0p(1)) <= e)
        return
    end

    [X1p, info1] = newton_step(f, X1p, e);

    info.nstep += info1.nstep;
    info.ncalc += info1.ncalc;
    info.Approx = [info.Approx; info1.Approx];
end
```

```
% Copyright © 2019 Panchishin Ivan

% вторая производная в точке
% second derivative value

function [res, ncalc] = deriv2(f, x0, h)
    if (nargin < 3)
        h = 0.001;
    end

    [d, ncalc] = grad(f, x0, h);
    [dh, ncalc1] = grad(f, x0 + h, h);
    ncalc += ncalc1;

    res = (dh - d) / h;
end
```

```
% Copyright © 2019 Panchishin Ivan

% градиент функции многих переменных (или производная функции одной переменной) в точке
% function gradient value

function [Gx0, ncalc] = grad(F, X0, h)
    y0 = F(X0);
    ncalc = 1;
```

```

    if (nargin < 3)
        h = 0.001;
    end

    Gx0 = [];
    for i = 1:length(X0)
        Xh = [X0(1:i-1), X0(i) + h, X0(i+1:end)];
        Gx0 = [Gx0, (F(Xh) - y0) / h];
        ++ncalc;
    end
end
```

Результаты нахождения корня представлены на рисунках 1, 2, 3. На каждом рисунке функция в различных положениях на плоскости. Благодаря правильному определению начального приближения алгоритм сходится во всех случаях.

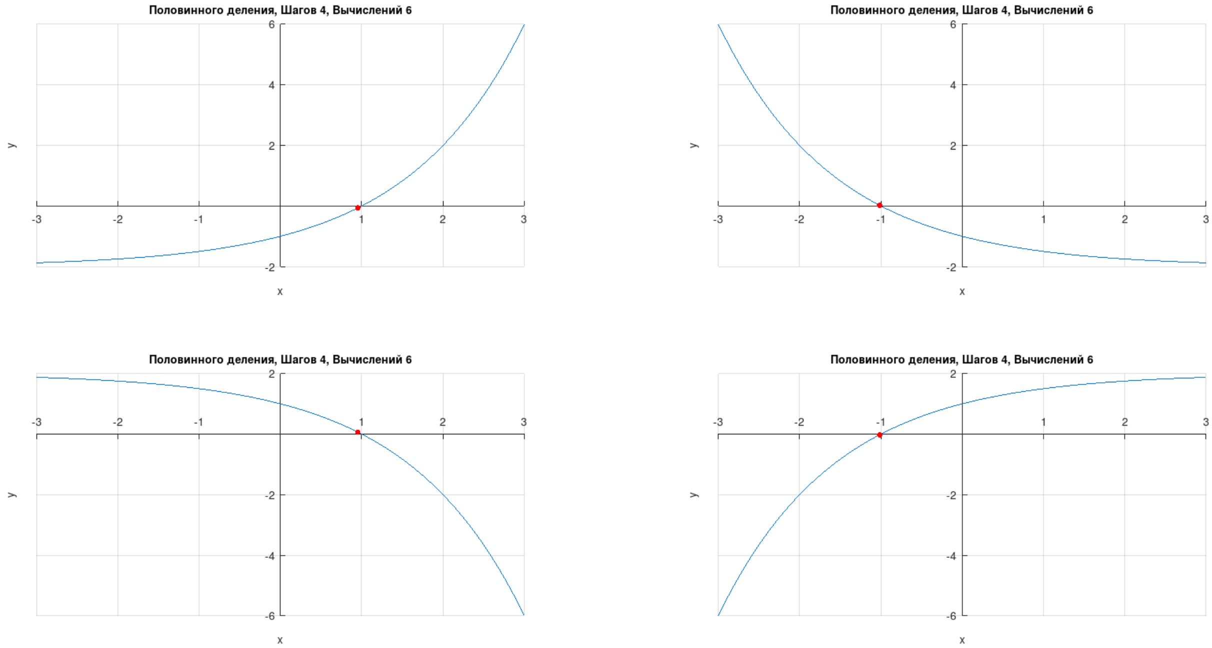

Рис. 1: Метод половинного деления

Зависимости количества вычислений функции от точности для каждого метода представлены на Рис. 4.

Вывод

Выполнил все поставленные задачи, вывел основные формулы и написал программную реализацию требуемых методов, сравнил их работу. Несмотря на то, что метод Ньютона показал наибольшее количество вычислений (которое обусловлено нахождением производной), он справился с задачей за наименьшее количество шагов.

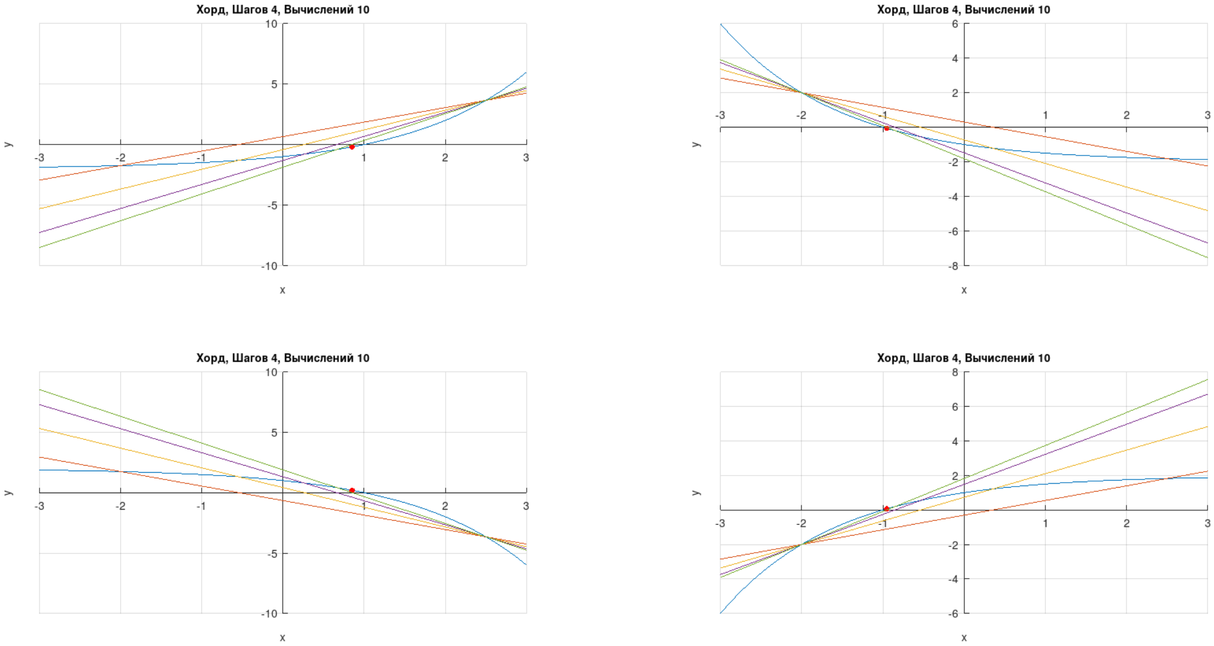

Рис. 2: Метод хорд

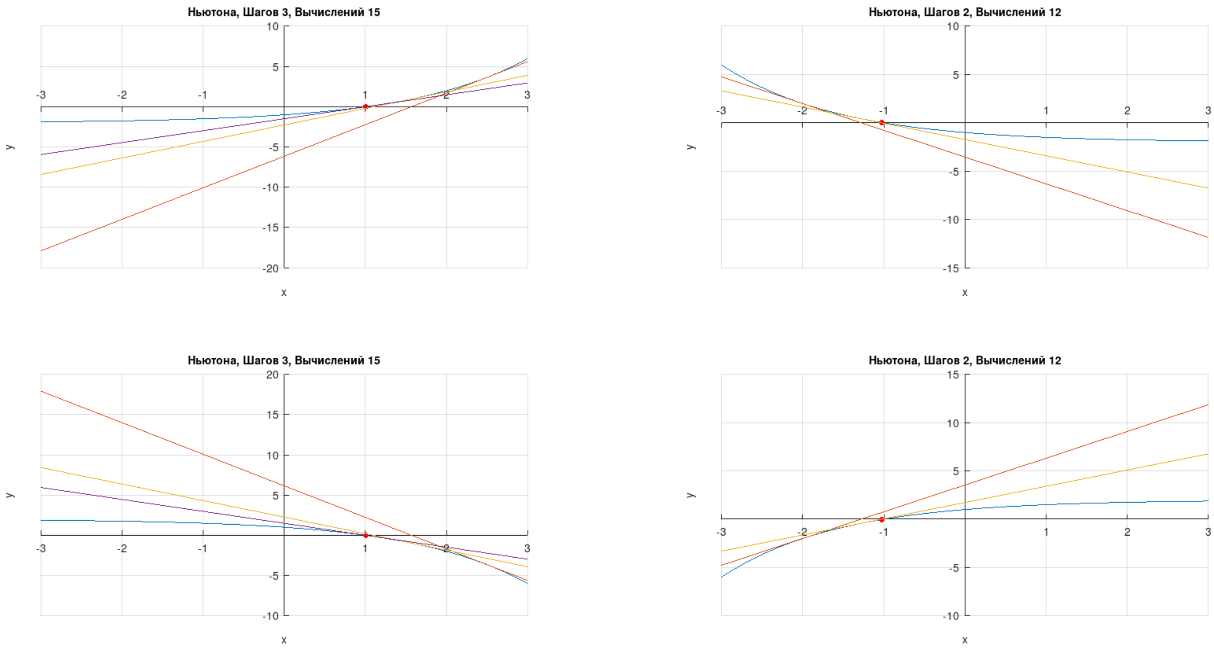

Рис. 3: Метод Ньютона

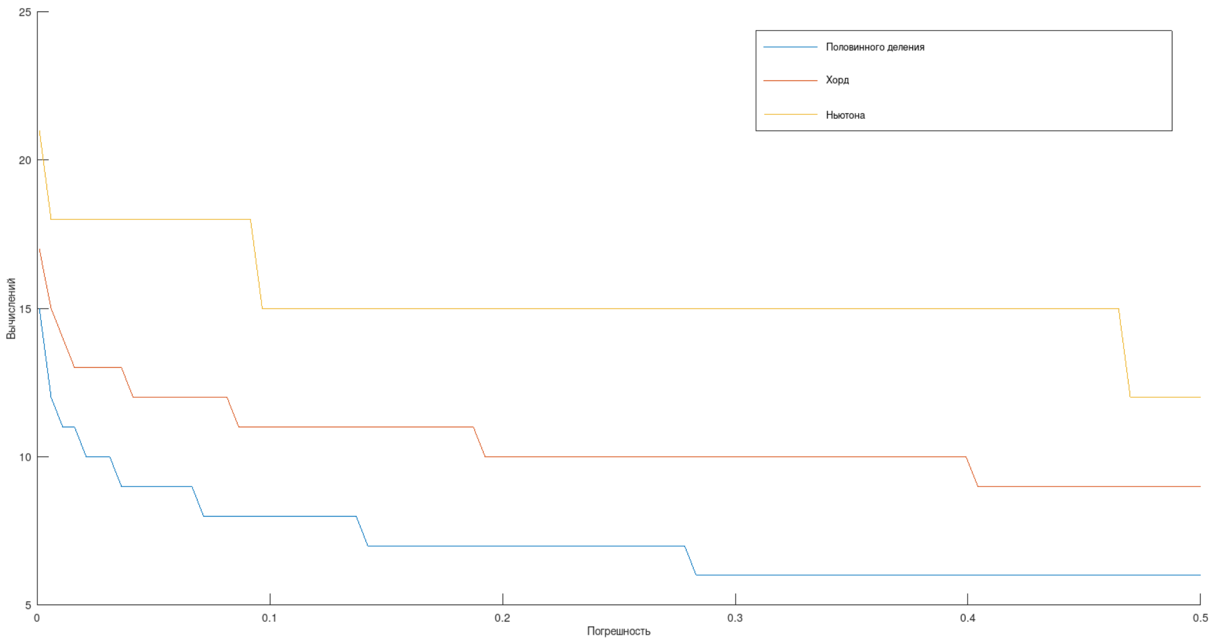

Рис. 4: Скорость работы