

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("[%15.6e, %15.6e] ", arr[i][j].L, arr[i][j].R)

fmt.Fprint(file, fmt.Sprintf("[%15.6e, %15.6e] ", 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-1; i++ {

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

fmt.Printf("[%15.6e, %15.6e] ", arr[i][j].L, arr[i][j].R)

fmt.Fprint(file, fmt.Sprintf("[%15.6e, %15.6e] ", 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("[%15.6e, %15.6e] ", arr[i].L, arr[i].R)

fmt.Println()

fmt.Fprintln(file, fmt.Sprintf("[%15.6e, %15.6e] ", arr[i].L, arr[i].R))

}

}

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

n := len(B)

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

alpha := A[k][k]

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

A[k][i] = Division(A[k][i], alpha)

}

B[k] = Division(B[k], alpha)

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

A[j][k] = Substraction(A[j][k], Multiplication(A[k][k], A[j][k]))

}

}

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

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]

}

// Обратный ход

x := make([]Interval, n)

x[n-1] = B[n-1]

for i := n - 2; i > 0; i-- {

x[i] = B[i]

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

x[i] = Substraction(x[i], Multiplication(A[i][j], x[j]))

}

}

return x, triangle

}

func CheckAnswer(triangle [][]Interval, B, X []Interval) ([]Interval, Interval) {

n := len(triangle)

res := make([]Interval, n)

// Умножение матрицы на вектор решения

res[n-1] = B[n-1]

for i := n - 2; i > 0; i-- {

res[i] = B[i]

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

res[i] = Substraction(res[i], Multiplication(triangle[i][j], X[i-1]))

}

}

nev := make([]Interval, n)

// Вычисление вектора невязки

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

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

}

// Вычисление нормы вектора невязки

var 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)

PrintVector(X, "Вектор X", file)

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

res, norm := CheckAnswer(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

Вектор А

[ 3.127049e+01, 3.129049e+01] [ -8.214710e-01, -8.014710e-01] [ -8.892974e-01, -8.692974e-01] [ -1.211200e-01, -1.011200e-01] [ 7.768025e-01, 7.968025e-01]

[ 8.614710e-01, 8.814710e-01] [ 3.129310e+01, 3.131310e+01] [ -8.214710e-01, -8.014710e-01] [ -8.892974e-01, -8.692974e-01] [ -1.211200e-01, -1.011200e-01]

[ 9.292974e-01, 9.492974e-01] [ 8.614710e-01, 8.814710e-01] [ 3.103704e+01, 3.105704e+01] [ -8.214710e-01, -8.014710e-01] [ -8.892974e-01, -8.692974e-01]

[ 1.611200e-01, 1.811200e-01] [ 9.292974e-01, 9.492974e-01] [ 8.614710e-01, 8.814710e-01] [ 3.073773e+01, 3.075773e+01] [ -8.214710e-01, -8.014710e-01]

[ -7.368025e-01, -7.168025e-01] [ 1.611200e-01, 1.811200e-01] [ 9.292974e-01, 9.492974e-01] [ 8.614710e-01, 8.814710e-01] [ 3.067036e+01, 3.069036e+01]

Вектор B

[ -6.546436e+00, -6.526436e+00]

[ 2.826622e+00, 2.846622e+00]

[ 9.591703e+00, 9.611703e+00]

[ 7.529023e+00, 7.549023e+00]

[ -1.465000e+00, -1.445000e+00]

Вектор X

[ -1.959900e-01, -1.946733e-01]

[ 1.048756e-01, 1.060236e-01]

[ 3.137569e-01, 3.148348e-01]

[ 2.435082e-01, 2.443678e-01]

[ -4.776600e-02, -4.708320e-02]

Матрица в треугольном виде

[ 9.993608e-01, 1.000640e+00] [ -2.626985e-02, -2.561388e-02] [ -2.843887e-02, -2.778152e-02] [ -3.873301e-03, -3.231653e-03] [ 2.482551e-02, 2.548097e-02] [ -2.093487e-01, -2.085757e-01]

[ -2.056377e-02, 2.055063e-02] [ 9.993613e-01, 1.000639e+00] [ -2.625087e-02, -2.559539e-02] [ -2.841832e-02, -2.776146e-02] [ -3.870502e-03, -3.229320e-03] [ 9.026963e-02, 9.096644e-02]

[ -2.060715e-02, 2.059398e-02] [ -2.056336e-02, 2.055023e-02] [ 9.993560e-01, 1.000644e+00] [ -2.646744e-02, -2.580642e-02] [ -2.865278e-02, -2.799035e-02] [ 3.088415e-01, 3.096849e-01]

[ -2.011584e-02, 2.010298e-02] [ -2.060671e-02, 2.059355e-02] [ -2.056801e-02, 2.055477e-02] [ 9.993498e-01, 1.000651e+00] [ -2.672517e-02, -2.605754e-02] [ 2.447847e-01, 2.455946e-01]

[ -2.045816e-02, 2.047124e-02] [ -2.011576e-02, 2.010291e-02] [ -2.061172e-02, 2.059845e-02] [ -2.057354e-02, 2.056017e-02] [ 9.993483e-01, 1.000652e+00] [ -4.776600e-02, -4.708320e-02]

Вектор невязки

[ -7.729816e-04, 7.729816e-04]

[ -6.968115e-04, 6.968115e-04]

[ -8.434061e-04, 8.434061e-04]

[ -8.099392e-04, 8.099392e-04]

[ -6.827981e-04, 6.827981e-04]

Норма вектора невязки

[ 1.707804e-03, 1.707804e-03]