

Код программы

package main

import (

"fmt"

"math"

"os"

"sort"

)

type Interval struct {

L, R float64

}

func NegativeInterval(X Interval) Interval {

res := Division(X, Interval{L: -1, R: -1})

return res

}

func Multiplication(X, Y Interval) Interval {

tmp := make([]float64, 4)

tmp[0] = X.L \* Y.L

tmp[1] = X.L \* Y.R

tmp[2] = X.R \* Y.L

tmp[3] = X.R \* Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return res

}

func Division(X, Y Interval) Interval {

tmp := make([]float64, 4)

tmp[0] = X.L / Y.L

tmp[1] = X.L / Y.R

tmp[2] = X.R / Y.L

tmp[3] = X.R / Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return res

}

func Addition(X, Y Interval) Interval {

tmp := make([]float64, 2)

tmp[0] = X.L + Y.L

tmp[1] = X.R + Y.R

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return res

}

func Substraction(X, Y Interval) Interval {

tmp := make([]float64, 2)

tmp[0] = X.L - Y.R

tmp[1] = X.R - Y.L

sort.Float64s(tmp)

res := Interval{L: tmp[0], R: tmp[len(tmp)-1]}

return res

}

func CreateMatrix(rad float64, V int, N int) ([][]Interval, []Interval) {

B := make([]Interval, N)

A := make([][]Interval, N)

var tmp float64

for i := 1; i < N; i++ {

A[i] = make([]Interval, N)

tmp = 10.0 \* math.Cos(float64(i)+float64(V))

B[i] = Interval{L: tmp - rad, R: tmp + rad}

for j := 1; j < N; j++ {

if i == j {

tmp = 31.0 + math.Sin(float64(i))/float64(V)

A[i][i] = Interval{L: tmp - rad, R: tmp + rad}

} else {

tmp = 0.01\*float64(V) + math.Sin(float64(i)-float64(j))

A[i][j] = Interval{L: tmp - rad, R: tmp + rad}

}

}

}

return A, B

}

func PrintArr(arr [][]Interval, name string, file \*os.File) {

n := len(arr)

fmt.Println("\n\t\t", name)

fmt.Fprintln(file, fmt.Sprintf("\n\t\t%s", name))

for i := 1; i < n; i++ {

for j := 1; j < n; j++ {

fmt.Printf("[%15.7f, %15.7f] ", arr[i][j].L, arr[i][j].R)

fmt.Fprint(file, fmt.Sprintf("[%15.7f, %15.7f] ", arr[i][j].L, arr[i][j].R))

}

fmt.Println()

fmt.Fprintln(file)

}

}

func PrintTriangleArr(arr [][]Interval, name string, file \*os.File) {

n := len(arr)

fmt.Println("\n\t\t", name)

fmt.Fprintln(file, fmt.Sprintf("\n\t\t%s", name))

for i := 1; i < n; i++ {

for j := 1; j < n+1; j++ {

fmt.Printf("[%15.7f, %15.7f] ", arr[i][j].L, arr[i][j].R)

fmt.Fprint(file, fmt.Sprintf("[%15.7f, %15.7f] ", arr[i][j].L, arr[i][j].R))

}

fmt.Println()

fmt.Fprintln(file)

}

}

func PrintVector(arr []Interval, name string, file \*os.File) {

n := len(arr)

fmt.Println("\n\t\t", name)

fmt.Fprintln(file, fmt.Sprintf("\n\t\t%s", name))

for i := 1; i < n; i++ {

fmt.Printf("[%15.3f, %15.3f] ", arr[i].L, arr[i].R)

fmt.Println()

fmt.Fprintln(file, fmt.Sprintf("[%15.3f, %15.3f] ", arr[i].L, arr[i].R))

}

}

func PrintVectorX(arr []Interval, name string, file \*os.File) {

n := len(arr)

fmt.Println("\n\t\t", name)

fmt.Fprintln(file, fmt.Sprintf("\n\t\t%s", name))

for i := 1; i < n; i++ {

fmt.Printf("[%15.10f, %15.10f] ", arr[i].L, arr[i].R)

fmt.Println()

fmt.Fprintln(file, fmt.Sprintf("[%15.10f, %15.10f] ", arr[i].L, arr[i].R))

}

}

func Gaus(A [][]Interval, B []Interval) ([]Interval, [][]Interval) {

n := len(B)

triangle := make([][]Interval, n)

for i := 1; i < n; i++ {

triangle[i] = make([]Interval, n+1)

for j := 1; j < n; j++ {

triangle[i][j] = A[i][j]

}

triangle[i][n] = B[i]

}

for i := 1; i < n; i++ {

// Проверка и приведение элемента triangle[i][i] к 1 (если он не ноль).

if triangle[i][i].L != 0 && triangle[i][i].R != 0 {

pivot := triangle[i][i]

for j := i; j <= n; j++ {

triangle[i][j] = Division(triangle[i][j], pivot)

}

}

// Вычитание строки i из остальных строк.

for j := i + 1; j < n; j++ {

factor := triangle[j][i]

for k := i; k <= n; k++ {

triangle[j][k] = Substraction(triangle[j][k], Multiplication(factor, triangle[i][k]))

}

}

}

// Обратный ход

solution := make([]Interval, n)

for i := n - 1; i >= 1; i-- {

solution[i] = triangle[i][n]

for j := i + 1; j < n; j++ {

solution[i] = Substraction(solution[i], Multiplication(triangle[i][j], solution[j]))

}

}

return solution, triangle

}

func MatrixVectorMultiply(matrix [][]Interval, vector []Interval) []Interval {

numRows := len(matrix)

numCols := len(matrix[1])

result := make([]Interval, numRows)

for i := 1; i < numRows; i++ {

result[i] = Interval{}

for j := 1; j < numCols; j++ {

term := Multiplication(matrix[i][j], vector[j])

result[i] = Addition(result[i], term)

}

}

return result

}

func checkSolution(A [][]Interval, B []Interval, X []Interval) ([]Interval, Interval) {

n := len(A)

res := make([]Interval, n)

res = MatrixVectorMultiply(A, X)

nev := make([]Interval, n)

for i := 1; i < n; i++ {

nev[i] = Substraction(B[i], res[i])

}

norm := Interval{}

for i := 1; i < n; i++ {

norm = Addition(norm, Interval{L: math.Pow(nev[i].L, 2), R: math.Pow(nev[i].R, 2)})

}

var normRes Interval

normRes = Interval{L: math.Sqrt(norm.L), R: math.Sqrt(norm.R)}

return nev, normRes

}

func main() {

V := 3

rad := 0.01

N := 5

file, err := os.Create("data.txt")

if err != nil {

fmt.Println("Unable to create file:", err)

os.Exit(1)

}

defer file.Close()

A, B := CreateMatrix(rad, V, N+1)

PrintArr(A, "Вектор А", file)

PrintVector(B, "Вектор B", file)

X, triangle := Gaus(A, B)

PrintVectorX(X, "Вектор X", file)

PrintTriangleArr(triangle, "Матрица в треугольном виде", file)

res, norm := checkSolution(A, B, X)

PrintVector(res, "Вектор невязки", file)

fmt.Print("\n\tНорма вектора невязки\n")

fmt.Printf("[%15.6e, %15.6e] \n", norm.L, norm.R)

fmt.Fprintln(file, fmt.Sprintf("\n\tНорма вектора невязки"))

fmt.Fprintln(file, fmt.Sprintf("[%15.6e, %15.6e] \n", norm.L, norm.R))

}

Результат data.txt

Матрица А

[ 31.2704903, 31.2904903] [ -0.8214710, -0.8014710] [ -0.8892974, -0.8692974] [ -0.1211200, -0.1011200] [ 0.7768025, 0.7968025]

[ 0.8614710, 0.8814710] [ 31.2930991, 31.3130991] [ -0.8214710, -0.8014710] [ -0.8892974, -0.8692974] [ -0.1211200, -0.1011200]

[ 0.9292974, 0.9492974] [ 0.8614710, 0.8814710] [ 31.0370400, 31.0570400] [ -0.8214710, -0.8014710] [ -0.8892974, -0.8692974]

[ 0.1611200, 0.1811200] [ 0.9292974, 0.9492974] [ 0.8614710, 0.8814710] [ 30.7377325, 30.7577325] [ -0.8214710, -0.8014710]

[ -0.7368025, -0.7168025] [ 0.1611200, 0.1811200] [ 0.9292974, 0.9492974] [ 0.8614710, 0.8814710] [ 30.6703586, 30.6903586]

Вектор B

[ -6.546, -6.526]

[ 2.827, 2.847]

[ 9.592, 9.612]

[ 7.529, 7.549]

[ -1.465, -1.445]

Вектор X

[ -0.1953298458, -0.1939676284]

[ 0.1098283179, 0.1111892796]

[ 0.3154609039, 0.3168688440]

[ 0.2314198920, 0.2328132642]

[ -0.0696208741, -0.0682307121]

Матрица в треугольном виде

[ 0.9993608, 1.0006396] [ -0.0262698, -0.0256139] [ -0.0284389, -0.0277815] [ -0.0038733, -0.0032317] [ 0.0248255, 0.0254810] [ -0.2093487, -0.2085757]

[ -0.0205638, 0.0205506] [ 0.9993270, 1.0006735] [ -0.0254681, -0.0247765] [ -0.0283094, -0.0276320] [ -0.0045850, -0.0039094] [ 0.0959369, 0.0967952]

[ -0.0206072, 0.0205940] [ -0.0217454, 0.0217308] [ 0.9992822, 1.0007184] [ -0.0255432, -0.0248218] [ -0.0292756, -0.0285533] [ 0.3117548, 0.3128703]

[ -0.0201158, 0.0201030] [ -0.0212736, 0.0212593] [ -0.0225001, 0.0224839] [ 0.9992666, 1.0007340] [ -0.0258899, -0.0251367] [ 0.2332224, 0.2345284]

[ -0.0204582, 0.0204712] [ -0.0211052, 0.0210910] [ -0.0223433, 0.0223273] [ -0.0231054, 0.0230885] [ 0.9992291, 1.0007715] [ -0.0696209, -0.0682307]

Вектор невязки

[ -0.042, 0.042]

[ -0.042, 0.042]

[ -0.043, 0.043]

[ -0.043, 0.043]

[ -0.042, 0.042]

Норма вектора невязки

[ 9.534664e-02, 9.535543e-02]

