МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Вычислительные системы»
I семестр
Задание 4
«Процедуры и функции в качестве параметров»

Группа:	М8О-107Б-20
Студент:	Алапанова Эльза Халилевна
Преподаватель:	Зайцев Валентин Евгеньевич
Оценка:	
Дата:	

Москва, 2021

Содержание

Постановка задачи 3
Теоретическая часть 4
Метод половинного деления 4
Метод итераций 4
Метод Ньютона 4
Проверка условий сходимости 5
Описание алгоритма 7
Описание программы 8
Использованные в программе структуры данных 8
Использованные в программе переменные 8
Использованные в программе функции 8
Программа 10
Входные и выходные данные 13
Входные данные 13
Выходные данные 13
Протокол исполнения и тесты 14
Тест #1 14
Тест #2 14
Тест #3 14
Тест #4 14
Вывод 16
Список литературы 17

Постановка задачи

Составить программы на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления — дихотомия). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию, например, с использованием *gnuplot*.

Вариант 13
Уравнение $x * tgx - \frac{1}{3} = 0$
Отрезок [0.2, 1]

Теоретическая часть

Метод 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=φ(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 φ(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

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 $\left|f(x) * f^{(2)}(x)\right| < \left(f^{(1)}(x)\right)^2$.

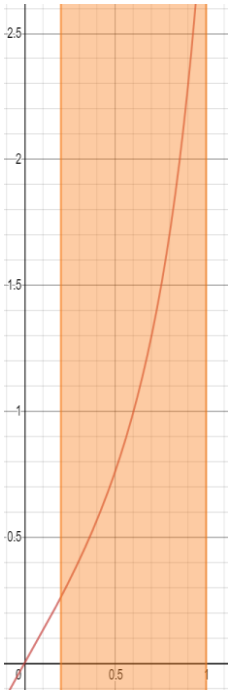
Проверка условий сходимости

Пусть уравнение $f_1(x)$. Функция непрерывна на заданном промежутке, значит метод дихотомии применим к ней. Найдём производную $f_1^{(1)}(x)$ для заданной функций:

$$f_1^{(1)}(x) = tgx + \frac{x}{cos^2(x)}$$

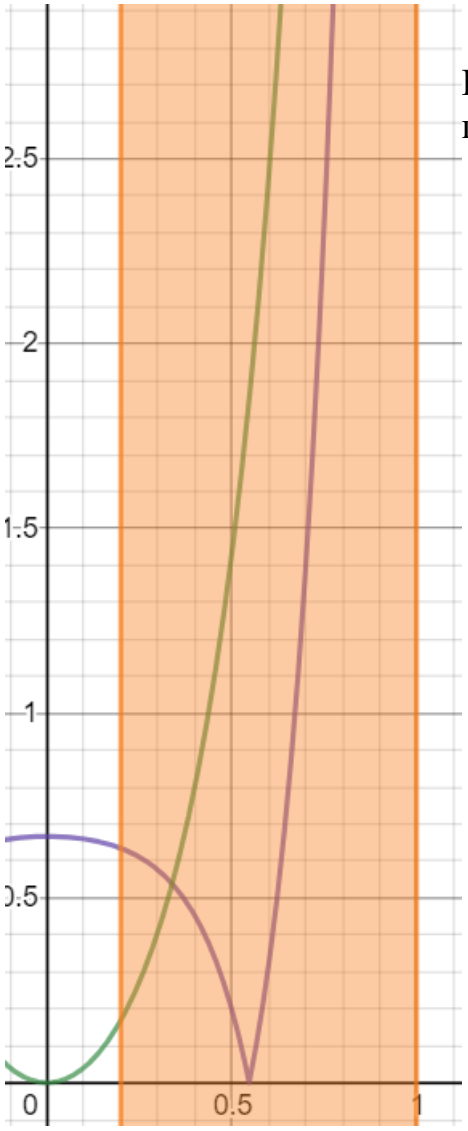
$\lambda_1 = f1^{(1)}(x_M) = 1.564962...$

Проверим условия сходимости для метода итераций, построив графики производных сжимающих отображений:



$$y = \lambda_1 * f_1^{(1)}(x)$$

Уравнение удовлетворяет условию сходимости метода итераций.



Проверим условия сходимости для метода Ньютона, построив графики левых и правых частей неравенства.

$$y_1 = \left| f_1(x) * f_1^{(2)}(x) \right|$$

$$y2 = \left(f_1^{(1)}(x) \right)^2$$

Уравнение удовлетворяет условию сходимости метода Ньютона.

Описание алгоритма

Рассмотрим алгоритм решения. Сперва нужно найти машинное эпсилон, на котором будет основываться точность вычисления. Это можно сделать просто деля 1 на 2 за $O(\log(10^{16}))\sim O(1)$. Опишем каждую из функций решения уравнений разными методами. Для метода дихотомии достаточно просто выполнять итерации, пока $x_R-x_L>\varepsilon$. Корень с помощью такого метода рассчитывается примерно за $O(\log_2 10^k)\sim O(k*\log_2 10)$. Все функции, их производные и сжимающие отображения зададим в виде функций-параметров. Алгоритмы для решения уравнений методом итерации и Ньютона реализованы так же, как и было описано в теории. Невозможно оценить из сложность, так как она зависит от самой функции. Будем сохранять все корни и сразу же из выводить, не затрачивая память на новые переменные.

Описание программы

Использованные в программе структуры данных

Название структуры	Переменные в структуре	Смысл структуры
root_x	int steps double x	steps – то самое N, число итераций, затраченное на вычисление корня. x – то самое x_0, искомое значение корня

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
k	int	То самое число K, используемое для вычисления точности. Так же обозначает, что вывод должен быть с точность до K знаков после запятой
e0	double	То самое машинное эпсилон. В случае с double $\varepsilon = 2.20 * 10^{-16}$
acc	double	Точность вычислений. Именно с этой переменной мы будем сравнивать ответы A_1 и A_2
x0	root	Значение корня, которое будут возвращать функции после вычисления.

Использованные в программе функции

Название функции	Тип возвращаемой переменной	Смысл функции
square	double	Возвращает квадрат числа
do_nothing	double	Копирует значение в память, где double выделяется 64 бита, а не 80 бит
solve_binary_search	root	Решение уравнения методом половинного деления. Принимает f – функцию-параметр f(x), l – x_L, r – x_R , k и acc

solve_iteration	root	Решение уравнения методом итерации. Принимает f – функцию-параметр φ(x), x0 – x_M, k и acc
solve_newton	root	Решение уравнения методом Ньютона. Принимает f – функцию-параметр f(x), derivative_f – функцию-параметр $f^{(1)}(x)$, x0 – x_M, k и acc
f13	double	То самое $f_1(x)$
squeeze_f13	double	То самое $\varphi_1(x)$
d_dx_f13	double	То самое $f_1^{(1)}(x)$

Программа

```
#include <math.h>
#include <stdio.h>

typedef struct root_x root;

struct root_x{
	double x;
	int steps;
};

double square(double x){
	return x*x;
}

double do_nothing(double x){
	return x;
}

root solve_binary_search(double f(double), double l, double r, int k, double acc){
	int step=0;
	double m;
	while(r-l>acc){
		step++;
		m=(l+r)/2.0;

		if(f(m)*f(l)<0){
			r=m;
		}else{
			l=m;
		}
	}
	root ans={m, step};
	return ans;
}

root solve_iteration(double f(double), double x0, int k, double acc){
	int step=0;
	double cur=x0;
```

```
	double prev=cur+1;
	while(fabs(cur-prev)>acc){
		prev=cur;
		cur=f(prev);
		step++;
	}
	root ans={cur, step};
	return ans;
}

root solve_newton(double f(double), double derivative_f(double), double x0, int k,
double acc){
	int step=0;
	double cur=x0;
	double prev=cur+1;
	while(fabs(cur-prev)>acc){
		prev=cur;
		cur=prev-f(prev)/derivative_f(prev);
		step++;
	}
	root ans={cur, step};
	return ans;
}

double f13(double x){
	return x*tan(x)-1.0/3.0;
}

double squeeze_f13(double x){
	return x-f13(x)/1.564962;
}

double d_dx_f13(double x){
	return tan(x)+x/square(cos(x));
}

int main(){
	int k;
	scanf("%d", &k);
	double e0=1.0;
```

```
	while(do_nothing(1.0+e0/2.0)>1.0){
		e0=e0/2.0;
	}
	printf("Machine epsilon equals %.8e\n", e0);
	printf("-------------------------------------\n");
	double acc=e0*pow(10, 16-k);
	root x0;

	printf("Answer for x*tg(x)-1/3=0\n");
	x0=solve_binary_search(f13, 0.2, 1.0, k, acc);
	printf("%.*f | %d\n", k, x0.x, x0.steps);

	x0=solve_iteration(squeeze_f13, (0.2+1.0)/2.0, k, acc);
	printf("%.*f | %d\n", k, x0.x, x0.steps);

	x0=solve_newton(f13, d_dx_f13, (0.2+1.0)/2.0, k, acc);
	printf("%.*f | %d\n", k, x0.x, x0.steps);

	return 0;
}
```

Входные и выходные данные

Входные данные

Единственная строка содержит одно целое число K ($0 \le K \le 16$) — коэффициент для точности вычисления искомого корня.

Выходные данные

Программа должна вывести значение машинного эпсилон, а затем 6 строк. В каждой строке необходимо вывести искомое значение корня x_0 с точность K знаков после запятой и N — число итераций. В первых трёх строках для первого уравнения, а в следующих трёх для другого. Следует выводить сначала значения, полученные методом дихотомии, потом методом итерации, затем методом Ньютона.

Протокол исполнения и тесты

Тест #1

Ввод:
3
Вывод:
Machine epsilon equals 2.22044605e-016

Answer for x*tg(x)-1/3=0
0.548 | 9
0.547 | 3
0.547 | 3

Тест #2

Ввод:
6
Вывод:
Machine epsilon equals 2.22044605e-016

Answer for x*tg(x)-1/3=0
0.547160 | 19
0.547161 | 6
0.547161 | 4

Тест #3

Ввод:
10
Вывод:
Machine epsilon equals 2.22044605e-016

Answer for x*tg(x)-1/3=0
0.5471607571 | 32
0.5471607573 | 11
0.5471607573 | 5

Тест #4

Ввод:
16
Вывод:
Machine epsilon equals 2.22044605e-016

Answer for x*tg(x)-1/3=0

0.5471607572603301 | 52
0.5471607572603301 | 17
0.5471607572603300 | 5

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.

Список литературы

1. Метод бисекции [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_бисекции
2. Метод Ньютона [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_Ньютона
3. Метод простой итерации [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_простой_итерации