Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Курсовая работа

Дисциплина: «Вычислительная системы»

I семестр

Задание 4: « Итерационные процессы. Функции как параметры»

|  |  |
| --- | --- |
| Группа: | М8О-107Б-20, №25 |
| Студент: | Чекменев Вячеслав Алексеевич |
| Преподаватель: | Найденов Иван Евгеньевич |
| Оценка: |  |
| Дата: | 09.01.2021 |

Москва, 2021

## **Итерационные процессы. Функции как параметры**

### **Задание**

Составить программу на языке Си с процедурами решения трансцедентных алгебраических уравнений различными численными методам (итераций, Ньютона и половинного деление — дихотомии). Уравнения оформить как функции параметры, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений — заданного вариантом и следующего за ним. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию, например, с использованием gnuplot.

Вариант 28

Уравнение



Отрезок [1.2, 2]

Вариант 1

Уравнение



Отрезок [3, 4]

### **Краткие сведения из численных методов**

Рассматривается уравнение вида F(x)=0. Предполагается, что функций F(x) достаточно гладкая, монотонная на этом отрезке и существует единственный корень уравнения *x\**∈ [a,b]. На отрезке [a, b] ищется приближенное решение x с точностью ε, т. е. такое, что |x - x\**| < ε.*

В данном задании предлагается изучить и запрограммировать три простейших численных метода решения алгебраических уравнений и провести вычислительные эксперименты по определению корней уравнений на указанных в задании отрезках монотонности и, в качестве дополнительного упражнения, вне их.

### **Метод дихотомии (половинного деления)**

**Метод половинного деления** — простейший численный метод для решения нелинейных уравнений вида f(x)=0. Предполагается только непрерывность функции f(x). Для начала итераций необходимо знать отрезок [xL, xR] значений x, на концах которого функция принимает значения противоположных знаков. Это можно проверить так: Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения. Далее нужно найти значение xM середины отрезка Вычислим значение функции f(xM) в середине отрезка. Если значения функции в середине отрезка и на левой границе разные , то нужно переместить правую границу в середину отрезка, иначе левую границу в середину отрезка. Затем нужно повторить алгоритм начиная с вычисления значения xM  Алгоритм заканчивается тогда, когда f(xM)=0 либо xL=xR

### **Метод итераций**

**Метод итераций** — довольно простой численный метод решения уравнений. Метод основан на принципе сжимающего отображения, который применительно к численным методам в общем виде так же может называться методом простой итерации. Идея состоит в замене исходного уравнения f(x)=0 на эквивалентное ему x=φ(x). При чём должно выполнятся условие сходимости |φ(1)(x)|<0 на всём отрезке [a, b]. Итерации начинаются со значения xM середины отрезка. Однако φ(x) может выбрано неоднозначно. Сохраняет корни уравнения такое преобразование: Здесь λ0 – постоянная, которая не зависит от количества шагов. В данном случае мы возьмём , что приводит к простому методу одной касательной и имеет условие сходимости . Тогда итерационный процесс выглядит так: Условием окончания итераций является достижение нужной точности между предыдущим и следующим значением.

### **Метод Ньютона**

итерационный численный метод нахождения корня заданной функции, который является частным случаем метода итераций. А именно за λ0 берётся значение производной в каждой новой точке. Тогда итерационный процесс имеет вид Условие окончания итераций и начальное значение абсолютно такие же, как и в методе итерации. Условие сходимости метода можно записать как

**Решение**

Подключим две нужные нам библиотеки:

*#include <math.h>*

*#include <stdio.h>*

напишем функцию по вычислению машинного эпсилон:

*double epsilon()*

*{*

*double eps = 1.0;*

*while ((1.0 + eps / 2.0) > 1.0) {*

*eps = eps / 2.0;*

*}*

*return eps;*

*}*

**1.Метод Ньютона.**

Напишем нужные функции.

первая функция:

*double func1(double x) {*

*return x - 2 + sin(1 / x);*

*}*

производная первой функции:

*double der\_f1(double x) {*

*return 1 - cos(1 / x) / (x \* x);*

*}*

вторая функция:

*double func2(double x) {*

*return exp(x) + log(x) - 10 \* x;*

*}*

производная второй функции:

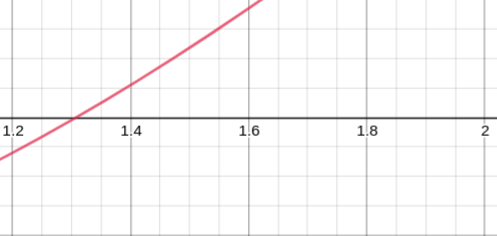
*double der\_f2(double x) {*

*return exp(x) + 1 / x - 10;*

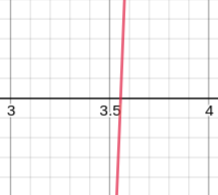
*}*

построим графики функций в desmos.

Func1:



Func2:



у всех функций обе производные положительны, поэтому можно за начальную точку для x взять b:

*double Newton(double f(double), double d\_f(double), double a, double b, double eps)*

*{*

*double x = b;*

*while (fabs(f(x) / d\_f(x)) >= eps) {*

*x -= f(x) / d\_f(x);*

*}*

*return x;*

*}*

**2.Решение с помощью метода дихотомии.**

За начальную точку возьмем c = (a + b) / 2

будем делить отрезок на два, и брать ту часть, на которой корень уравнения

*double Dichotomy(double f(double), double a, double b, double eps)*

*{*

*double c = 0;*

*while (f(c) != 0 && fabs(b - a) > eps) {*

*c = (a + b) / 2;*

*(f(c) \* f(a) > 0) ? (a = c) : (b = c);*

*}*

*return c;*

*}*

**3.Метод итераций.**

Возьмем за настоящее значение точки curr = (a + b) / 2, а за предыдущее prev = curr + 1

будем изменять prev и curr точки, пока fabs(curr - prev) > eps

использовать будем новые функции, полученные путем перенесения x в другую часть уравнения.

Func1:

double func1\_used(double x) {

return 2 - sin(1 / x);

}

func2:

*double func2\_used(double x) {*

*return exp(x) / 10 + log(x) / 10;*

*}*

полная функция:

*double Iteration(double f(double), double a, double b, double eps)*

*{*

*double curr = (a + b) / 2, prev = curr + 1;*

*while(fabs(curr - prev) > eps) {*

*prev = curr;*

*curr = f(prev);*

*}*

*return curr;*

*}*

**полная программа:**

*#include <math.h>*

*#include <stdio.h>*

*double epsilon()*

*{*

*double eps = 1.0;*

*while ((1.0 + eps / 2.0) > 1.0) {*

*eps = eps / 2.0;*

*}*

*return eps;*

*}*

*double func1(double x) {*

*return x - 2 + sin(1 / x);*

*}*

*double der\_f1(double x) {*

*return 1 - cos(1 / x) / (x \* x);*

*}*

*double func1\_used(double x) {*

*return 2 - sin(1 / x);*

*}*

*double func2(double x) {*

*return exp(x) + log(x) - 10 \* x;*

*}*

*double der\_f2(double x) {*

*return exp(x) + 1 / x - 10;*

*}*

*double func2\_used(double x) {*

*return exp(x) / 10 + log(x) / 10;*

*}*

*double Newton(double f(double), double d\_f(double), double a, double b, double eps)*

*{*

*double x = b;*

*while (fabs(f(x) / d\_f(x)) >= eps) {*

*x -= f(x) / d\_f(x);*

*}*

*return x;*

*}*

*double Dichotomy(double f(double), double a, double b, double eps)*

*{*

*double c = 0;*

*while (f(c) != 0 && fabs(b - a) > eps) {*

*c = (a + b) / 2;*

*(f(c) \* f(a) > 0) ? (a = c) : (b = c);*

*}*

*return c;*

*}*

*double Iteration(double f(double), double a, double b, double eps)*

*{*

*double curr = (a + b) / 2, prev = curr + 1;*

*while(fabs(curr - prev) > eps) {*

*prev = curr;*

*curr = f(prev);*

*}*

*return curr;*

*}*

*int main(void)*

*{*

*double eps = epsilon();*

*printf("function: x - 2 + sin(1 / x)\n");*

*printf("Newton Method value: %13f\n", Newton(func1, der\_f1, 1.2, 2, eps));*

*printf("Dichotomy Method value: %10f\n", Dichotomy(func1, 1.2, 2, eps));*

*printf("Iteration Method value: %10f\n\n", Iteration(func1\_used, 1.2, 2, eps));*

*printf("function: exp(x) + log(x) - 10 \* x\n");*

*printf("Newton Method value: %13f\n", Newton(func2, der\_f2, 3, 4, eps));*

*printf("Dichotomy Method value: %10f\n", Dichotomy(func2, 3, 4, eps));*

*}*

Запустим ее и посмотрим результат ее выполнения:

*function: x - 2 + sin(1 / x)*

*Newton Method value: 1.307663*

*Dichotomy Method value: 1.307663*

*Iteration Method value: 1.307663*

*function: exp(x) + log(x) - 10 \* x*

*Newton Method value: 3.526498*

*Dichotomy Method value: 3.526498*

Мы видим,что вычисление по методу дихотомии и по методу Ньютона,дают ответ,совпадающий с тем,что дан нам в задании, однако метод итераций не работает для второго уравнения.

Пояснение ошибки или почему не получилось решить уравнение 2 методом простых итераций:

запишем производную от функции x = *func2\_used*(x):

1 = (1 + x \* exp(x)) / (10 \* x)

построим график правой части:

 3

2

1

видим, что f\_deriv(x) > 1 => метод не сходится, выдает ошибку.

**Вывод**

В работе описаны идеи и принципы трёх численных методов: дихотомии, итераций и Ньютона. Проверены условия сходимости данных уравнений методам и проведены нужные вычисления для использования методов. Составлен алгоритм решения уравнений, на основе которого составлена программа на языке Си. Описан формат ввода и вывода, проведено тестирование программы, составлен протокол исполнения программы.