Московский государственный технический университет

им. Н.Э. Баумана

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Лабораторная работа № 4 по курсу

«Методы вычислений»

**«Метод Ньютона»**

Работу выполнил

студент группы ИУ7-23М

Кадыров Руслан

Москва, 2018г.

**Цель работы**

Изучение метода Ньютона для решения задачи одномерной минимизации.

**Постановка задачи**

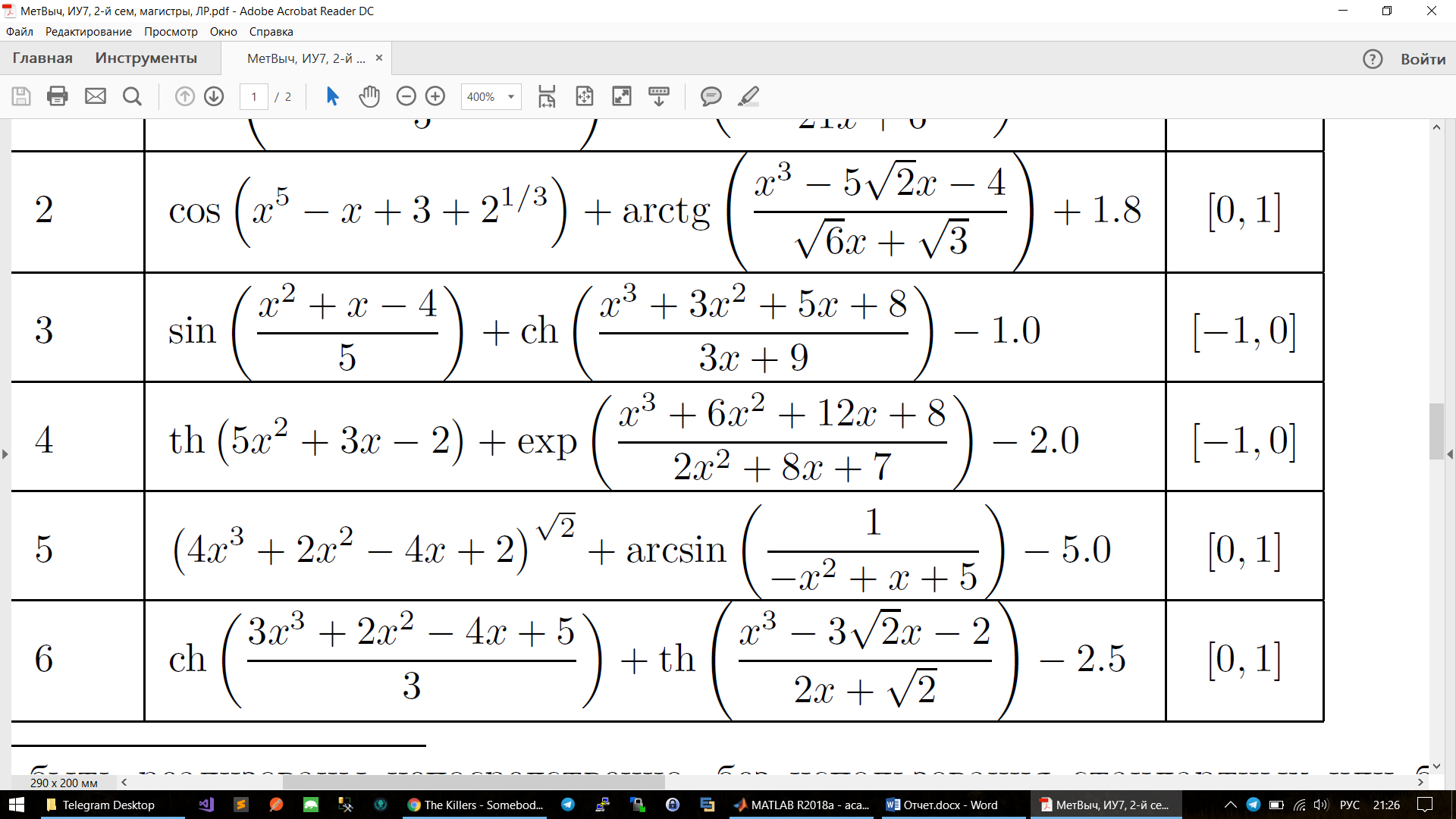
1. Реализовать модифицированный метод Ньютона с конечно-разностной аппроксимацией производных в виде программы на ЭВМ;
2. Провести решение задачи

для индивидуального варианта;

1. Организовать вывод на экран графика целевой функции, найденной точки минимума (*x\**, *f* (*x\**)) и последовательности точек (*xi*, *f* (*xi*)), приближающих точку искомого минимума;
2. Провести решение задачи с использованием стандартной функции fminbnd пакета MATLAB.

Исходные данные задачи: вариант №6

Целевая функция



на отрезке [0,1];

**Описание метода Ньютона**

Метод Ньютона – итерационный численный метод нахождения корня (нуля) заданной функции. Поиск решения осуществляется путём построения последовательных приближений и основан на принципах простой итерации. Метод обладает квадратичной сходимостью. В случае решения задач оптимизации предполагается, что функция *f(x)* дважды непрерывно дифференцируема. Отыскание минимума функции *f(x)* производится при помощи отыскания стационарной точки *x\**, т.е. точки, удовлетворяющей уравнению *f'(x) = 0*, которое решается методом Ньютона.

Приведем алгоритм метода Ньютона.

**Шаг 1.** Выбрать начальную точку *x.* Если *f(a)\* f ``(a)> 0*, положить *x* = *a*, иначе *x = b.*

**Шаг 2.** Найти приближение корня *x*i+1 по выражению *xi+1 = xi  - f(xi) / f ’(xi).*

**Шаг 3.** Проверка на окончание поиска. Если *| xi+1 - xi |<**ε*, то вычисления завершить. Положить *x\* = xi+1, f\* = f(x\*)*, иначе перейти к шагу 2.

**Текст программы**

Листинг скрипта MetodyVich\_Lab\_4.m

eps = 0.01;

a = 0;

b = 1;

FuncString = 'cosh((3\*(x^3) + 2\*(x^2) - 4\*x + 5)/3) + tanh((x^3 - 3\*sqrt(2)\*x -2)/(2\*x + sqrt(2))) - 2.5;'

[xRes, fRes, xi, fi, iterCount] = NewtonMethod(FuncString,a,b, eps);

%Получение данных для построения графика целевой функции

xArr = zeros(1,iterCount);

fArr = zeros(1,iterCount);

a = 0;

step = 1;

while(step < 600)

xArr(step) = step\*0.01 + a;

fArr(step) = Func(step\*0.01 + a);

step = step + 1;

end

%Получение данных для построения точек, приближающихся к минимуму

xI\_Arr = zeros(1,iterCount);

fI\_Arr = zeros(1,iterCount);

step = 1;

if(iterCount > 0)

for i=1:iterCount

if(xi(i)~=0)

xI\_Arr(step) = xi(i);

fI\_Arr(step) = fi(i);

step = step + 1;

end

end

end

fprintf('Решение с помощью метода Ньютона \n');

fprintf('Количество вычислений целевой функции: %1.d.\n',iterCount);

fprintf('x\* = %1.10f.\n',xRes);

fprintf('f(x\*) = %1.10f.\n',fRes);

[xRes, fRes, xi, fi, iterCount] = StandartFunc(FuncString,a,b);

fprintf('Решение с помощью стандартной функции fminbnd \n');

fprintf('Количество вычислений целевой функции: %1.d.\n',iterCount);

fprintf('x\* = %1.10f.\n',xRes);

fprintf('f(x\*) = %1.10f.\n',fRes);

plot(xArr, fArr,xRes, fRes, 'ro',xI\_Arr,fI\_Arr,'k\*');

grid on;

title('График целевой функции f(x)');

xlabel('x');

ylabel('Значение целевой функции f(x)');

ylim([-2 15])

%Вычисление значения целевой функции в точке х

function X = Func(x)

X = cosh((3\*(x^3)+2\*(x^2)-4\*x+5)/3)+tanh((x^3-3\*sqrt(2)\*x-2)/(2\*x+sqrt(2)))-2.5;

end

%Метод Ньютона

function [xResult, fResult, xI,fI, iterationCount] = NewtonMethod(FunctionString, a, b, eps)

% a - начало отрезка, b - конец отрезка, eps - точность поиска

% xResult - оптимальный x\*, %fResult - значение целевой функции в x\*

% xI-последовательность xi, приближающих точку искомого минимума

% fI-последовательность fi, приближающих точку искомого минимума

% iterationCount - число вычислений значения целевой функции

inlineFunc = inline(FunctionString);

iterCount = 0;

xI = zeros(0, MaxIterationCount());

fI = zeros(0, MaxIterationCount());

df = char(diff(str2sym(FunctionString)));

ddf = char(diff(str2sym(df)));

dfun = inline(df);

ddfun = inline(ddf);

if (inlineFunc(a) \* ddfun(a) > 0)

xResult = b;

else

xResult = a;

end

while (abs(dfun(xResult)) > eps)

X0 = xResult; % X0 - значение предыдущего шага

xResult = X0 - (dfun(X0)/(ddfun(X0))); % расчет нового значения

fResult = inlineFunc(xResult);

iterCount = iterCount + 1;

xI(iterCount) = xResult;

fI(iterCount) = fResult;

end

iterationCount = iterCount;

end

% Стандартная функция fminbnd

function [xResult, fResult, xI, fI, iterCount, iterOverflow] = StandartFunc(FuncString, a, b)

[xResult, fResult, exitflag, output] = fminbnd(FuncString, a, b);

iterCount = output.funcCount;

xI = zeros(0, MaxIterationCount());

fI = zeros(0, MaxIterationCount());

end

function num = MaxIterationCount

num = 250;

end

**Результаты расчета задачи индивидуального варианта**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | ***ε*** | *N* | *x\** | *f (x\*)* |
| 1 | 10-2 | 3 | 0.4824197102 | -1.4738661812 |
| 2 | 10-4 | 3 | 0.4824197102 | -1.4738661812 |
| 3 | 10-6 | 4 | 0.4824144999 | -1.4738932844 |

**Сводная таблица результатов**

**расчета задачи индивидуального варианта для** ε = 10−6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | ***Метод*** | *N* | *x\** | *f (x\*)* |
| 1 | поразрядного поиска | 49 | 0.4824180603 | -1.4738932844 |
| 2 | золотого сечения | 31 | 0.4824181966 | -1.4738932844 |
| 3 | парабол | 3 | 0.4824182540 | -1.4738932844 |
| 4 | Ньютона модифицированный | 4 | 0.4824144999 | -1.4738932843 |
| 5 | Функция fminbnd | 9 | 0.4824218012 | -1.4738932843 |