Для интервального расширения функции и заданного N построить таблицу значений интервальной ф-ии на заданном отрезке в N точках, включая концевые, с заданным радиусом R интервалов и шагом

Параметры a, b, N, R запрашиваются программой.

package main

import (

"fmt"

"math"

)

func f(x float64) float64 {

return 3 + 4\*x - math.Tanh(x-5)

}

func df(x float64) float64 {

return 4 - (1 / math.Pow(math.Cosh(x-5), 2))

}

func signChange(x1, x2 float64) bool {

return df(x1)\*df(x2) <= 0

}

func bisection(x1, x2 float64, maxIterations, epsilon float64) float64 {

if !signChange(x1, x2) {

// fmt.Println("Не удовлетворяет условиям метода бисекции.")

return math.NaN()

}

var mid float64

for i := 0.0; i < maxIterations; i++ {

mid = (x1 + x2) / 2.0

if math.Abs(df(mid)) < epsilon {

return mid

}

if signChange(x1, mid) {

x2 = mid

} else {

x1 = mid

}

}

return mid

}

func solution(x1, x2 float64, N int, R, maxIterations, epsilon float64) {

h := (x2 - x1) / float64(N-1)

count := 0

fmt.Println("\nk\tx1\tx2\ty1\ty2\twid(Y)")

for c := x1; c <= x2; c += h {

var points []float64

X1 := c - R

X2 := c + R

H := 0.01

for i := X1; i < X2; i += H {

border1 := i

border2 := i + h

if signChange(border1, border2) {

points = append(points, bisection(border1, border2, maxIterations, epsilon))

}

}

points = append(points, X1, X2)

var solutionY []float64

for \_, point := range points {

solutionY = append(solutionY, f(point))

}

minVal := math.Inf(1)

maxVal := math.Inf(-1)

for \_, y := range solutionY {

if y < minVal {

minVal = y

}

if y > maxVal {

maxVal = y

}

}

count++

fmt.Printf("%d\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\n", count, X1, X2, minVal, maxVal, math.Abs(maxVal-minVal))

}

}

func main() {

var x1, x2 float64

fmt.Print("Введите диапазон [a, b]: ")

fmt.Scan(&x1, &x2)

var N int

fmt.Print("Введите N: ")

fmt.Scan(&N)

var R float64

fmt.Print("Введите R: ")

fmt.Scan(&R)

maxIterations := 20000.0

epsilon := 1e-15

solution(x1, x2, N, R, maxIterations, epsilon)

}

