Московский государственный технический университет

им. Н.Э. Баумана

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

Лабораторная работа № 2 по курсу

«Методы вычислений»

**«Метод золотого сечения»**

Работу выполнил

студент группы ИУ7-23М

Кадыров Руслан

Москва, 2018г.

**Цель работы**

Изучение метода золотого сечения для решения задачи одномерной минимизации.

**Постановка задачи**

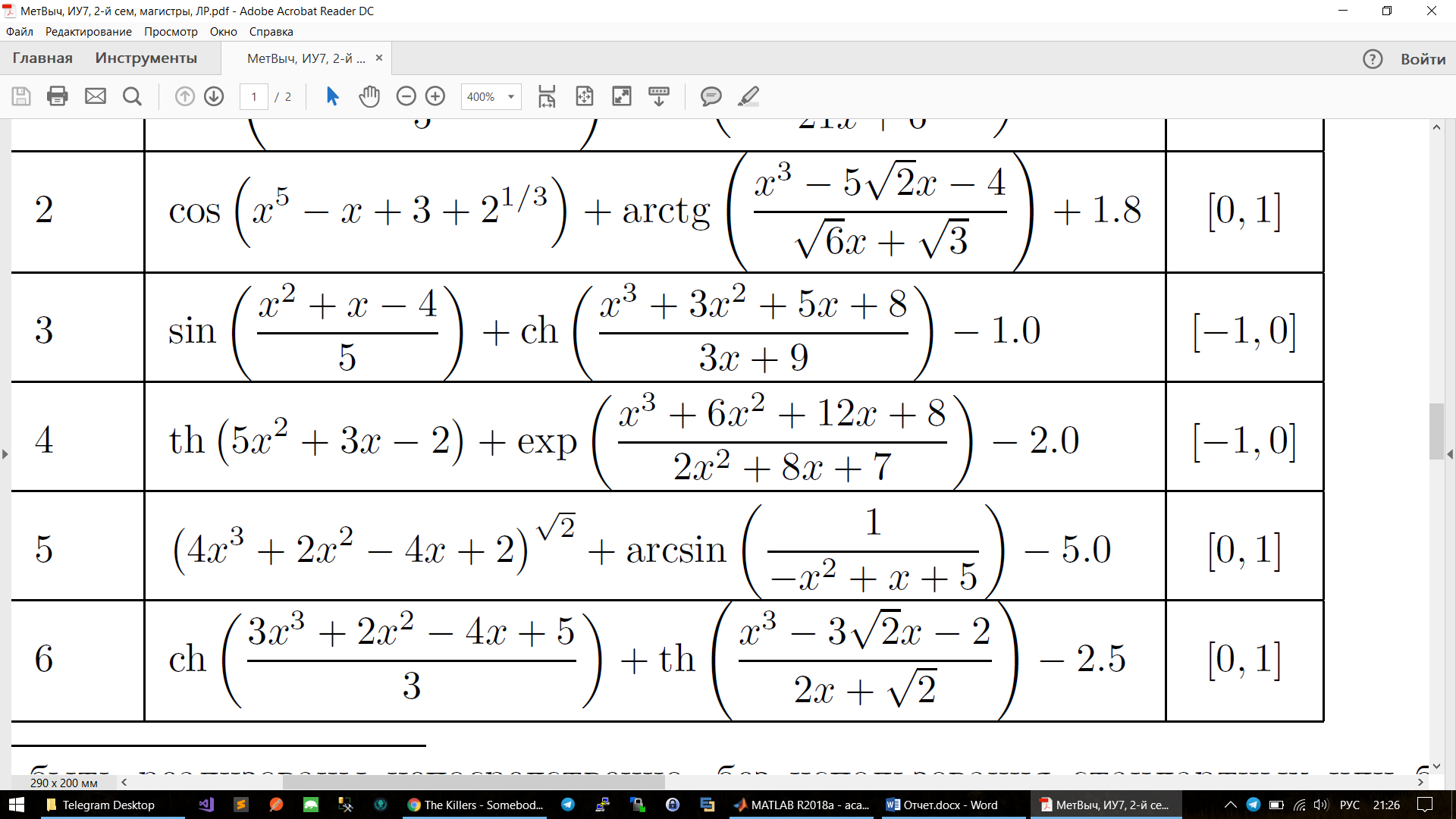
1. Реализовать метод золотого сечения в виде программы на ЭВМ;
2. Провести решение задачи

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.

Исходные данные задачи: вариант №6

Целевая функция



на отрезке [0,1];

**Описание метода золотого сечения**

Для использования конкретного метода одномерной минимизации, работающего по принципу последовательного сокращения интервала неопределённости, следует задать правило выбора на каждом шаге двух внутренних точек. Желательно, чтобы одна из них всегда использовалась в качестве внутренней и для следующего интервала. Тогда число вычислений функции сократится вдвое и одна итерация потребует расчёта только одного нового значения функции.

Говорят, что точка *x* осуществляет золотое сечение отрезка *[a, b]*, если

\frac{b-a}{b-x}=\frac{b-x}{x-a}=\phi=\frac{1+\sqrt{5} }{2}

В качестве *x1* и *x2*выберем точку золотого сечения отрезка и симметричную ей. Если *a < x1 < x2 < b*, то при указанном выборе точек получаем, что *x1* - точка золотого сечения отрезка [*a, x2*], а *x2* - точка золотого сечения отрезка [*x1, b*]. Таким образом, на каждом шаге, кроме первого, необходимо вычислять значение только в одной точке, вторая берется из предыдущего шага.

Приведем алгоритм метода золотого сечения.

**Шаг 1.** Задать значения начального интервала неопределенности, точность *ε> 0*.

**Шаг 2.** Вычислить *x*1 = *b* – *(b – a*) / φ, вычислить *x*2 = *a* + (*b – a*) / φ.

**Шаг 3.** Вычислить *f(x1)* и *f(x2)*.

**Шаг 4.** Сравнить *f(x1)* и *f(x2)*.

1. Если *f(x1)* > *f(x2)*, то положить *a =* *x*1 , *x*1 = *x2, x2 = b – (x*1 *- a);*
2. Если *f(x1)* < *f(x2)*, то положить *b =* *x*2 , *x*2 = *x1, x1 = a + (b – x*2*).*

**Шаг 5.** Вычислить (b – a)/2 и проверить условие окончания:

1. Если (b – a)/2 < *ε,* процесс поиска завершается. x\*. В качестве приближенного решения можно взять середину последнего интервала. x\* = (a + b) / 2;
2. Если (b – a)/2 >= *ε,* перейти к шагу 3.

**Текст программы**

Листинг скрипта MetodyVich\_Lab\_2.m

eps = 0.01;

a = 0;

b = 1;

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

%Получение данных для построения графика целевой функции

xArr = zeros(1,iterCount);

fArr = zeros(1,iterCount);

a = -4;

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

xiArr = zeros(1,iterCount);

fiArr = zeros(1,iterCount);

step = 1;

for i=1:MaxIterationCount()

if(xi(i)~=0)

xiArr(step) = xi(i);

fiArr(step) = fi(i);

step = step + 1;

end

end

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', xiArr, fiArr, 'b\*');

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)

firstPart = cosh((3\*(x^3) + 2\*(x^2) - 4\*x + 5)/3);

secondPart = tanh((x^3 - 3\*sqrt(2)\*x -2)/(2\*x + sqrt(2)));

X = firstPart + secondPart - 2.5;

end

%Метод золотого сечения

function [xResult, fResult, xI,fI, iterationCount] = GoldenSection(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

xi = zeros(1,MaxIterationCount());

fi = zeros(1,MaxIterationCount());

phi = (1 + sqrt(5)) / 2;

x1 = b - (b - a)/phi;

x2 = a + (b - a)/phi;

xi(1) = x1;

xi(2) = x2;

f1 = Func(x1);

f2 = Func(x2);

fi(1) = f1;

fi(2) = f2;

iter = 2;

while(abs(b - a) > eps)

iter = iter + 1;

if(f1 < f2)

b = x2;

x2 = x1;

f2 = f1;

x1 = a + (b - a)/phi;

f1 = Func(x1);

else

a = x1;

x1 = x2;

f1 = f2;

x2 = b - (b - a)/phi;

f2 = Func(x2);

end

xi(iter) = (x1 + x2)/2;

fi(iter) = Func((x1 + x2)/2);

end

iterationCount = iter;

xI = xi;

fI = fi;

xResult = (x1 + x2)/2;

fResult = Func(xResult);

end

function num = MaxIterationCount

num = 150;

end

**Результаты расчета задачи индивидуального варианта**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | ***ε*** | *N* | *x\** | *f (x\*)* |
| 1 | 10-2 | 9 | 0.4918693812 | -1.4735659896 |
| 2 | 10-4 | 14 | 0.4923020569 | -1.4735352616 |
| 3 | 10-6 | 19 | 0.4922985390 | -1.4735355170 |