Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №3**

з дисципліни: «Інформатика 2»

|  |  |
| --- | --- |
|  | Виконав : Першко Федір Сергійович  Група: РЕ-11  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

**Мета роботи**: Створити програму для знаходження кореня нелінійного рівняння трьома методами, при цьому зобразивши ітераційний процес його пошуку.

**Ключові моменти**:

1. Задання початкових даних користувачем:

do{

printf("\n\tEnter the left boundary of calculating (x1 < x2)\n X(first)=");

scanf("%lf", &x1);

printf("\n\tEnter the right boundary of calculating\n X(last)=");

scanf("%lf", &x2);

}while (x1 >= x2);

if ( f(x1) \* f(x2) > 0){

printf("\n\tIf such boundaries are set, the program cannot cope with the search for roots.");

printf("\n\tWhat would you like to do next (press 1 to select 1. or any other key to select 2.)");

printf("1.change search boundaries\n2.let the program to shorten the interval");

scanf("%u", &var1);

if (var1 == 1){

do{

printf("\n\tEnter the left boundary of calculating (x1 < x2)\n X(first)=");

scanf("%lf", &x1);

printf("\n\tEnter the right boundary of calculating\n X(last)=");

scanf("%lf", &x2);

}while (x1 >= x2);

}

else {

printf("\n\tEnter the value of step of shortening the root search interval");

scanf("%lf", &h);

while ((x1 \* x2) < 0){

x1 += h;

}

}

}

do{

printf("\tEnter the maximum number of interations (N>0)\nN=");

scanf("%u", &N);

}while(N <= 0);

n = N;

printf("\n\tEnter the measurment error of calculating\n Measurment error=");

scanf("%lf", &measurement\_error);

1. Вибір способу вираховування:

do

{

printf("\nChoose the method of calculating:\n");

printf("\t1. By Chords method:\n");

printf("\t2. By Half division method:\n");

printf("\t3. By Newton's method (tangent method):\n");

scanf("%u", &var2);

if (var2!=1 && var2!=2 && var2!=3)

printf("\nYou are mistaken\n");

}while (var2!=1 && var2!=2 && var2!=3);

1. «Засікання часу»:

s\_time1 = time (NULL);

1. Реалізація рахуючої функції (на прикладі методу хорд):

double chord\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, unsigned int var3, unsigned int debug){

shapka ();

if (var3 == 2) { //Користувач вирішив не призупиняти роботу програми (див. рядки 109 - 161)

do {

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

xi = (f(x2)\*x1-f(x1)\*x2) / ( f(x2)-f(x1) );

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d |%.8f |%.8f |", i, xi, f(xi));

i ++;

} while (fabs(f(xi)) > measurement\_error);

}

else {

for (i; i < N+1; i++){

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

xi = (f(x2)\*x1-f(x1)\*x2) / ( f(x2)-f(x1) );

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d |%.8f |%.8f |", i, xi, f(xi));

if( fabs(f(xi)) <= measurement\_error ){

break;

}

}

}

return xi;

}

1. Перевірка чи вкладається результат у похибку, та визначення часу, витраченого на знаходження кореня:

if (fabs (f(xi)) <= measurement\_error){

s\_time2 = time (NULL);

all\_time = (s\_time2 - s\_time1);

}

1. Вибір подальшого ходу програми користувачем, якщо результат не вкладається у похибку:

do{

printf("\n\n\tThe number of iterations you specified has been exhausted.\n\tWhat would you like to do next:");

printf("\n\n1. perform the same number of iterations \n2. Perform iterations until the result \n3. Exit the program\n");

scanf("%u", &var3);

if (var3!=1 && var3!=2 && var3!=3)

printf("\nYou are mistaken\n");

}while (var3!=1 && var3!=2 && var3!=3);

**Код**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

double f(double x)

{

return 4\*pow(x+1, 3)-3\*x-3;

}

double chord\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, unsigned int var3, unsigned int debug);

double half\_division\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, unsigned int var3, unsigned int debug);

double Newton\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, double delta, unsigned int var3, unsigned int debug);

void shapka(void);

int main()

{

double x1, x2, xi=0, measurement\_error, h, delta;

unsigned int N, n, var1, var2, var3=0, s\_time1, s\_time2, all\_time, i=1, debug=0;

printf("\n\t\*==================================\*");

printf("\n\t| lab3 |");

printf("\n\t\*----------------------------------\*");

printf("\n\t| nonlinear equation |");

printf("\n\t\*==================================\*\n");

do{

printf("\n\tEnter the left boundary of calculating (x1 < x2)\n X(first)=");

scanf("%lf", &x1);

printf("\n\tEnter the right boundary of calculating\n X(last)=");

scanf("%lf", &x2);

}while (x1 >= x2);

if ( f(x1) \* f(x2) > 0){

printf("\n\tIf such boundaries are set, the program cannot cope with the search for roots.");

printf("\n\tWhat would you like to do next (press 1 to select 1. or any other key to select 2.)");

printf("1.change search boundaries\n2.let the program to shorten the interval");

scanf("%u", &var1);

if (var1 == 1){

do{

printf("\n\tEnter the left boundary of calculating (x1 < x2)\n X(first)=");

scanf("%lf", &x1);

printf("\n\tEnter the right boundary of calculating\n X(last)=");

scanf("%lf", &x2);

}while (x1 >= x2);

}

else {

printf("\n\tEnter the value of step of shortening the root search interval");

scanf("%lf", &h);

while ((x1 \* x2) < 0){

x1 += h;

}

}

}

do{

printf("\tEnter the maximum number of interations (N>0)\nN=");

scanf("%u", &N);

}while(N <= 0);

n = N;

printf("\n\tEnter the measurment error of calculating\n Measurment error=");

scanf("%lf", &measurement\_error);

do

{

printf("\nChoose the method of calculating:\n");

printf("\t1. By Chords method:\n");

printf("\t2. By Half division method:\n");

printf("\t3. By Newton's method (tangent method):\n");

scanf("%u", &var2);

if (var2!=1 && var2!=2 && var2!=3)

printf("\nYou are mistaken\n");

}while (var2!=1 && var2!=2 && var2!=3);

system ("cls");

switch(var2)

{

case 1:

{

printf("\n\n\t======\*Chords method\*======\n");

s\_time1 = time (NULL);

xi = chord\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

} break;

case 2:

{

printf("\n\n\t======\*Half division method\*======\n");

s\_time1 = time (NULL);

xi = half\_division\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

} break;

case 3:

{

printf("\n\n\t======\*Newton's method (tangent method)\*======\n");

do{

printf("\n\tEnter delta x (0 < delta x < 1)\n");

scanf("%lf", &delta);

}while (delta <= 0 || delta >= 1);

s\_time1 = time (NULL);

xi = Newton\_method(x1, x2, xi, N, i, measurement\_error, delta, var3, debug);

} break;

}

shapka();

if (fabs (f(xi)) <= measurement\_error){

s\_time2 = time (NULL);

all\_time = (s\_time2 - s\_time1);

}

else{

do{

s\_time2 = time (NULL);

all\_time += (s\_time2 - s\_time1);

do{

printf("\n\n\tThe number of iterations you specified has been exhausted.\n\tWhat would you like to do next:");

printf("\n\n1. perform the same number of iterations \n2. Perform iterations until the result \n3. Exit the program\n");

scanf("%u", &var3);

if (var3!=1 && var3!=2 && var3!=3)

printf("\nYou are mistaken\n");

}while (var3!=1 && var3!=2 && var3!=3);

system ("cls");

s\_time1 = time (NULL);

switch(var3){

case 1:

{

printf("\n\n\t======\*%d intervals more\*======\n", n);

i += n;

N += n;

debug = 1;

s\_time1 = time (NULL);

if (var2 == 1) {

xi = chord\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

}

else if (var2 == 2) {

xi = half\_division\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

}

else {

xi = Newton\_method(x1, x2, xi, N, i, measurement\_error, delta, var3, debug);

}

} break;

case 2:

{

printf("\n\n\t======\*Until the end\*======\n");

debug = 1;

s\_time1 = time (NULL);

if (var2 == 1) {

xi = chord\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

}

else if (var2 == 2) {

xi = half\_division\_method(x1, x2, xi, N, i, measurement\_error, var3, debug);

}

else {

xi = Newton\_method(x1, x2, xi, N, i, measurement\_error, delta, var3, debug);

}

shapka();

s\_time2 = time (NULL);

all\_time += (s\_time2 - s\_time1);

printf("\n\n\t\tIt took %d seconds to complete\n", all\_time);

printf("\t\tFinal answer:\n\t\tX = %.8f \n", xi);

return 0;

} break;

case 3:

{

printf("\n\n\t\tIt took %d seconds to complete\n", all\_time);

return 0;

}

}

shapka();

}while (fabs(f(xi)) > measurement\_error);

}

printf("\n\n\t\tIt took %d seconds to complete\n", all\_time);

printf("\t\tFinal answer:\n\t\tX = %.8f \n", xi);

return 0;

}

//---------------------------------------------------------------------------------------------------------------------------------------------

double chord\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, unsigned int var3, unsigned int debug){

shapka ();

if (var3 == 2) { //Користувач вирішив не призупиняти роботу програми (див. рядки 109 - 161)

do {

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

xi = (f(x2)\*x1-f(x1)\*x2) / ( f(x2)-f(x1) );

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d |%.8f |%.8f |", i, xi, f(xi));

i ++;

} while (fabs(f(xi)) > measurement\_error);

}

else {

for (i; i < N+1; i++){

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

xi = (f(x2)\*x1-f(x1)\*x2) / ( f(x2)-f(x1) );

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d |%.8f |%.8f |", i, xi, f(xi));

if( fabs(f(xi)) <= measurement\_error ){

break;

}

}

}

return xi;

}

//----------------------------------------------------------------------------------------------------------------------------------------------

double half\_division\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, unsigned int var3, unsigned int debug){

shapka();

if (var3 == 2) { //Користувач вирішив не призупиняти роботу програми (див. рядки 109 - 161)

do{

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

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

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d |%.8f |%.8f |", i, xi, f(xi));

i ++;

} while (fabs(f(xi)) > measurement\_error);

}

else{

for (i; i <= N; i++){

if (debug == 0){

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

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d | %.8f | %.8f |", i, xi, f(xi));

if( fabs(f(xi)) <= measurement\_error ){

break;

}

}

}

return (xi);

}

//------------------------------------------------------------------------------------------------------------------------------------------------

double Newton\_method(double x1, double x2, double xi, unsigned int N, unsigned int i, double measurement\_error, double delta, unsigned int var3, unsigned int debug){

double derivative;

derivative = ( f(x1+delta) - f(x1) ) / delta;

shapka();

if (var3 == 2) { //Користувач вирішив не призупиняти роботу програми (див. рядки 109 - 161)

do{

if (debug == 0){ // debug може приймати значення 1 (див. рядки 152 і 171)

xi = x1 - (f(x1) / derivative);

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d | %.8f | %.8f |", i, xi, f(xi));

i ++;

} while (fabs(f(xi)) > measurement\_error);

}

else{

for (i=1; i <= N; i++){

if (debug == 0){

xi = x1 - (f(x1) / derivative);

}

if ( f(xi)\*f(x1) > 0 ){

x1 = xi;

}

else{

x2 = xi;

}

debug = 0;

printf("\n\t| %d | %.8f | %.8f |", i, xi, f(xi));

if (fabs(f(xi)) <= measurement\_error) {

break;

}

}

}

return xi;

}

void shapka(){

printf ("\n\n\t================================================");

printf ("\n\t| N | X | F(X) |");

printf ("\n\t================================================\n");

}

**Висновок:**

Мені вдалося створити програму для знаходження кореня нелінійного рівняння трьома методами, при цьому зобразивши ітераційний процес його пошуку.