Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Отчет

по лабораторной работе № 3

"Итерационные методы решения систем линейных уравнений."

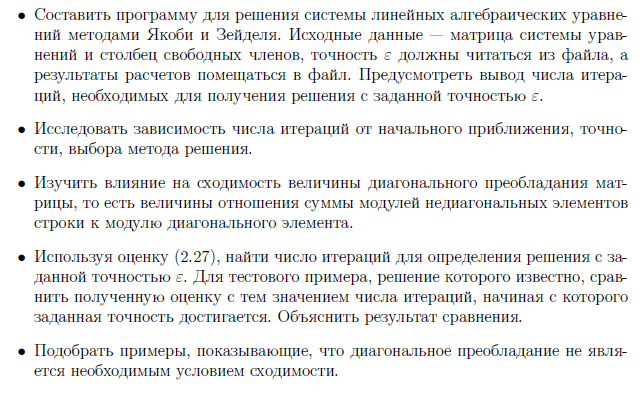
по дисциплине "Вычислительная математика"

Студент группы ПИ-72 Н. А. Черемнов

Преподаватель С. А. Кантор

Барнаул 2020

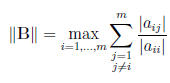
Задание:



Алгоритм:

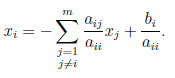
1. Критерий достижения необходимой точности: ||xk+1 – xk|| < ε

2. Оценка количества итераций:

3. Вычисление диагонального преобладания матрицы А: 

4. Метод Якоби:

При начальном приближении x0 выполнять для каждого i от 0 до i < n:



пока не будет выполнятся критерий (1).

5. Метод Зейделя: аналогично п. 4, за исключением того, что:

Текст программы:

package main

import (

"errors"

"fmt"

"log"

"math"

"myReader"

)

func main() {

n, e, A, b, xFirst, err := myReader.ReadSystem("src.txt")

fmt.Printf("Параметр диагонального преобладание матрицы A: %g\n", dp(A, n))

if err != nil {

log.Panic(err)

}

kJ, kS := predict(A, b, n, e, xFirst)

if kJ > 00 {

fmt.Printf("1. Метод Якоби\nОценочное число итераций: %d\n", kJ)

} else {

fmt.Printf("1. Метод Якоби\nОценочное число итераций отсутствует\n")

}

x, count, err := Jacobi(A, b, n, e, xFirst, math.MaxInt32)

if err != nil{

fmt.Println(err)

} else {

fmt.Println("Рещение:")

fmt.Println(x)

fmt.Printf("Число итераций: %d\n", count)

}

if kS > 0 {

fmt.Printf("2. Метод Зейделя\nОценочное число итераций: %d\n", kS)

} else {

fmt.Printf("2. Метод Зейделя\nОценочное число итераций отсутствует\n")

}

x, count, err = Seidel(A, b, n, e, xFirst, math.MaxInt32)

if err != nil{

fmt.Println(err)

} else {

fmt.Println("Рещение:")

fmt.Println(x)

fmt.Printf("Число итераций: %d\n", count)

}

}

//оценка числа итераций

func predict(A [][]float64, b []float64, n int, e float64, x0 []float64)(int, int){

var normB float64 = 0

for i := 0; i < n; i++ {

var sum float64 = 0

for j := 0; j < n; j++{

if i != j {

sum += math.Abs(A[i][j] / A[i][i])

}

}

if sum > normB {

normB = sum

}

}

x1J := make([]float64, n)

x1S := make([]float64, n)

for i := 0; i < n; i++ {

var sum float64 = 0

for j := 0; j < n; j++{

if i != j {

sum += (A[i][j] / A[i][i]) \* x0[j]

}

}

x1J[i] = b[i] / A[i][i] - sum

}

for i := 0; i < n; i++ {

var sum float64 = 0

for j := 0; j < i; j++{

sum += (A[i][j] / A[i][i]) \* x1S[j]

}

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

sum += (A[i][j] / A[i][i]) \* x0[j]

}

x1S[i] = b[i] / A[i][i] - sum

}

\_, normJ := crt1(x1J, x0, e)

\_, normS := crt1(x1S, x0, e)

//j := math.Log((e \* (1 - normB) / normJ)) / math.Log(normB)

//fmt.Println(j)

kJ := int(math.Log((e \* (1 - normB) / normJ)) / math.Log(normB)) + 1

kS := int(math.Log((e \* (1 - normB) / normS)) / math.Log(normB)) + 1

return kJ, kS

}

//метод Якоби

//вход: матрица системы, столбец свободных членов, кол-во неизвестных, точность, начальное приблежение, максимальное число итераций

//выход: вектор x, число итераций, ошибка

func Jacobi(A [][]float64, b []float64, n int, e float64, x0 []float64, maxCount int) ([]float64, int, error){

// запуск итерационного процесса решения

xk := make([]float64, n)

for i := 0; i < n; i++{

xk[i] = x0[i]

}

xk1 := make([]float64, n)

var norm, oldNorm float64

var count int //кол-во числа итераций

isEnd := false

for count = 0; !isEnd && count < maxCount; count++{

for i := 0; i < n; i++{

var sum float64 = 0

for j := 0; j < n; j++{

if i != j{

sum += (A[i][j]/A[i][i]) \* xk[j]

}

}

xk1[i] = -sum + b[i] / A[i][i]

}

isEnd, norm = crt1(xk1, xk, e) //если выполнилось условие завершения

if norm >= oldNorm && count != 0{ //если метод якоби не сходится

return nil, -1, errors.New("Метод Якоби расходится")

}

// иначе свдинуться по приближениям

a := xk

xk = xk1

xk1 = a

oldNorm = norm

//fmt.Printf("%d:", count)

//fmt.Println(xk1)

}

return xk1, count, nil

}

//метод Зейделя

//вход: матрица системы, столбец свободных членов, кол-во неизвестных, точность, начальное приблежение, максимальное число итераций

//выход: вектор x, число итераций, ошибка

func Seidel(A [][]float64, b []float64, n int, e float64, x0 []float64, maxCount int) ([]float64, int, error){

//запуск итерационного процесса решения

xk := make([]float64, n)

for i := 0; i < n; i++{

xk[i] = x0[i]

}

xk1 := make([]float64, n)

var norm, oldNorm float64

var count int //кол-во числа итераций

isEnd := false

for count = 0; !isEnd && count < maxCount; count++{

for i := 0; i < n; i++{

var sum float64 = 0

for j := 0; j < i; j++{

sum += (A[i][j]/A[i][i]) \* xk1[j]

}

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

sum += (A[i][j]/A[i][i]) \* xk[j]

}

xk1[i] = -sum + b[i] / A[i][i]

}

isEnd, norm = crt1(xk1, xk, e) //если выполнилось условие завершения

if norm >= oldNorm && count != 0{ //если метод якоби не сходится

return nil, -1, errors.New("Метод Зейделя расходится")

}

// иначе свдинуться по приближениям

a := xk

xk = xk1

xk1 = a

oldNorm = norm

}

return xk1, count, nil

}

//критериии завершения итераций

//1.

func crt1(xk1 []float64, xk []float64, e float64) (bool, float64){

var norm float64 = -1

for i, \_ := range xk {

if math.Abs(xk1[i] - xk[i]) > norm {

norm = math.Abs(xk1[i] - xk[i])

}

}

return norm < e, norm

}

//вычисления диагонального преобладания матрицы

func dp(A [][]float64, n int) float64 {

var res float64 = 0

for i := 0; i < n; i++{

var sum float64 = 0

for j := 0; j < n; j++{

if i != j {

sum += math.Abs(A[i][j] / A[i][i])

}

}

if sum > res{

res = sum

}

}

return res

}

Тесты:

|  |  |
| --- | --- |
| исходные данные | результат |
| 3 0.0001  3 1 1 0  1 3 1 0  1 1 3 0  1 1 1 | Параметр диагонального преобладание матрицы A: 0.6666666666666666  1. Метод Якоби  Оценочное число итераций: 27  Рещение:  [5.940319206354907e-05 5.940319206354907e-05 5.940319206354907e-05]  Число итераций: 25  2. Метод Зейделя  Оценочное число итераций: 27  Рещение:  [6.384105651503854e-07 2.1753854347359737e-05 -7.46408830417004e-06]  Число итераций: 8 |
| 2 0.00001  2.2 1.1 3.3  0.02 0.15 0  1 1 | Параметр диагонального преобладание матрицы A: 0.5  1. Метод Якоби  Оценочное число итераций: 18  Рещение:  [1.6071420576131685 -0.21428411522633742]  Число итераций: 11  2. Метод Зейделя  Оценочное число итераций: 18  Рещение:  [1.6071420576131685 -0.2142856076817558]  Число итераций: 7 |
| 3 0.01  1 2 3 4  5 6 7 8  9 10 11 12  1 1 1 | Параметр диагонального преобладание матрицы A: 5  1. Метод Якоби  Оценочное число итераций отсутствует  Метод Якоби расходится  2. Метод Зейделя  Оценочное число итераций отсутствует  Рещение:  [-1 0.9999999999999999 1]  Число итераций: 2 |
| 4 0.0001  0.1 0.2 0.3 0.4 1  1 2 3 4 1  3 4 10 0.1 1  1 1 1 1 2.1  1 1 1 1 | Параметр диагонального преобладание матрицы A: 9  1. Метод Якоби  Оценочное число итераций отсутствует  Метод Якоби расходится  2. Метод Зейделя  Оценочное число итераций отсутствует  Метод Зейделя расходится |
| 4 0.0001  1 0.2 0.3 0.4 1  1 3 0.3 0.4 1  3 0.4 10 0.1 1  1 1 1 10 2.1  1 1 1 1 | Параметр диагонального преобладание матрицы A: 0.9  1. Метод Якоби  Оценочное число итераций: 112  Рещение:  [1.009503168163987 4.3381411057696884e-05 -0.20409813231846743 0.12949128274395438]  Число итераций: 17  2. Метод Зейделя  Оценочное число итераций: 109  Рещение:  [1.0094285169942687 8.779149512916362e-08 -0.20412337645663722 0.12946947716708734]  Число итераций: 7 |

Вывод:

1) Чем ближе начальное приближение к решению, тем меньшее количество итераций требуется.

2) Чем выше заданная точность (чем меньше ε), тем большее количество итераций требуется.

3) Метод Зейделя требует меньшее количество итераций, чем метод Якоби, в может сходится, когда метод Якоби не сходится.

4) Почти всегда при диагональном преобладании > 1 итерационные методы расходятся.

5) Оценочное число итераций, как правило, больше фактического. Это объясняется неточностью оценки, т.к. для точной оценки неизвестно истинное решение системы.