Санкт-Петербургский Государственный Политехнический Университет

Институт прикладной математики и механики

Кафедра «Прикладная математика»

# 

# 

# 

# 

Курсовой проект по дисциплине «Реализация численных методов»

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

Студент: Зубкова Д. А.

Группа: 23631/3

Преподаватель: Павлова Л. В.

Оглавление

[Лабораторная 1: Численные методы решения нелинейных уравнений (Метод хорд и метод половинного деления) 2](#_Toc533073377)

[Формулировка задачи и ее формализация. 2](#_Toc533073378)

[Алгоритм метода и условия его применимости 2](#_Toc533073379)

[Предварительный анализ задачи и проверка условий применимости 3](#_Toc533073380)

[Тестовый пример с детальными расчетами для задачи малой размерности (метод хорд) 3](#_Toc533073381)

[Перечень контрольных тестов для иллюстрации метода средствами пакета MATLAB 4](#_Toc533073382)

[Модульная структура программы 5](#_Toc533073383)

[Численный анализ решения задачи 6](#_Toc533073384)

[Вывод 9](#_Toc533073385)

[Лабораторная 2: Решения СЛАУ прямыми методами(LU-разложение) 9](#_Toc533073386)

[Формулировка задачи и ее формализация 9](#_Toc533073387)

[Алгоритм метода и условия его применимости 9](#_Toc533073388)

[Предварительный анализ задачи и проверка условий применимости 10](#_Toc533073389)

[Тестовый пример с детальными расчетами для задачи малой размерности (LU-разложение) 11](#_Toc533073390)

[Алгоритм создания матрицы с заданным числом обусловленности в MATLAB 11](#_Toc533073391)

[Модульная структура программы 12](#_Toc533073392)

[Численный анализ решения задачи 13](#_Toc533073393)

[Вывод 15](#_Toc533073394)

[Лабораторная 3: Решение СЛАУ итерационными методами(МПИ) 15](#_Toc533073395)

[Формулировка задачи и ее формализация 15](#_Toc533073396)

[Алгоритм метода и условия его применимости 15](#_Toc533073397)

[Предварительный анализ задачи и проверка условий применимости 17](#_Toc533073398)

[Тестовый пример с детальными расчетами для задачи малой размерности (МПИ) 17](#_Toc533073399)

[Алгоритм создания матрицы с заданным числом обусловленности в MATLAB 18](#_Toc533073400)

[Модульная структура программы 18](#_Toc533073401)

[Численный анализ решения задачи 19](#_Toc533073402)

[Вывод 19](#_Toc533073403)

[Лабораторная 4: Решение алгебраической проблемы собственных значений(Метод Якоби) 19](#_Toc533073404)

[Формулировка задачи и ее формализация 19](#_Toc533073405)

[Алгоритм метода и условия его применимости 19](#_Toc533073406)

[Предварительный анализ задачи и проверка условий применимости 21](#_Toc533073407)

[Тестовый пример с детальными расчетами для задачи малой размерности (Метод Якоби) 21](#_Toc533073408)

[Алгоритм создания матрицы с заданным числом обусловленности в MATLAB 22](#_Toc533073409)

[Модульная структура программы 22](#_Toc533073410)

[Численный анализ решения задачи 23](#_Toc533073411)

[Вывод 24](#_Toc533073412)

# Лабораторная 1: Численные методы решения нелинейных уравнений (Метод хорд и метод половинного деления)

## Формулировка задачи и ее формализация.

Определение корней алгебраического и трансцендентного уравнений методом хорд и методом половинного деления.

## Алгоритм метода и условия его применимости

1. Метод хорд  
   Алгоритм:  
   1. Выбор постоянной точки b:

Если

b:   
то b – является постоянной точкой, иначе a – постоянная точка.

2. Проведем хорду через точку x\*, такую что

х(i – 1) = a(в начале), x1 = b – постоянная точка.  
После х(i – 1) = x(i);

Последующие точки выбираются также.

3. Останавливается, когда

Где ɛ - точность, , на [a,b]

Условия применимости:

1. f(x) – определена и непрерывна до второй производной на отрезке [a,b].
2. f(a)\*f(b)<0
3. f’(x) – знакопостоянна.
4. f”(x) – знакопостоянна.
5. x1:   
   где x1 – постоянная точка.
6. Метод половинного деления

Алгоритм:

1. Выберем точку c:
2. Если (b – a) < 2\*ɛ, то х = с и остановиться.
3. Вычислить f(c).
4. Если f(c) = 0, то x = c и остановиться.
5. Если f(a)\*f(c)<0, то b = c и f(b) = f(c), вернуться к шагу 1, иначе a = c и f(a) = f(c), вернуться к шагу 1.
6. Условие выхода:

Условия применимости:

1. f(x) – определена и непрерывна до первой производной на отрезке [a,b].
2. f(a)\*f(b)<0

## Предварительный анализ задачи и проверка условий применимости

Условия f(x) – определена и непрерывна до второй производной на отрезке [a,b] и f(a)\*f(b)<0 гарантируют, что на интервале [*a,* *b*] находится хотя бы 1 корень.

Из того, что f’(x) – знакопостоянна, следует, что *f*(*x*) на данном интервале монотонна => можно применять метод половинного деления.

f”(x) – знакопостоянна => можно применять метод хорд

## Тестовый пример с детальными расчетами для задачи малой размерности (метод хорд)

*ɛ = 0,0001*

⬄ (-5)\*4 < 0 => существует хотя бы 1 корень.

точка b – постоянная.

f’(x) b f”(x) – знакопостоянны => f(x) – монотонна.

m1 = min {f(0), f(3)} = min {-5, 4} = -5;

M1 = max {f(0), f(3)} = max {-5, 4} = 4;



Значит ищем дальше.

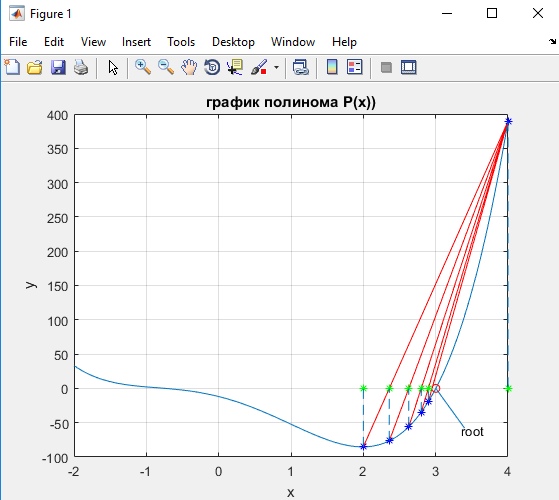
## Перечень контрольных тестов для иллюстрации метода средствами пакета MATLAB

1. Алгебраическое уравнение

|  |  |  |
| --- | --- | --- |
| С помощью fzero | Метод хорд | МПД |
| 3,002106294136305 | 3,002090 | 3,002084 |
| -0,738326766577249 | -0,738354 | -0,738342 |

1. Трансцендентное уравнение

|  |  |  |
| --- | --- | --- |
| С помощью fzero | Метод хорд | МПД |
| 0.438430779481519 | 0,438430 | 0,438416 |
| -0.438430779481519 | -0,438431 | -0,438403 |



## Модульная структура программы

double One(double x):

входящие данные: х – значение

выходящие данные: значение алгебраич. ур-я в х.

double Two(double x):

входящие данные: х – значение

выходящие данные: значение трансцендент. ур-я в х.

double FunHord(double a, double b, double f1, double f2, double e, int\* k, double num, double m1, double M1):

входящие данные: a – нижний конец промежутка, b – верхний конец, f1 и f2 – значения функций в точках а и b, k –количество итераций, e –эпсилон, num - ярлык, чтобы знать какое ур-е, m1 –минимум значения производной, M1 – максимум.

выходящие данные: корень.

double FunTwoDel(double a, double b, double f1, double f2, double e, int\* k, double num):

входящие данные: то же самое, что и предыдущая, только без минимума и максимума значения производной.

выходящие данные: корень.

void PrintHord(double x, int k, int num):

входящие данные: х – корень, k – количество итераций, num – ярлычок для ур-я.

void PrintTwoDel(double x, int k, int num):

входящие данные: то же самое, что и предыдущая функция.

int Prov(double a, double b, double num):

входящие данные: концы отрезка и ярлычок.

выходящие данные: число, по которому мы понимаем, знакопостоянна ли производная.

double Prover(double a, double b, double num):

входящие данные: концы отрезка и ярлычок.

выходящие данные: число, по которому мы понимаем, знакопостоянна ли производная.

## Численный анализ решения задачи

1. Алгебраическое уравнение  
   Формула для поиска верхней и нижней границы:

Где ,

Для поиска верхней границы положительных корней

Для нижней границы положительных корней

Для верхней границы отрицательных корней

Для нижней границы отрицательных корней

Для положительных:

Верхняя граница:

Нижняя граница:

Для отрицательных:

Верхняя граница:

Нижняя граница:

a1 = 1,3

b1 = 4

a2 = -1

b1 = -0,5

f(a1)\*f(b1)<0: (-76,54) \* 379,02 < 0 => существует хотя бы 1 корень.

f(a2)\*f(b2)<0: 2,21 \* (-2,45) < 0 => существует хоты бы 1 корень.

Если , то – постоянная точка, иначе если , то – постоянная точка.

a1: (-76,54) \* 26,04 < 0

b1: 379,02 \* 562,76 > 0 => b1 – const

a2: 2,21 \* 10,49 > 0 => a2 – const

b2: (-2,45) \* 3,81 < 0

m1(1) = min{f(1,3), f(4)} = min{-76,54, 379,02 } = -76,54;

M1(1) = max{f(1,3), f(4)} = max{-76,54, 379,02 } = 379,02;

m1(2) = min{f(-1), f(-0,5)} = min{2,21, -2,45} = -2,45;

M1(2) = max{f(-1), f(-0,5)} = max{2,21, -2,45} = 2, 21;

1. Трансцендентное

a3 = 0

b3 = 0,5

a4 = -0,2

b4 = -0,5

f(a)\*f(b)<0: 3 \* (-1) < 0 => существует хотя бы 1 корень.

f(a)\*f(b)<0: (-0,77) \* 0,25 < 0 => существует хотя бы 1 корень.

Пусть ɛ = 0,0001

Тогда на [1,3, 4]:

Тогда на [-1, -0,5]:

Теперь ищем корни по методам.

Корни:

1. Алгебраического уравнения

Метод хорд:

На промежутке [1,3,4] x = 3,002090

На промежутке [-1,-0,5] x = -0,738354

Метод половинного деления:

На промежутке [1,3,4] x = 3,002084

На промежутке [-1,-0,5] x = -0,738342

1. Трансцендентного уравнения

Метод хорд:

На промежутке [0,0,5] x = 0,438430

На промежутке [-0,5,-0,2] x = -0,438431

Метод половинного деления:

На промежутке [0,0,5] x = 0,438416

На промежутке [-0,5,-0,2] x = -0,438403

Исследование зависимость числа итераций от заданной точности:

1. Алгебраическое уравнение

Промежуток [1,3, 4], корень 3,002090

|  |  |  |
| --- | --- | --- |
| Значение эпсилон | Количество итераций | |
| МХ | МПД |
| 0,001 | 15 | 11 |
| 0,0001 | 29 | 14 |
| 0,00001 | 35 | 18 |
| 0,000001 | 43 | 21 |

Промежуток [-1, -0,5], корень -0,738354

|  |  |  |
| --- | --- | --- |
| Значение эпсилон | Количество итераций | |
| МХ | МПД |
| 0,001 | 3 | 6 |
| 0,0001 | 9 | 12 |
| 0,00001 | 10 | 14 |
| 0,000001 | 19 | 18 |

1. Трансцендентное уравнение

Промежуток [0, 0,5], корень 0,438430

|  |  |  |
| --- | --- | --- |
| Значение эпсилон | Количество итераций | |
| МХ | МПД |
| 0,001 | 6 | 8 |
| 0,0001 | 11 | 12 |
| 0,00001 | 16 | 14 |
| 0,000001 | 19 | 14 |

Промежуток [-0,5, -0,2], корень -0,438403

|  |  |  |
| --- | --- | --- |
| Значение эпсилон | Количество итераций | |
| МХ | МПД |
| 0,001 | 3 | 8 |
| 0,0001 | 10 | 11 |
| 0,00001 | 12 | 14 |
| 0,000001 | 14 | 14 |

## Вывод

В ходе лабораторной работы мы изучили и научились использовать 2 метода отделения корней.

В ходе исследования было установлено, что метод хорд является более эффективным, чем метод половинного деления, когда график не резко убывает или возрастает, т.е. на менее крутом промежутке лучше работает метод хорд, а на крутом наклоне функции лучше работает метод половинного деления (смотреть график MatLab). Также в методе половинного деления требуется меньше условий применимости, чем в методе хорд.

# Лабораторная 2: Решения СЛАУ прямыми методами(LU-разложение)

## Формулировка задачи и ее формализация

Найти решение СЛАУ (Ах = b) методом LU-разложения, сравнить точность решения для матриц с разным числом обусловленности. Исследовать матрицу 10х10.

## Алгоритм метода и условия его применимости

LU-разложение:

А = LU, где

Алгоритм

………………………………………………………………………….

В итоге получаются формулы:

Условия применимости

Все главные миноры матрицы А отличны от нуля:

Вычеркиваем (n – j) столбцы и (n – j) строки из матрицы А:

## Предварительный анализ задачи и проверка условий применимости

Мы задаем матрицу так, что это условие выполняется, иначе нет смысла рассматривать систему.

## Тестовый пример с детальными расчетами для задачи малой размерности (LU-разложение)

,

## Алгоритм создания матрицы с заданным числом обусловленности в MATLAB

Поскольку мы будем смотреть на зависимость коэффициентов при внесении возмущений от числа обусловленности, то для удобства сами задаем нужную нам матрицу:

Задаем рандомный столбец ;

;

;

, с помощью которой можем найти число обусловленности.

Так как мы сами задаем матрицу D, то все главные миноры матрицы А будут отличны от 0;

Вычисляем число обусловленности матрицы и заносим всё в файл.

## Модульная структура программы

double\* ReadLine(FILE\* input, int n, double\* cond) :

входящие данные: файл, количество столбцов(строк), число обусловленности матрицы.

выходящие данные: матрицу.

void Print(double\* a, int n):

входящие данные: матрицу, количество столбцов(строк).

void Method(double\* a, int n, double\*\* l, double\*\* u):

входящие данные: исходную матрицу А, количество строк(столбцов), указатель на матрицы l и u.

void Prover(double\* l, double\* u, int n):

входящие данные: матрицы l и u, количество строк(столбцов).

double\* Resh(double\* l, double\* u, double\* b, int n):

входящие данные: матрицы l и u, столбец b, количество строк(столбцов).

выходящие данные: столбец х.

void Prover1(double\* a, double\* x, int n):

входящие данные: матрицы a, столбец x, количество строк(столбцов).

double Koef(double norm1, double norm2, double norm3, double norm4):

входящие данные: нормы(в 1 случае х, х, b, b; во 2 – Х, х + х, А, А).

выходящие данные: коэффициент.

void VozAX(double\* a, double\* b, double\* x, int n, double\* normX2, double\* normA, double\* normA1):

входящие данные: матрицу А, столбцы b и x, количество строк(столбцов), указатель на нормы х+х, А, А.

void VozBX(double\* l, double\* u, double\* b, double\* x, int n, double\* normX, double\* normX1, double\* normB, double\* normB1):

входящие данные: матрицы l и u, столбцы b и x, количество строк(столбцов), указатель на нормы х, х, b, b.

double VectorD(double\* a, double\* x, double\* b, int n):

входящие данные: матрицу А, столбцы b и x, количество строк(столбцов).

выходящие данные: вектор невязки.

## Численный анализ решения задачи

Известно, что справедливы следующие неравенства:

Эти неравенства всегда выполнены.

Но интерес представляют коэффициенты, при которых получается равенство.

Поэтому получаются такие формулы:

Для хорошо обусловленной матрицы:

сond(A) = 15,3333;

Дельта Х:

Х1 = 0.00000011998099269661

Х2 = 0.00000014467633469017

Х3 = -0.00000024322049241299

Х4 = -0.00000029711252425013

Х5 = -0.00000027514972866793

Х6 = 0.00000032045981751549

Х7 = 0.00000019766081305050

Х8 = 0.00000034980822594122

Х9 = -0.00000001039884017784

Х1- = -0.00000009589727500980

Норма вектора невязки: 0.00000000000000442421

Внесение возмущений в b.

Норма вектора невязки: 0.00000000000001230095

.

Внесение возмущений в A.

Норма вектора невязки: 0.00000000000000903588

.

Для плохо обусловленной матрицы:

сond(A) = 27272727,2698;

Дельта Х:

Х1 = -0.00000027518183346592

Х2 = 0.00000027595105492328

Х = -0.00000003392442187930

Х4 = -0.00000029511282884087

Х5 = -0.00000041435155830050

Х6 = -0.00000017805646082558

Х7 = 0.00000017146599051027

Х8 = 0.00000031333223898811

Х9 = 0.00000025987847773745

Х10 = 0.00000025110885037982

Норма вектора невязки: 0.000000010391657

Внесение возмущений в b.

Норма вектора невязки: 0.000000012431603

.

Внесение возмущений в A.

Норма вектора невязки: 0.000000000230689

.

## Вывод

При внесении возмущений в вектор b, коэффициент практически не зависит от числа обусловленности. А при внесении возмущений в матрицу А, коэффициент при плохо обусловленной матрице становится очень большим, но все равно меньше числа обусловленности.

# Лабораторная 3: Решение СЛАУ итерационными методами(МПИ)

## Формулировка задачи и ее формализация

Найти решение СЛАУ (АХ = b) методом простых итераций и сравнить точность решения. Исследовать матрицу 10х10.

## Алгоритм метода и условия его применимости

Метод простых итераций.

Дана система линейных алгебраических уравнений: AХ = b,

Где А = (аij)ni,j=1 – n x n – матрица, а x = (x1;…;xn)T и b = (b1;…;bn)T – n – мерные векторы-столбцы. Тем или иным способом система может быть преобразована к эквивалентной системе вида: x = C\*х + g, где х – тот же вектор неизвестных, а С и g – некоторые новые матрица и вектор соответственно.

Эту систему можно трактовать как задачу о неподвижной точке линейного отображения С в пространстве Rn и по аналогии со скалярным случаем определить последовательность приближений x(k) к неподвижной точке x\* рекуррентным равенством: x(k+1) = C\* x(k) + g, k = 0,1,2,…

Итерационный процесс начинается с некоторого вектора x(0)=(x1(0);…;xn(0)).

Условия сходимости

Необходимым и достаточным условием сходимости МПИ ( Ck-> при k->∞) при любом начальном векторе x(0) к решению х\* является требование, чтобы все собственные числа матрицы С были по модулю меньше 1:

Пусть . Тогда при любом начальном векторе x(0) МПИ сходится к единственному решению х\*.

А так как , то является достаточным условием сходимости МПИ.

Но оказывается, что МПИ будет сходится, если матрица А – симметричная и положительно определённая, даже не выполняя необходимое и достаточное условие применимости.

С и g можно найти разными способами.

Например, так:

………………………………………………………...

Из этого системы мы находим матрицу С и столбец g:

Но очень сложно получить матрицу, где норма матрицы С будет меньше 1, поэтому мы пользуемся тем, что задаем симметричную и положительно определённую матрицу А:

И тогда матрицу С и столбец g можно найти таким способом:

С = Е – α\*А, где Е – единичная матрица, α = , где N – норма матрицы А.

g = α \* b;

И при этом нам уже не важно, какая норма у матрицы С.

Алгоритм

1. Выбираем начальное приближение.

В своей работе я выбрала такое начальное приближение: х(0)= g.

1. х(1)= g + C\* х(0).
2. х(2)= g + C\* х(1).

В итоге получаем формулу итерационного процесса: х(k) = g + C\* х(k-1)

Выход из итерационного процесса при

## Предварительный анализ задачи и проверка условий применимости

Мы задаем матрицу А так, чтобы она была положительно определённой и симметричной, потому что тогда нам неважно какая норма матрицы С.

## Тестовый пример с детальными расчетами для задачи малой размерности (МПИ)

ɛ = 0.001;

α =

C = - \* =

g = \* =

1. x(0) = g.

x(1) = \* + =

1. x(2) =
2. И так далее.

## Алгоритм создания матрицы с заданным числом обусловленности в MATLAB

Задаем рандомный столбец ;

;

;

.

Так как мы сами задаем матрицу D, то все главные миноры матрицы А будут отличны от 0, ʌ > 0;

Матрица А является симметричной и положительно определенной.

Альфа = , где N – норма матрицы А.

Находим альфа и заносим всё в файл.

## Модульная структура программы

double\* ReadLine(FILE\* input, int n)

входящие данные: файл, количество столбцов(строк).

выходящие данные: матрица.

void Print(double\* a, int n)

входящие данные: матрица, количество столбцов(строк).

double NormMatrix(double\* a, int n)

входящие данные: матрица, количество столбцов(строк).

выходящие данные: норма матрицы.

double NormColumn(double\* a, int n)

входящие данные: матрица, количество столбцов(строк).

выходящие данные: норма матрицы.

void Method2(double\*\* c, double\*\* g, double\* a, double\* b, int n, double l)

входящие данные: указатель на матрицу С, указатель на столбец g, матрица А, столбец b, количество столбцов(строк), альфа.

double\* Mul(double\* c, double\* x, int n)

входящие данные: матрица, столбец, количество столбцов(строк).

выходящие данные: столбец.

double\* Resh(double\* c, double\* g, int n, double normC, int\* k, double e)

входящие данные: матрица С, столбец g, количество столбцов(строк), норма матрицы С, указатель на количество итераций, эпсилон.

выходящие данные: столбец.

double\* Nez(double\* a, double\* b, double\* x, int n)

входящие данные: матрица А, столбец b, столбец х, количество столбцов(строк).

выходящие данные: столбец невязки.

## Численный анализ решения задачи

|  |  |  |  |
| --- | --- | --- | --- |
| Det(A) | ɛ | Количество итераций | Норма вектора невязки |
| 7302202876799.989258 | 10-3 | 78 | 0.01145871361319583670 |
| 10-4 | 103 | 0.00110443313056673986 |
| 10-5 | 127 | 0.00011677935591847266 |
| 4.7849e-19 | 10-3 | 203 | 0.00001433545546412353 |
| 10-4 | 284 | 0.00000144184403887506 |
| 10-5 | 356 | 0.00000014501905901262 |

## Вывод

Когда определитель матрицы А не близок к нулю, то вектор невязки больше, чем вектор невязки, когда определитель матрицы А близок к нулю. Получается, чем ближе к нулю определитель, тем меньше вектор невязки и больше количество итераций.

# Лабораторная 4: Решение алгебраической проблемы собственных значений(Метод Якоби)

## Формулировка задачи и ее формализация

Решить алгебраическую проблему собственных значений итерационным методом вращения Якоби. Исследовать метод Якоби при хорошей и плохой отделимости искомых собственных чисел.

## Алгоритм метода и условия его применимости

Определение. Ненулевой вектор http://mathprofi.ru/k/sobstvennye_znachenija_i_sobstvennye_vektory_clip_image018_0000.gif, который при умножении на некоторую квадратную матрицу http://mathprofi.ru/k/sobstvennye_znachenija_i_sobstvennye_vektory_clip_image016_0000.gif превращается в самого же себя с числовым коэффициентом http://mathprofi.ru/k/sobstvennye_znachenija_i_sobstvennye_vektory_clip_image025.gif, называется **собственным вектором** матрицы http://mathprofi.ru/k/sobstvennye_znachenija_i_sobstvennye_vektory_clip_image016_0001.gif. Число http://mathprofi.ru/k/sobstvennye_znachenija_i_sobstvennye_vektory_clip_image025_0000.gif называют **собственным значением**или**собственным числом** данной матрицы.

Метод вращения Якоби применяется только для симметричных матриц.

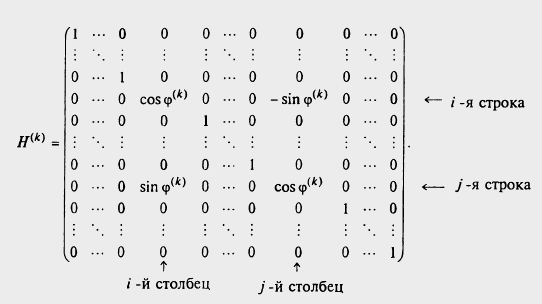
Метод основывается на использовании ортогональных матриц.

Итерационная формула, для которой А(0) = А: А(k+1) = (H(k))-1\*A(k)\*H(k) = (H(k))T\*A(k)\*H(k) k = 0, 1, 2,…; где H – ортогональная матрица плоских вращений (на k-ом шаге обнуляется максимальный по модулю элемент матрицы H(k) предыдущего шага(а значит и симметричный ему элемент)). Это определяет способ фиксирования пары индексов i и j и угол поворота ϕ.

На каждом шаге таких преобразований пересчитываются только 2 строчки матрицы предыдущего шага.

Но полученный на некотором этапе преобразований нулевые элементы на следующем этапе станут ненулевыми, но предельное поведение всё равно будет: A(k)->D при k->∞, где D = diag(**λ**i) I = 1,2,…,n.

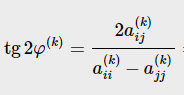
На k-й итерации для элемента а(k)ij, ij, определяется ортогональная матрица H(k), приводящая этот элемент а(k+1)ij к нулю. При этом на каждом шаге итерации в качестве а(k+1)ij выбирается наибольший по модулю.

Матрица H(k) зависит от угла ϕ(k) и имеет вид:

В данной ортогональной матрице элементы на главной диагонали единичные кроме

hii(k)=cosϕ, hjj(k)=cosϕ, hij(k)=-sinϕ, hji(k)=sinϕ.

А угол ϕ вычисляется по формуле:



Алгоритм

1. Положить A(0) = A и задать ɛ.
2. Выделить в матрице А максимальный по модулю элемент |aij(k)| i<j.
3. Найти угол поворота ϕ(k)= )
4. Составить матрицу H(k).
5. Вычислить приближение A(k+1) = (H(k))T\*A(k)\*H(k).
6. Если |aij(k)| ɛ для всех ij, то процесс завершить.

Найти собственные значения **λ**i(A(k))=a(k)ii i = 1,2,…

Собственные векторы Xi находятся как i-ые столбцы матрицы, получившейся в результате перемножения:

V = H(0)\*H(1)\* H(2)\* H(3)\*… \*H(k-1)=(Х1, Х2,…,Хn).

Если |aij(k)| ɛ , то перейти к пункту 2.

Условия применимости

Матрица А должна быть симметрична:

АТ = А.

## Предварительный анализ задачи и проверка условий применимости

Мы задаем матрицу А так, чтобы она была симметричной.

## Тестовый пример с детальными расчетами для задачи малой размерности (Метод Якоби)

1. A(0) = A, ɛ = 0.001.
2. a(0)13= 2 > ɛ.
3. ϕ(0)= = 0.553574 => cosϕ(0) = 0.85065, sinϕ(0) = 0.52537.
4. Матрица вращения H(0)=
5. A(1)= (H(0))T\* A(0)\* H(0)
6. Вычисляем максимальный элемент новой матрицы А:

a(1)12=1.376>ɛ и так далее.

## Алгоритм создания матрицы с заданным числом обусловленности в MATLAB

Задаем матрицу А нужного нам вида, чтобы сравнить при хорошей отделимости собственных чисел и плохой отделимости собственных чисел:

Задаем рандомный столбец ;

;

;

.

Матрица А является симметричной.

## Модульная структура программы

double\* ReadLine(FILE\* input, int n)

входящие данные: файл, количество столбцов(строк).

выходящие данные: матрица.

void Print(double\* a, int n)

входящие данные: матрица, количество столбцов(строк).

double\* Triangle(double\* a, int n)

входящие данные: матрица, количество столбцов(строк).

выходящие данные: верхняя треугольная матрица.

double Max(double\* k, int n, int\* i, int\* j)

входящие данные: матрица, количество столбцов(строк), указатели на строку и столбец.

выходящие данные: максимальный по модулю элемент.

double\* Eye(int n)

входящие данные: количество столбцов(строк).

выходящие данные: единичная матрица.

double\* Trans(double\* h, int n)

входящие данные: файл, количество столбцов(строк).

выходящие данные: матрица.

double\* Mul(double\* m1, double\* m2, int n)

входящие данные: 1 матрица, 2 матрица, количество столбцов(строк).

выходящие данные: умноженная матрица.

void PrintVectors(double\* v, int n)

входящие данные: матрица, количество столбцов(строк).

double NormColumn(double\* a, int n)

входящие данные: столбец, количество столбцов(строк).

выходящие данные: норма столбца.

void Prov(double\* a, double\* v, double\* y, int n)

входящие данные: матрица А, столбцы Х , столбец лямбда, количество столбцов(строк).

## Численный анализ решения задачи

Хорошее число обусловленности: 6.125

Хорошая отделимость собственных чисел:

|  |  |  |
| --- | --- | --- |
| ɛ | Количество итераций | max||AX-ʌX|| |
| 10-4 | 9 | 0.00009897077077144359 |
| 10-5 | 15 | 0.00001401590273127118 |
| 10-6 | 24 | 0.00000004509284285265 |
| 10-8 | 52 | 0.00000000929018519856 |
| 10-10 | 73 | 0.00000000025292851204 |
| 10-12 | 85 | 0.00000000000092550451 |

Хорошее число обусловленности: 1.0003

Плохая отделимость собственных чисел:

|  |  |  |
| --- | --- | --- |
| ɛ | Количество итераций | max||AX-ʌX|| |
| 10-4 | 1 | 0.00013187565143475632 |
| 10-5 | 9 | 0.00002120055173694791 |
| 10-6 | 20 | 0.00000135144435131038 |
| 10-8 | 28 | 0.00000003178962137626 |
| 10-10 | 41 | 0.00000000021298246203 |
| 10-12 | 59 | 0.00000000000247761524 |

Плохое число обусловленности: 4.9870е+05

Хорошая отделимость собственных чисел:

|  |  |  |
| --- | --- | --- |
| ɛ | Количество итераций | max||AX-ʌX|| |
| 10-4 | 134 | 0.00022533396258950233 |
| 10-5 | 138 | 0.00000358688284229913 |
| 10-6 | 139 | 0.00000049965456128120 |
| 10-8 | 144 | 0.00000038765462876213 |
| 10-10 | 149 | 0.00000025506520867348 |
| 10-12 | 153 | 0.00000023757208571053 |

Плохое число обусловленности: 5.5124е+05

Плохая отделимость собственных чисел:

|  |  |  |
| --- | --- | --- |
| ɛ | Количество итераций | max||AX-ʌX|| |
| 10-4 | 122 | 0.00016172905452549458 |
| 10-5 | 126 | 0.00002491101622581482 |
| 10-6 | 132 | 0.00000157998874783516 |
| 10-8 | 142 | 0.00000143656507134438 |
| 10-10 | 159 | 0.00000140676274895668 |
| 10-12 | 166 | 0.00000135874634172465 |

## Вывод

Видим, что количество итераций в матрице с хорошим числом обусловленности и с плохой отделимостью собственных значений меньше, чем количество итераций при том же эпсилон с хорошей отделимостью собственных значений. Значит метод работает лучше с плохой отделимостью собственных значений для матриц с хорошим числом обусловленности.

Количество итераций также зависит от числа обусловленности: при хорошем числе обусловленности количество итераций меньше, чем при плохом числе обусловленности.