**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ**

**Факультет физико-математических и естественных наук**

**Кафедра «Математического моделирования и искусственного интеллекта»**

Компьютерный практикум

Лабораторная работа №1

«Численные методы поиска корней уравнений »

|  |  |
| --- | --- |
| Студент | Камалиева Лия Дамировна |
| Группа | НБИбд-03-23 |
|  |  |

**Москва**

**2024**

**Оглавление**

Введение. 3

1.Операции с комплексными числами 4

1.1. Сумма комплексных чисел 4

1.2. Разность комплексных чисел 5

1.3. Произведение комплексных чисел 5

1.4. Частное комплексных чисел 6

2.Безусловная одномерная оптимизация. 8

1.1.Метод дихотомии(деление пополам). 8

1.2.Метод итераций(последовательного приближения). 9

1.3.Метод секущих(хорд) 10

1.4.Метод Ньютона(метод касательных) 10

2.Алгоритмы. 11

3.Программа реализации алгоритмов 16

Заключение. 27

Литература 28

# Введение.

В данной работе будут рассмотрены алгоритмы работы с комплексными числами и алгоритмы одномерной оптимизации методами дихотомии, итераций, секущих и касательных. Целью данной работы является ознакомление с методами одномерной оптимизации и поработать с комплексными числами.

В первой части работы приведена теория, касающаяся оптимизации, в общем, методов дихотомии, итераций, секущих и касательных, в частном, также будет сказано и о комплексных числах.

Во второй части работы приведены алгоритмы данных методов оптимизации.

В третьей части реализована сама программа.

# Операции с комплексными числами.

Комплексным числом z называют выражение:

z = a + i \* b,

где а и b – действительные числа, i – мнимая единица,

определяемая равенством:

i =√(-1); i2 = -1

Если a = 0, то число i b называется чисто мнимым

Если b = 0, то получается действительное число a.

Множество комплексных чисел обычно обозначают буквой C , а элементы этого множества – буквой z.

Вещественное число a будем называть вещественной частью комплексного числа a + bi и обозначать Re(a + bi ) или Re z , а b - его мнимой частью и обозначать Im(a + bi) или Im z. ( Re и Im - начальные буквы латинских слов realis – действительный и imaginarius мнимый).

Например, Re (2 – 3i) = 2, Im(2-3i) = -3

Таким образом, операции 1)-3) можно сформулировать следующим образом:

два комплексных числа равны между собой тогда и только тогда, когда равны их вещественные и мнимые части

## Сумма комплексных чисел.

Сумма двух комплексных чисел, записанных в алгебраической форме, — это комплексное число, действительная часть которого и коэффициент при мнимой части равны соответственно сумме действительных частей и сумме коэффициентов при мнимых частях слагаемых.

Сложение комплексных чисел в алгебраической форме



можно записать с помощью формулы:



Свойства при сложении:

1) коммутативный закон сложения

(a + b) = b + a

2) ассоциативный закон сложения

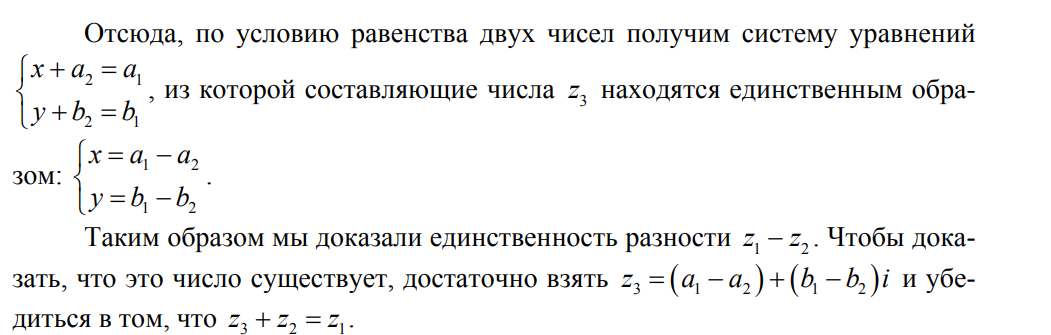
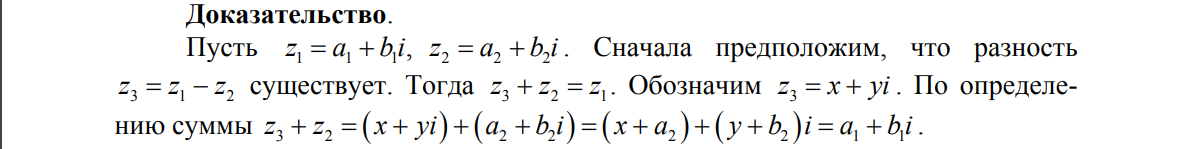
(a + b) + c = a + (b + c)

## Разность комплексных чисел

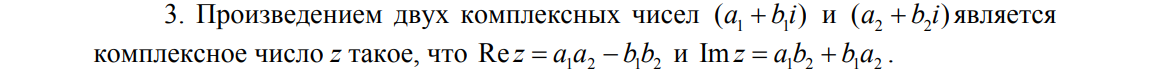
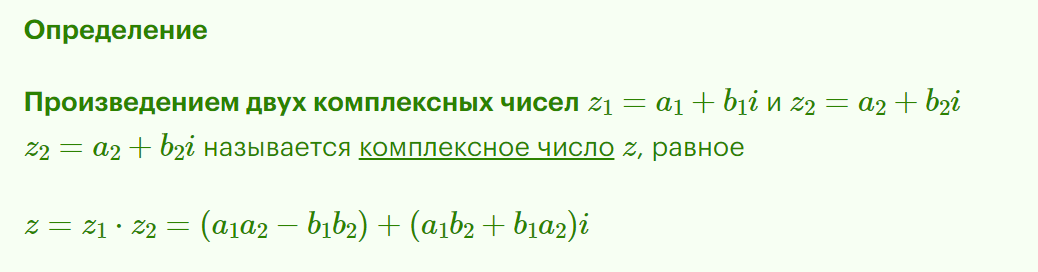
Разностью двух комплексных чисел z1  - z2 называется такое число z3 , которое в сумме с вычитаемым дает уменьшаемое, т.е. z3 + z2 = z1.

Теорема.

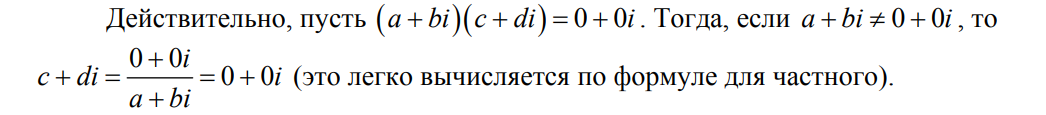
Для любой пары комплексных чисел z1 и z2 существует разность z1 - z2 , причем это число единственное.



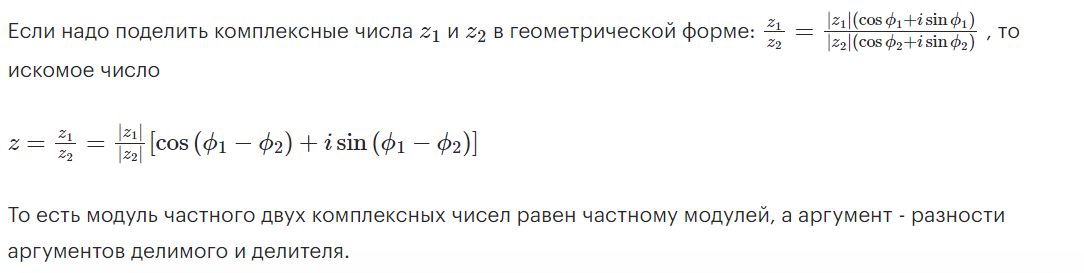
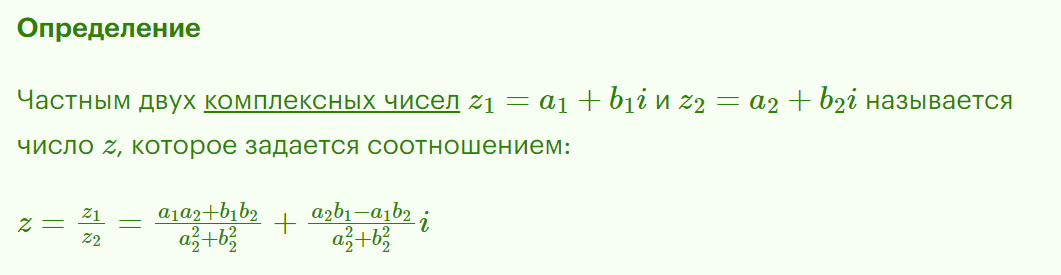
**1.3. Произведение комплексных чисел**



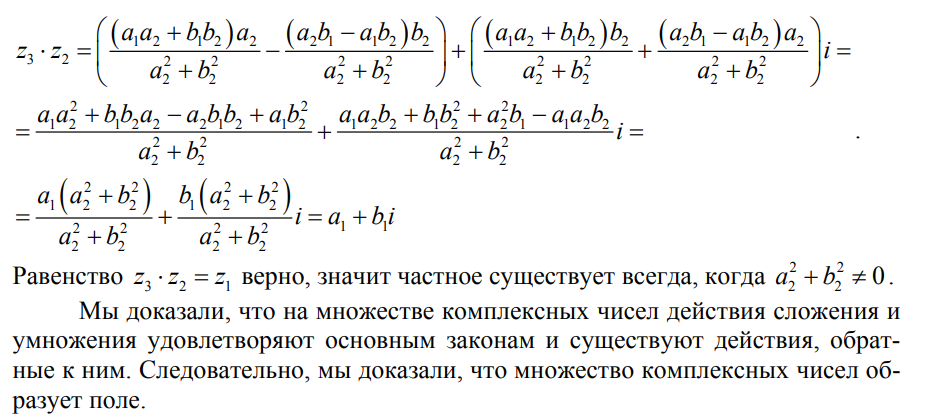
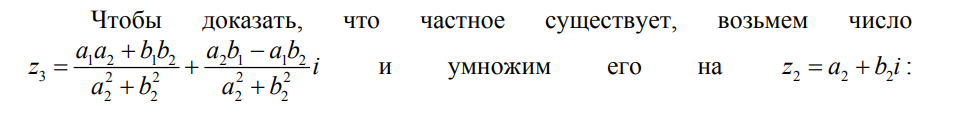
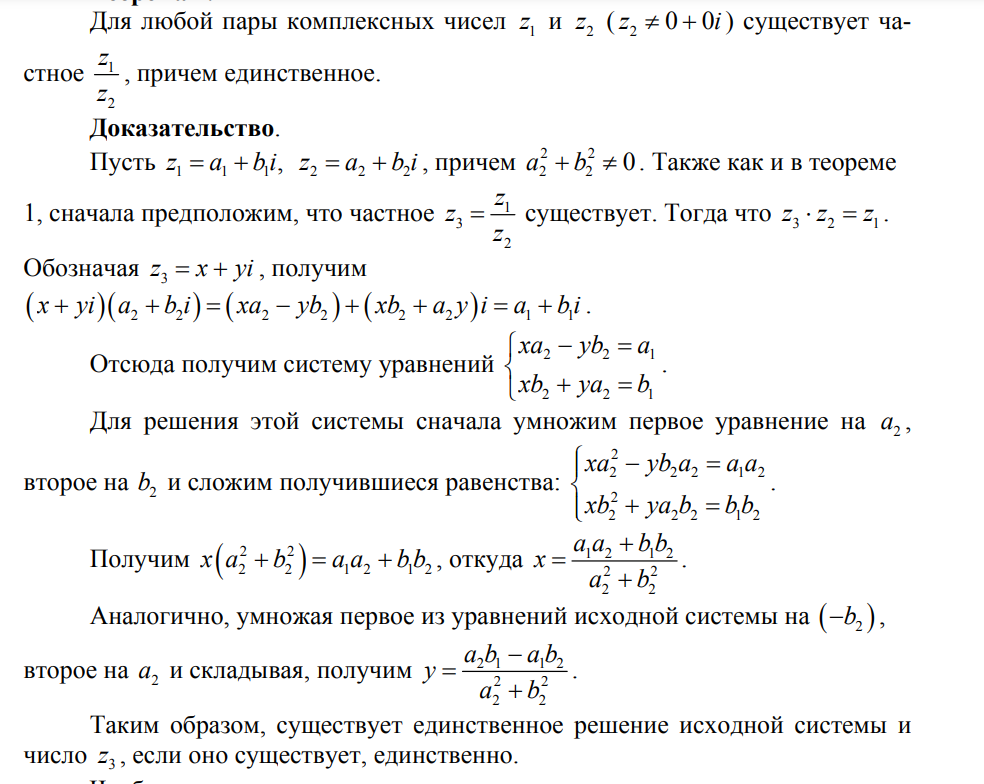
Если произведение двух комплексных чисел равно нулю, то, по крайней мере, один из сомножителей равен нулю.



**1.4. Частное комплексных чисел**



Теорема:



# Безусловная одномерная оптимизация

Оптимизация — это процесс нахождения точки экстремального значения некоторой заданной целевой функции f(x). Это один из крупнейших краеугольных камней прикладной математики, физики, инженерии, экономики, промышленности. Область её применений необъятна и может распространяться от минимизации физических величин на микро- и макроуровнях до максимизации прибыли или эффективности логистических цепочек. Машинное обучение также заострено на оптимизации: всевозможные регрессии и нейроные сети пытаются минимизировать ошибку между предсказанием и реальными данными.  
  
Экстремум может быть как минимумом, так и максимумом, но обычно принято изучать любую оптимизацию исключительно как поиск минимума, поскольку любая максимизация эквивалентна минимизации из-за возможности поменять знак перед целевой функцией: f(x) –> – f(x). Следовательно, в любом месте ниже под оптимизацией мы будем понимать именно минимизацию.

***Методы безусловной оптимизации.*** Методы безусловной оптимизации делятся на методы одномерной и многомерной оптимизации.

Методы одномерной оптимизации. Несмотря на то, что безусловная оптимизация функции одной переменной наиболее простой тип оптимизационных задач, она занимает центральное место в теории оптимизации как с теоретической, так и с практической точек зрения. Это связано с тем, что задачи однопараметрической оптимизации достаточно часто встречаются в инженерной практике и, кроме того, находят свое применение при реализации более сложных интерактивных процедур многопараметрической оптимизации.

Своеобразным индикатором важности методов оптимизации функции одной переменной является огромное множество реализованных алгоритмов, которые условно можно сгруппировать следующим образом:  методы исключения интервалов (метод половинного деления; метод "золотого" сечения; метод Фибоначчи); методы полиномиальной аппроксимации; методы с использованием производных[.](http://dl.sumdu.edu.ua/mo/rus/m_proizv.html)

Все методы одномерной оптимизации основаны на предположении, что исследуемая целевая функция в допустимой области по крайней мере обладает свойством унимодальности, так как для унимодальной функции W(x) сравнение значений W(t) в двух различных точках интервала поиска позволяет определить, в каком из заданных двумя указанными точками подинтервалов точки оптимума отсутствуют.

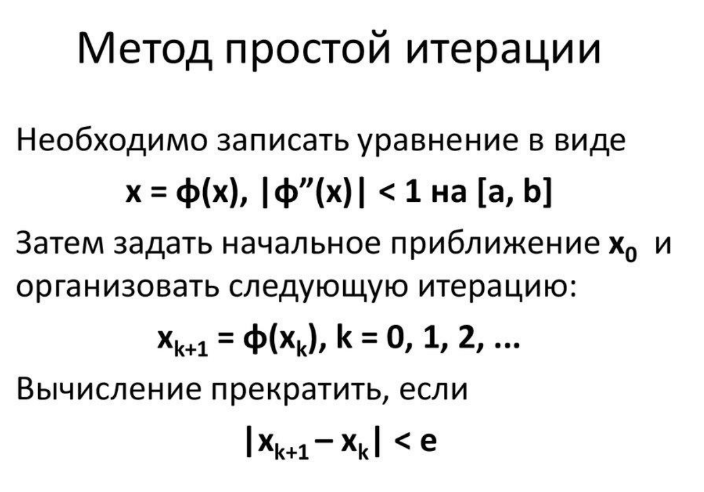
Методы поиска, которые позволяют определить оптимум функции одной переменной путем уменьшения интервала поиска, носят название методов исключения интервалов.

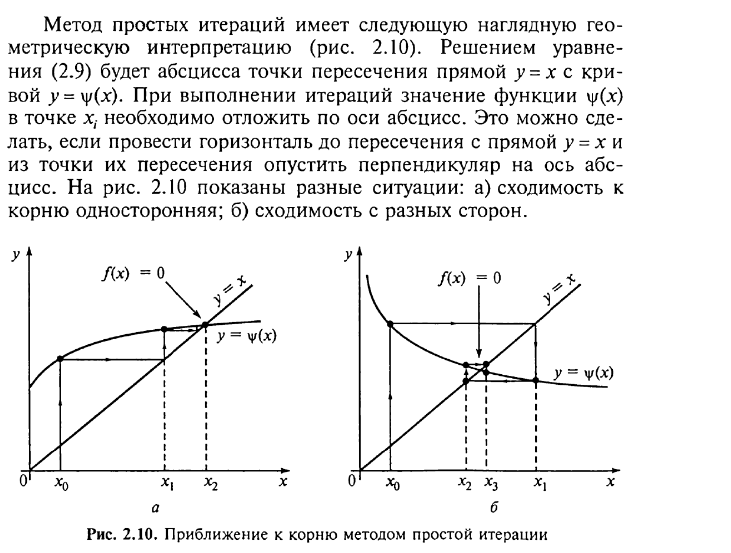
**1.1. Метод дихотомии(деление пополам)**

Метод дихотомии (также известный как метод деления отрезка пополам) - это численный метод для нахождения корня уравнения f(x)=0. Он основан на принципе "разделяй и властвуй", и заключается в последовательном делении отрезка [a, b] пополам до тех пор, пока не будет найден корень уравнения.

**1.2. Метод итераций(последовательного приближения)**

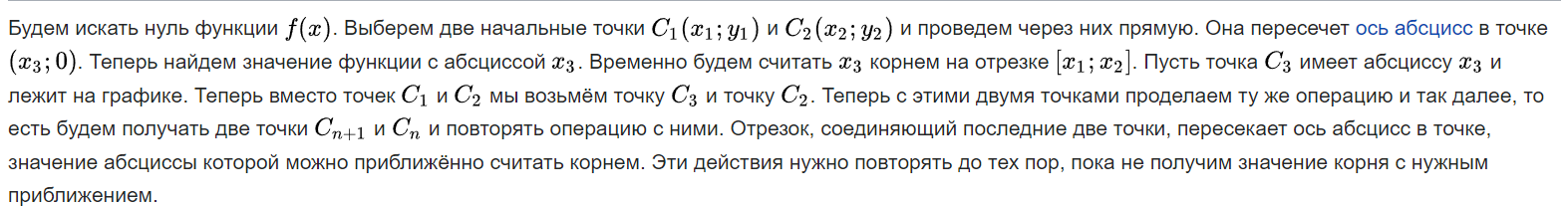
Метод итерации - это метод решения задач или задачей способом последовательных приближений. Он заключается в том, что задача разбивается на несколько шагов, при выполнении которых каждый следующий шаг основывается на предыдущем.



……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**1.3. Метод секущих(хорд)**

Метод секущих - это численный метод приближенного решения уравнения ( f(x) = 0 ), который основан на аппроксимации производной функции ( f(x) ) в точке методом конечной разности.Метод секущих имеет сходство с методом Ньютона, но отличается тем, что он не требует нахождения производной функции ( f(x) ), а использует приближенное значение наклона касательной к графику функции.

**1.4. Метод Ньютона(метод касательных)**

Метод касательных, также известный как метод Ньютона, является численным методом решения нелинейных уравнений. Он использует локальное приближение функции ее касательной, чтобы найти корень уравнения.

Метод касательных обычно сходится быстрее к корню уравнения по сравнению с методом половинного деления или методом простой итерации. Однако он требует наличия производной функции и может быть неустойчивым в некоторых случаях.

# Алгоритмы.

Алгоритм для дихотомии представлен ниже:

1) На каждом шаге процесса поиска делим отрезок [a,b] пополам, x=(a+b)/2 - координата середины отрезка [a,b].

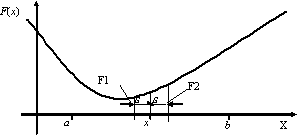
2) Вычисляем значение функции F(x) в окрестности +/-  вычисленной точки x: F1=F(x-), F­=F(x+).

3) Сравниваем F1 и F2 и отбрасываем одну из половинок отрезка [a,b]:

Если F1<F2, то отбрасываем отрезок [x,b], тогда b=x.

Иначе отбрасываем отрезок [a,x], тогда a=x.

4) Деление отрезка [a,b] продолжается, пока его длина не станет меньше заданной точности , т.е. |b-a|≤.



Результат работы алгоритма, представленного на блок-схеме: x – координата точки, в которой функция F(x) имеет минимум (или максимум), FM – значение функции F(x) в этой точке. Ниже представлена блок-схема алгоритма, который реализован в программе.

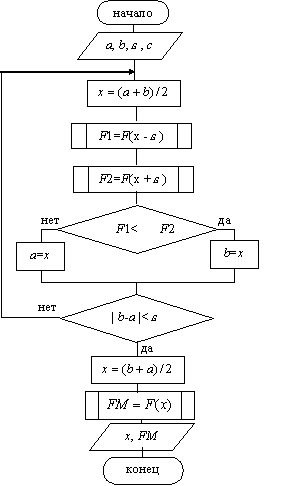


Рис. 2.1. Блок-схема алгоритма дихотомии.

Алгоритм для метода итераций(последовательного приближения):

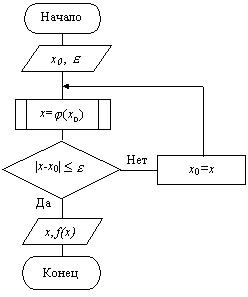
1. Задать начальное приближение ( x0) и выбрать требуемую точность \( \varepsilon \).

2. Вычислить следующее приближение ( x1) по формуле ( x1 = g(x0) ), где функция ( g(x) ) определяется из исходного уравнения как ( x = f(x) ).

3. Повторять шаг 2 до тех пор, пока условие ( | xn - x(n-1) | < \varepsilon \) не выполнится.

4. Вывести найденное приближенное решение ( x\_n ).

Примечание: Важно выбрать функцию ( g(x) ) таким образом, чтобы процесс итераций был сходящимся. Для этого можно использовать метод линеаризации исходного уравнения.

 Алгоритм метода секущих(хорд) ниже:

1. Выбираем две начальные точки x0 и x1, причем x0 < x1.

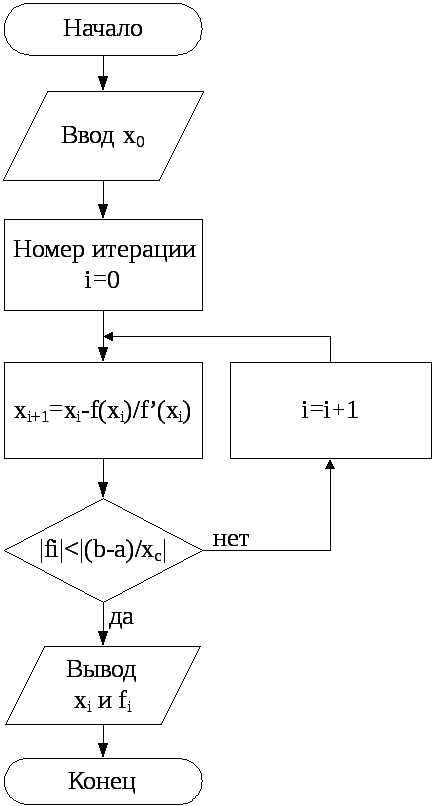
2. Вычисляем значения функции в этих точках: f0 = f(x0) и f1 = f(x1).

3. Находим следующее приближение x2 как пересечение прямой, проходящей через точки (x0, f0) и (x1, f1), с осью абсцисс.

4. Вычисляем значение функции в точке x2: f2 = f(x2).

5. Если значение f2 близко к нулю, то остановить и вернуть x2 как приближенное решение.

6. Иначе обновляем значения точек: x0 = x1, x1 = x2, f0 = f1, f1 = f2 и повторяем шаги с 3 по 5 до достижения необходимой точности или количества итераций.

Алгоритм метода Ньютона(метод касательных) ниже:

1. Выберите начальное приближение решения x0.

2. Вычислите значение функции f(x) и ее производной f'(x) в точке x0.

3. Вычислите приращение итерации по формуле: deltaX = f(x)/f'(x).

4. Обновите значение x0 по формуле: x1 = x0 - deltaX.

5. Повторяйте шаги 2-4 до тех пор, пока значение функции f(x) не станет достаточно близким к нулю или до достижения заданной точности.

6. Выведите найденное приближенное решение x1.



# Программа реализации алгоритмов (задание)

Используя язык программирования С++ и Python найти сумму, разность, произведение, частное двух комплексных чисел: 𝑧1 = 2 + 𝑖 , 𝑧2 = 2 − 2

#include <iostream>

#include <complex>

using namespace std;

int main() {

complex<double> z1(2.0, 1.0); //комплексное число, составленное из двуух элементов типа double (действительной и мнимой части)

complex<double> z2(2.0, -2.0);

cout << "z1 = 2 + i " << endl;

cout << "z2 = 2 - 2i " << endl;

complex<double> sum = z1 + z2;

cout << "Sum: " << sum << endl;

complex<double> diff = z1 - z2;

cout << "Difference: " << diff << endl;

complex<double> prod = z1 \* z2;

cout << "Product: " << prod << endl;

complex<double> div = z1 / z2;

cout << "Division: " << div << endl;

return 0;

}

PYTHON

from numpy import complex

z1 = complex(2.0, 1.0) # комплексное число, составленное из двух элементов типа double (действительной и мнимой части)

z2 = complex(2.0, -2.0)

print("z1 = 2 + i")

print("z2 = 2 - 2i")

sum = z1 + z2

print("Sum:", sum)

diff = z1 - z2

print("Difference:", diff)

prod = z1 \* z2

print("Product:", prod)

div = z1 / z2

print("Division:", div)

возвести в четвертую степень и вычислить корень третьей степени комплексного числа 𝑧3 = −1 + 2i

С++

#include <iostream>  
#include <complex>  
#include <cmath>  
  
using namespace std;  
  
int main() {  
 complex<double> z3(-1.0, 2.0);  
  
 complex<double> z4 = pow(z3, 4); // возводим в четвертую степень  
 cout << "z4 (z3 в 4 степени): " << z4 << endl;  
  
 complex<double> z5 = pow(z3, 1.0/3.0); // извлекаем корень третьей степени  
 cout << "z5 (корень из z3): " << z5 << endl;  
  
 return 0;  
}

PYTHON

import cmath

z3 = complex(-1.0, 2.0)

z4 = z3 \*\* 4 # возводим в четвертую степень

print("z4 (z3 в 4 степени):", z4)

z5 = z3 \*\* (1.0/3.0) # извлекаем корень третьей степени

print("z5 (корень из z3):", z5)

Используя язык программирования С++ и Python найти численно с помощью метода дихотомии 𝜀 = 10^−6 𝑥𝑒^(𝑥) − 𝑥 − 1 = 0

C++

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

double func(double x)

{

return pow(M\_E, x) \* x - x - 1;

}

int main()

{

int iter = 0;

double eps = 1e-6;

double a = -2.0, b = 2.0, t, f1, f2, x; //начальные значения интервала

do

{

f1 = func(a);

t = (a + b) / 2.0;

f2 = func(t);

if (f1 \* f2 <= 0)

b = t;

else

a = t;

iter++;

} while (fabs(b - a) > eps);

x = (a + b) / 2.0;

f1 = func(x);

if (fabs(f1) <= eps)

{

cout <<"\nThe root of the equation with error ";

cout << fixed << setprecision(16) <<"eps = "<< eps <<" is x = " << x; //задает количество знаков после запятой при выводе десятичных чисел

cout <<"\nFunction value f(x) = "<< func(x) <<".";

cout <<"\nNumber of iterations is "<< iter <<".";

}

else

{

cout <<"On this segment the equation has no roots!!!";

}

return 0;

}

PYTHON

import math

def func(x):

return math.exp(x) \* x - x - 1

def main():

iter = 0

eps = 1e-6

a = -2.0

b = 2.0

t = 0

f1 = func(a)

f2 = 0

x = 0

while abs(b - a) > eps:

f1 = func(a)

t = (a + b) / 2.0

f2 = func(t)

if f1 \* f2 <= 0:

b = t

else:

a = t

iter += 1

x = (a + b) / 2.0

f1 = func(x)

if abs(f1) <= eps:

print(f"\nThe root of the equation with error eps = {eps} is x = {x}")

print(f"Function value f(x) = {func(x)}")

print(f"Number of iterations is {iter}")

else:

print("On this segment the equation has no roots!!!")

if \_\_name\_\_ == "\_\_main\_\_": main()

Используя язык программирования С++ и Python найти численно с помощью метода итераций 𝜀 = 10^−6 𝑥𝑒^(𝑥) − 𝑥 − 1 = 0

C++

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

double func(double x)

{

return pow(M\_E, x) \* x - x - 1;

}

double find(double x, double eps)

{

double x\_1, x\_2;

int iter = 0;

x\_2 = x;

do {

x\_1 = x\_2;

x\_2 = 1 + x\_1 / pow(M\_E, x\_1);

iter++;

} while (fabs(x\_2 - x\_1) > eps && iter < 20);

cout << "Number of iterations is " << iter << endl;

return x\_2;

}

int main()

{

double x0 = 1.5, eps = 1e-6, root;

root = find(x0, eps);

cout << "Root x = " << fixed << setprecision(16) << root << endl;

cout << "Function value f(x) = " << func(root) << endl;

return 0;

}

PYTHON

import math

def func(x):

return math.exp(x) \* x - x - 1

def find(x, eps):

x\_1 = x

x\_2 = x

iter = 0

while True:

x\_1 = x\_2

x\_2 = 1 + x\_1 / math.exp(x\_1)

iter += 1

if abs(x\_2 - x\_1) <= eps or iter >= 20:

break

print(f"Number of iterations is {iter}")

return x\_2

def main():

x0 = 1.5

eps = 1e-6

root = find(x0, eps)

print(f"Root x = {root}")

print(f"Function value f(x) = {func(root)}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Используя язык программирования С++ и Python найти численно с помощью метода хорд 𝜀 = 10^−6 𝑥𝑒^(𝑥) − 𝑥 − 1 = 0

C++

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

double func(double x)

{

return pow(M\_E, x) \* x - x - 1;

}

double find(double x0, double x1, double eps)

{

int iter = 0;

double x\_next = 0;

double tmp;

do

{

tmp = x\_next;

x\_next = x1 - func(x1) \* (x0 - x1) / (func(x0) - func(x1));

x0 = x1;

x1 = tmp;

iter++;

} while (fabs(x\_next - x1) > eps && iter < 20000);

cout << "\nNumber of iterations is " << iter << ". " << endl;

return x\_next;

}

int main()

{

double x0, x1, x, eps;

cout << "a = ";

cin >> x0;

cout << "b = ";

cin >> x1;

eps = pow(10, -6);

x = find(x0, x1, eps);

cout << "Root x = " << fixed << setprecision(16) << x;

cout << "\nFunction value f(x)=" << func(x) << endl;

return 0;

}

PYTHON

import math

def func(x):

    return \* x - x - 1

def find(x0, x1, eps):

    iter = 0

    x\_next = 0

    tmp = 0

    while abs(x\_next - x1) > eps and iter < 20000:

        tmp = x\_next

        x\_next = x1 - func(x1) \* (x0 - x1) / (func(x0) - func(x1))

        x0 = x1

        x1 = tmp

        iter += 1

    print("\nNumber of iterations is", iter, ".")

    return x\_next

if \_\_name\_\_ == "\_\_main\_\_":

    x0 = float(input("a = "))

    x1 = float(input("b = "))

    eps = math.pow(10, -6)

    x = find(x0, x1, eps)

    print("Root x =", format(x, '.16f'))

    print("Function value f(x) =", func(x))

Используя язык программирования С++ и Python найти численно с помощью метода Ньютона 𝜀 = 10^−6 𝑥𝑒^(𝑥) − 𝑥 − 1 = 0

C++

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

double func(double x)

{

    return pow(M\_E, x) \* x - x - 1;//pow(M\_E, x) вычисляет значение числа e (основание натурального логарифма) в степени x

}

double dfunc(double x)

{

    return pow(M\_E, x) \* (x + 1);

}

double find(double x, double eps)

{

    double f, df;

    int iter = 0;

    cout << "x\_0 = " << x << " ";

    do {

        f = func(x);

        df = dfunc(x);

        x = x - f / df;

        iter++;

    } while (fabs(f) > eps && iter < 20000);

    cout << "\nNumber of iterations is " << iter << endl;

    return x;

}

int main()

{

    double x0 = 0.0, x, eps = 1e-6;

    x = find(x0, eps);

    cout << "Root x = " << fixed << setprecision(16) << x;

    cout << "\nFunction value f(x) = " << func(x) << endl;

    return 0;

}

PYTHON

import math

def func(x):

    return math.exp(x) \* x - x - 1

def dfunc(x):

    return math.exp(x) \* (x + 1)

def find(x, eps):

    iter = 0

    print("x\_0 = ", x, end=" ")

    while True:

        f = func(x)

        df = dfunc(x)

        x = x - f / df

        iter += 1

        if abs(f) <= eps or iter >= 20000:

            break

    print("\nNumber of iterations is", iter)

    return x

def main():

    x0 = 0.0

    eps = 1e-6

    x = find(x0, eps)

    print("Root x =", format(x, '.16f'))

    print("Function value f(x) =", func(x))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

# Заключение.

В данной работе были показаны такие алгоритмы поиска корней уравнения: с помощью методов дихотомии, итераций, хорд и Ньютона. Также были посчитаны сумма, разность, произведение, частное двух комплексных чисел и их возведение в степень, нахождение корня третьей степени комплексного числа.

# Литература

1. https://books.ifmo.ru/file/pdf/599.pdf
2. [http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_%D0%B7%D0%BE%D0%BB%D0%BE%D1%82%D0%BE%D0%B3%D0%BE\_%D1%81%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F](http://ru.wikipedia.org/wiki/Метод_золотого_сечения)
3. https://www.vsavm.by/knigi/kniga3/810.html