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

$$x - 2 + \sin\frac{1}{x} = 0$$

Отрезок [1.2, 2]

Вариант 1
Уравнение

$$e^x + \ln x - 10x = 0$$

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 $x^* \in [a,b]$. На отрезке [a, b] ищется приближенное решение x с точностью ε, т. е. такое, что $|x - x^*| < \varepsilon$.

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 [x_L, x_R] значений x, на концах которого функция принимает значения противоположных знаков. Это можно проверить так: $f(x_L) * f(x_R) < 0$Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения. Далее нужно найти значение x_M середины отрезка $x_M = \frac{x_L + x_R}{2}$Вычислим значение функции $f(x_M)$ в середине отрезка. Если значения функции в середине отрезка и на левой границе разные $f(x_M) * f(x_L) < 0$, то нужно переместить правую границу в середину отрезка, иначе левую границу в середину отрезка. Затем нужно повторить алгоритм начиная с вычисления значения x_M Алгоритм заканчивается тогда, когда $f(x_M)=0$ либо $x_L=x_R$

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 $x=\varphi(x)$. При чём должно выполнятся условие сходимости $|\varphi^{(1)}(x)|<0$ на всём отрезке [a, b]. Итерации начинаются со значения x_M середины отрезка. Однако $\varphi(x)$ может выбрано неоднозначно. Сохраняет корни уравнения такое преобразование: $\varphi(x) = x - \lambda_0 * f(x)$Здесь λ_0 – постоянная, которая не зависит от количества шагов. В данном случае мы возьмём $\lambda_0 = \frac{1}{f^{(1)}(x_M)}$, что приводит к простому методу одной касательной и имеет условие сходимости $\lambda_0 * f^{(1)}(x) > 0$. Тогда итерационный процесс выглядит так: $x_{k+1} = x_k - \lambda_0 * f(x_k)$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 λ_0 берётся значение производной в каждой новой точке. Тогда итерационный процесс имеет вид $x_{k+1} = x_k - \frac{f(x_k)}{f^{(1)}(x_k)}$Условие окончания итераций и начальное значение абсолютно такие же, как и в методе итерации. Условие сходимости метода можно записать как $|f(x) * f^{(2)}(x)| < (f^{(1)}(x))^2$

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;
}
```

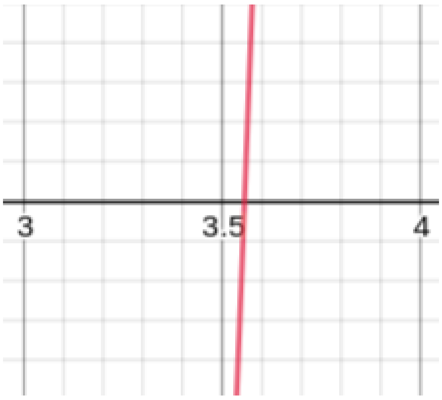
производная второй функции:

```
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
```

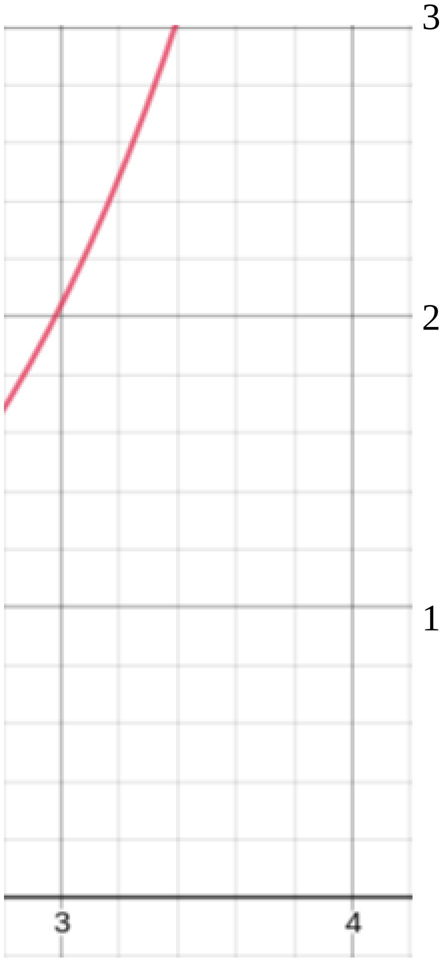
Dichotomy Method value: 3.526498

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:



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.