Московский авиационный институт
(Национальный Исследовательский Институт)

Институт №8 информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1
по курсу «Численные методы»

Студент: Суханов Е.А

Группа: М8О-406Б-19

Оценка:

Преподаватель: Ревизников Д.Л.

Подпись:

Москва
2022

Задание 1.1

Реализовать алгоритм LU - разложения матриц (с выбором главного элемента) в виде программы. Используя разработанное программное обеспечение, решить систему линейных алгебраических уравнений (СЛАУ). Для матрицы СЛАУ вычислить определитель и обратную матрицу.

Вариант:

$$\begin{cases} -2 \cdot x_1 - 9 \cdot x_2 - 3 \cdot x_3 + 7 \cdot x_4 = -26 \\ -7 \cdot x_1 + 8 \cdot x_2 + 2 \cdot x_3 + 5 \cdot x_4 = -25 \\ -6 \cdot x_1 + 2 \cdot x_2 = -16 \\ -3 \cdot x_2 + 8 \cdot x_3 - 3 \cdot x_4 = -5 \end{cases}$$

Листинг

main.go:

```go
package main

import (
  "fmt"

  "github.com/Reterer/number_methods/internal/lu_decompose"
  "github.com/Reterer/number_methods/internal/utils"
  "github.com/Reterer/number_methods/internal/utils_lab_1_1/config"
  "github.com/Reterer/number_methods/pkg/matrix"
)

func MakeLU(A *matrix.RMatrix) *lu_decompose.LU {
  cnfg := config.Get()
  permFunc := lu_decompose.PermMin

  switch cnfg.AutoPerm {
  case config.APM_nope:
    permFunc = nil
  case config.APM_min:
    permFunc = lu_decompose.PermMin
  case config.APM_all:
    permFunc = lu_decompose.PermEveryIteration
  }

  switch cnfg.UserPerm {
  case config.UPM_nope:
    // nothing
  case config.UPM_once:
    permFunc = lu_decompose.UPermOnceMake(permFunc, cnfg.UserPermVal)
```

```
  case config.UPM_everyime:
    permFunc = lu_decompose.UPermEverytime
  }

  return lu_decompose.MakeLU(permFunc, A)
}

func main() {
  config.Init()

  A := utils.ReadRMatrix()
  LU := MakeLU(A)
  LU.Decompose()

  utils.PrintMatrix(LU.P)
  utils.PrintMatrix(LU.L)
  utils.PrintMatrix(LU.U)

  fmt.Println("Проверка --- PA ?= LU")
  utils.PrintMatrix(LU.P.MulByR(A.Copy()))
  utils.PrintMatrix(LU.L.MulByR(LU.U))

  // Нахождение обратной матирцы AX = E
  fmt.Println("Нахождение обратной матрицы")
  n, _ := A.Shape()
  E := matrix.MakeRealMatrix(n, n)
  for i := 0; i < n; i++ {
    E.SetEl(i, i, 1)
  }
  invA := LU.Solve(E)
  utils.PrintMatrix(invA)
  fmt.Println("Проверка --- A*invA ?= E")
  utils.PrintMatrix(A.MulByR(invA))
  utils.PrintMatrix(E)

  fmt.Println("Решение СЛАУ")
  // Решение СЛАУ
  b := utils.ReadRMatrix()
  x := LU.Solve(b)
  utils.PrintEq(A, x, b)
  utils.PrintMatrix(x)
}
```

internal/decompositor.go:

```
package lu_decompose

import (
  "github.com/Reterer/number_methods/pkg/matrix"
)
```

```
type LU struct {
  perm       Permutator
  P          *matrix.PMatrix
  L          *matrix.RMatrix
  U          *matrix.RMatrix
  n          int
  decomposed bool
}

func MakeLU(perm Permutator, A *matrix.RMatrix) *LU {
  n, m := A.Shape()
  if n != m {
    panic("n != m")
  }
  dec := LU{
    perm: perm,
    P:    matrix.MakePermutationMatrix(n),
    L:    matrix.MakeRealMatrix(n, n),
    U:    matrix.MakeRealMatrix(n, n),
    n:    n,
  }

  for i := 0; i < n; i++ {
    dec.L.SetEl(i, i, 1)
    copy(dec.U.GetCol(i), A.GetCol(i))
  }

  return &dec
}

func (lu *LU) Decompose() {
  if lu.decomposed {
    return
  }

  niter := lu.n - 1
  for i := 0; i < niter; i++ {
    // Сначала делаем permutation
    if lu.perm != nil {
      lu.perm(lu, i)
    }

    // Затем обновляем L и U
    // Строка, ниже которой мы будем обнулять i столбец
    mainCol := lu.U.GetCol(i)
    if mainCol[i] == 0 {
      panic("Main element is eq 0")
    }
    // Для каждой более нижней строки
    for j := i + 1; j < lu.n; j++ {
      currCol := lu.U.GetCol(j)
```

```
      del := currCol[i] / mainCol[i]
      // update U
      for k := i; k < lu.n; k++ {
        currCol[k] -= del * mainCol[k]
      }
      // update L
      lu.L.SetEl(j, i, del)
    }
  }

  lu.decomposed = true
}

func (lu *LU) Solve(b *matrix.RMatrix) *matrix.RMatrix {
  if !lu.decomposed {
    return nil
  }
  n, m := b.Shape()
  if !(n == lu.n) {
    return nil
  }

  nB := lu.P.MulByR(b.Copy())

  z := matrix.MakeRealMatrix(n, m)
  for k := 0; k < m; k++ {
    z.SetEl(0, k, nB.GetEl(0, k))
    for i := 1; i < n; i++ {
      var sum float64
      for j := 0; j < i; j++ {
        sum += lu.L.GetEl(i, j) * z.GetEl(j, k)
      }
      z.SetEl(i, k, nB.GetEl(i, k)-sum)
    }
  }
  x := matrix.MakeRealMatrix(n, m)
  for k := 0; k < m; k++ {
    x.SetEl(n-1, k, z.GetEl(n-1, k)/lu.U.GetEl(n-1, n-1))
    for i := n - 2; i >= 0; i-- {
      var sum float64
      for j := i + 1; j < n; j++ {
        sum += lu.U.GetEl(i, j) * x.GetEl(j, k)
      }
      x.SetEl(i, k, (z.GetEl(i, k)-sum)/lu.U.GetEl(i, i))
    }
  }

  return x
}
```

internal/permutation_func.go:

```go
package lu_decompose

import (
  "math"
)

type Permutator func(lu *LU, iter int)

func PermDefault(lu *LU, iter int, j int) {

  // Обновляем матрицы
  lu.P.SwapCol(iter, j)
  lu.U.SwapCol(iter, j)

  lu.L.SwapCol(iter, j)

  lu.L.SetEl(iter, j, 0)
  lu.L.SetEl(iter, iter, 1)
  lu.L.SetEl(j, iter, 0)
  lu.L.SetEl(j, j, 1)
}

func PermMin(lu *LU, i int) {
  if lu.U.GetEl(i, i) != 0 {
    return
  }
  maxJ := i
  valueJ := lu.U.GetEl(i, i)
  for j := i; j < lu.n; j++ {
    if math.Abs(lu.U.GetEl(j, i)) > math.Abs(valueJ) {
      maxJ = j
      valueJ = lu.U.GetEl(j, i)
    }
  }
  if valueJ != 0 {
    PermDefault(lu, i, maxJ)
    return
  }
  panic("эту матрицу нельзя разложить в LU")
}

func PermEveryIteration(lu *LU, i int) {
  maxJ := i
  maxV := lu.U.GetEl(i, i)
  for j := i; j < lu.n; j++ {
    el := lu.U.GetEl(j, i)
    if math.Abs(el) < math.Abs(maxV) {
      maxV = el
      maxJ = j
    }
  }
```

```
  if maxV != 0 {
    PermDefault(lu, i, maxJ)
    return
  }
  panic("эту матрицу нельзя разложить в LU")
}

// user permutation
func UPermOnceMake(perm Permutator, col int) Permutator {
  return func(lu *LU, i int) {
    if i == 0 {
      PermDefault(lu, i, col)
    } else {
      perm(lu, i)
      lu.perm = perm
    }
  }
}

func UPermEverytime(dec *LU, i int) {
  // TODO
  panic("Не определена")
}
```

Вывод программы

```
4 4
-2 -9 -3 7
-7 8 2 5
-6 2 0 0
0 -3 8 -3
Нахождение обратной матрицы
4 4
 -0.0192   0.0196  -0.1832  -0.0121
 -0.0576   0.0589  -0.0495  -0.0363
  0.0025   0.0627  -0.0739   0.1103
  0.0643   0.1082  -0.1477  -0.0029
Решение СЛАУ
4 1
-26
-25
```

```
-16
-5
4 1
  3.0000
  1.0000
 -1.0000
 -2.0000
```

Задание 1.2

Реализовать метод прогонки в виде программы, задавая в качестве входных данных ненулевые элементы матрицы системы и вектор правых частей. Используя разработанное программное обеспечение, решить СЛАУ с трехдиагональной матрицей.

Вариант:

$$\begin{cases} -12 \cdot x_1 - 7 \cdot x_2 = -102 \\ -7 \cdot x_1 - 11 \cdot x_2 - 3 \cdot x_3 = -92 \\ -7 \cdot x_2 + 21 \cdot x_3 - 8 \cdot x_4 = -65 \\ 4 \cdot x_3 - 13 \cdot x_4 + 5 \cdot x_5 = 38 \\ -6 \cdot x_4 + 14 \cdot x_5 = -12 \end{cases}$$

Листинг

main.go:

```go
package main

import (
  "fmt"

  "github.com/Reterer/number_methods/internal/run_through"
  "github.com/Reterer/number_methods/internal/utils"
  "github.com/Reterer/number_methods/pkg/matrix"
)

func readThreeDiagMatrix() *matrix.RMatrix {
  var n, m int
  if _, err := fmt.Scan(&n, &m); err != nil {
    panic("can't read matrix shape")
  }

  mat := matrix.MakeRealMatrix(n, m)

  for i := 0; i < n; i++ {
    col := mat.GetCol(i)
    for k := 0; k < 3; k++ {
      j := -1 + k + i
      if j < 0 || j >= m {
        continue
      }

      if _, err := fmt.Scan(&col[j]); err != nil {
```

```
        panic("can't read element")
      }
    }
  }

  return mat
}

func main() {
  A := readThreeDiagMatrix()
  b := utils.ReadRMatrix()

  x := run_through.Do(A, b)
  utils.PrintMatrix(x)
  utils.PrintEq(A, x, b)
}
```

internal/run_through.go:

```
package run_through

import "github.com/Reterer/number_methods/pkg/matrix"

func Do(A *matrix.RMatrix, B *matrix.RMatrix) *matrix.RMatrix {
  n, m := A.Shape()
  nn, mm := B.Shape()
  if n != m && n > 0 {
    return nil
  } else if mm != 1 {
    return nil
  } else if nn != n {
    return nil
  }

  P := make([]float64, n+1)
  Q := make([]float64, n+1)

  getElFromCol := func(col []float64, j int) float64 {
    if j < 0 || j >= m {
      return 0
    }
    return col[j]
  }

  for i := 0; i < n; i++ {
    col := A.GetCol(i)
    a := getElFromCol(col, i-1)
    b := getElFromCol(col, i)
    c := getElFromCol(col, i+1)
    d := B.GetEl(i, 0)
    P[i+1] = -c / (b + a*P[i])
```

```
    Q[i+1] = (d - a*Q[i]) / (b + a*P[i])
  }

  x := matrix.MakeRealMatrix(n, 1)
  prevX := float64(0)
  for i := n - 1; i >= 0; i-- {
    x.SetEl(i, 0, P[i+1]*prevX+Q[i+1])
    prevX = x.GetEl(i, 0)
  }

  return x
}
```

Вывод программы

```
5 5
-12 -7
-7 -11 -3
-7 21 -8
4 -13 5
-6 14
5 1
-102
-92
-65
38
-12
Ответ:
5 1
  5.0000
  6.0000
 -3.0000
 -5.0000
 -3.0000
```

Задание 1.3

Реализовать метод простых итераций и метод Зейделя в виде программ, задавая в качестве входных данных матрицу системы, вектор правых частей и точность вычислений. Используя разработанное программное обеспечение, решить СЛАУ. Проанализировать количество итераций, необходимое для достижения заданной точности.

Вариант:

$$\begin{cases} 18 \cdot x_1 - 2 \cdot x_2 + 7 \cdot x_4 = 50 \\ -x_1 + 14 \cdot x_2 - 3 \cdot x_3 + 2 \cdot x_4 = 2 \\ 5 \cdot x_1 + 5 \cdot x_2 + 26 \cdot x_3 + 7 \cdot x_4 = 273 \\ -2 \cdot x_1 - 6 \cdot x_2 + 9 \cdot x_3 + 24 \cdot x_4 = 111 \end{cases}$$

Листинг

main.go:

```
package main

import (
  "fmt"
  "math"

  "github.com/Reterer/number_methods/internal/utils"
  "github.com/Reterer/number_methods/pkg/matrix"
)

func prepareA(A *matrix.RMatrix) {
  n, _ := A.Shape()
  for i := 0; i < n; i++ {
    if A.GetEl(i, i) == 0 {
      for j := i + 1; j < n; j++ {
        if A.GetEl(j, i) != 0 {
          A.SwapCol(i, j)
          return
        }
      }
      panic("Я не смог поменять местами строки так, что бы на главной диагонали не было нулей")
    }

  }
}

func doIteration(A, b *matrix.RMatrix, eps float64) *matrix.RMatrix {
```

```
  n, m := A.Shape()
  nn, mm := b.Shape()
  if n != m && n > 0 {
    return nil
  } else if mm != 1 {
    return nil
  } else if nn != n {
    return nil
  }

  prepareA(A)
  beta := matrix.MakeRealMatrix(n, 1)
  alpha := matrix.MakeRealMatrix(n, m)

  for i := 0; i < n; i++ {
    aCol := A.GetCol(i)
    aii := aCol[i]
    alphaCol := alpha.GetCol(i)
    if aii == 0 {
      return nil
    }
    beta.SetEl(i, 0, b.GetEl(i, 0)/aii)

    for j := 0; j < m; j++ {
      if j == i {
        continue
      }
      alphaCol[j] = -aCol[j] / aii
    }
  }

  x := beta.Copy()
  norm := calcNorm(x)
  for iter := 0; norm > eps; iter++ {
    nx := beta.Add(alpha.MulByR(x))
    norm = calcNorm(nx.Add(x.MulByConstant(-1)))
    x = nx
    fmt.Println(norm)
  }

  return x
}

func calcNorm(A *matrix.RMatrix) float64 {
  var norm float64

  n, m := A.Shape()
  for i := 0; i < n; i++ {
    colA := A.GetCol(i)
    for j := 0; j < m; j++ {
      norm += colA[j] * colA[j]
```

```
    }
  }
  norm = math.Sqrt(norm)

  return norm
}

func doZeidel(A, b *matrix.RMatrix, eps float64) *matrix.RMatrix {
  n, m := A.Shape()
  nn, mm := b.Shape()
  if n != m && n > 0 {
    return nil
  } else if mm != 1 {
    return nil
  } else if nn != n {
    return nil
  }

  prepareA(A)
  beta := matrix.MakeRealMatrix(n, 1)
  alpha := matrix.MakeRealMatrix(n, m)

  for i := 0; i < n; i++ {
    aCol := A.GetCol(i)
    aii := aCol[i]
    alphaCol := alpha.GetCol(i)
    if aii == 0 {
      return nil
    }
    beta.SetEl(i, 0, b.GetEl(i, 0)/aii)

    for j := 0; j < m; j++ {
      if j == i {
        continue
      }
      alphaCol[j] = -aCol[j] / aii
    }
  }

  x := beta.Copy()
  norm := calcNorm(x)

  for iter := 0; norm > eps; iter++ {
    // Придется работать вручную
    norm = 0
    for i := 0; i < n; i++ {
      alphaCol := alpha.GetCol(i)
      var summ float64
      for j := 0; j < m; j++ {
        summ += x.GetEl(j, 0) * alphaCol[j]
      }
```

```
      prev := x.GetEl(i, 0)
      x.SetEl(i, 0, summ+beta.GetEl(i, 0))
      norm += math.Pow(prev-x.GetEl(i, 0), 2)
    }
    norm = math.Sqrt(norm)
    fmt.Println(norm)

  }

  return x
}

func main() {
  var eps float64
  fmt.Scan(&eps)

  A := utils.ReadRMatrix()
  b := utils.ReadRMatrix()

  x := doIteration(A, b, eps)
  fmt.Println("Метод итераций: ")
  utils.PrintMatrix(x)
  x = doZeidel(A, b, eps)
  fmt.Println("Метод Зейделя")
  utils.PrintMatrix(x)
}
```

Вывод программы

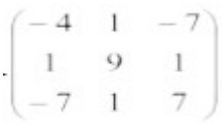
```
0.001
4 4
18 0 -2 7
-1 14 -3 2
5 5 26 7
-2 -6 9 24
4 1
50
2
273
111
Метод итераций:
4 1
```

```
  3.0001
  2.0000
  8.9999
  2.0003
Метод Зейделя
4 1
  3.0000
  2.0000
  9.0000
  2.0000
```

Задание 1.4

Реализовать метод вращений в виде программы, задавая в качестве входных данных матрицу и точность вычислений. Используя разработанное программное обеспечение, найти собственные значения и собственные векторы симметрических матриц. Проанализировать зависимость погрешности вычислений от числа итераций.

Вариант:



Листинг

main.go:

```
package main

import (
  "fmt"
  "math"

  "github.com/Reterer/number_methods/internal/utils"
  "github.com/Reterer/number_methods/pkg/matrix"
)

func accYakobi(A *matrix.RMatrix) float64 {
  acc := float64(0)
  n, m := A.Shape()
  for i := 0; i < n; i++ {
    colA := A.GetCol(i)
    for j := i + 1; j < m; j++ {
      acc += math.Pow(colA[j], 2)
    }
  }
  acc = math.Sqrt(acc)
  return acc
}

func doYakobi(A *matrix.RMatrix, eps float64) (l []float64, x *matrix.RMatrix) {

  n, m := A.Shape()
  x = matrix.MakeRealMatrix(n, m)
  for i := 0; i < m; i++ {
    x.SetEl(i, i, 1)
```

```
}

currEps := accYakobi(A)
for iter := 0; currEps > eps; iter++ {
  fmt.Println(currEps)
  utils.PrintMatrix(x)
  utils.PrintMatrix(A)
  fmt.Println("--------------\t", iter)

  var maxI, maxJ int = 0, 1
  var maxV float64
  for i := 0; i < n; i++ {
    aCol := A.GetCol(i)
    for j := i + 1; j < m; j++ {
      if math.Abs(aCol[j]) > maxV {
        maxV = math.Abs(aCol[j])
        maxI = i
        maxJ = j
      }
    }
  }

  Ui := matrix.MakeRealMatrix(n, m)
  UiT := matrix.MakeRealMatrix(m, n)
  for i := 0; i < n; i++ {
    if i == maxI || i == maxJ {
      continue
    }
    Ui.SetEl(i, i, 1)
    UiT.SetEl(i, i, 1)
  }
  // Расчет углов
  aij := A.GetEl(maxI, maxJ)
  aii := A.GetEl(maxI, maxI)
  ajj := A.GetEl(maxJ, maxJ)
  theta := math.Pi / 4
  if aii != ajj {
    theta = math.Atan((2*aij)/(aii-ajj)) / 2
  }
  Ui.SetEl(maxI, maxI, math.Cos(theta))
  Ui.SetEl(maxI, maxJ, -math.Sin(theta))
  Ui.SetEl(maxJ, maxI, math.Sin(theta))
  Ui.SetEl(maxJ, maxJ, math.Cos(theta))

  UiT.SetEl(maxI, maxI, math.Cos(theta))
  UiT.SetEl(maxI, maxJ, math.Sin(theta))
  UiT.SetEl(maxJ, maxI, -math.Sin(theta))
  UiT.SetEl(maxJ, maxJ, math.Cos(theta))

  // Применения
  x = x.MulByR(Ui)
```

```
    A = UiT.MulByR(A.MulByR(Ui))

    currEps = accYakobi(A)
  }

  l = make([]float64, n)
  for i := 0; i < n; i++ {
    l[i] = A.GetEl(i, i)
  }

  return l, x
}

func main() {
  var eps float64
  fmt.Scan(&eps)
  A := utils.ReadRMatrix()

  l, x := doYakobi(A, eps)
  fmt.Println(l)
  utils.PrintMatrix(x)
}
```

Вывод программы

```
0.001
3 3
-4 1 -7
1 9 1
-7 1 7
[-7.510517884559564 8.960914230314827 10.549603654244734]
3 3
  0.8955   0.2028  -0.3960
 -0.0807   0.9494   0.3037
  0.4376  -0.2400   0.8666
```

Задание 1.5

Реализовать алгоритм QR – разложения матриц в виде программы. На его основе разработать программу, реализующую QR – алгоритм решения полной проблемы собственных значений произвольных матриц, задавая в качестве входных данных матрицу и точность вычислений. С использованием разработанного программного обеспечения найти собственные значения матрицы.

Вариант:

$$\begin{pmatrix} -9 & -9 & -3 \\ -9 & 0 & -2 \\ -5 & -1 & -4 \end{pmatrix}$$

Листинг

main.go:

```go
package main

import (
  "fmt"
  "math"
  "math/cmplx"

  "github.com/Reterer/number_methods/internal/utils"
  "github.com/Reterer/number_methods/pkg/matrix"
)

func sign(num float64) float64 {
  if num == 0 {
    return 0
  } else if num < 0 {
    return -1
  } else {
    return 1
  }
}

func doQR(A *matrix.RMatrix) (Q, R *matrix.RMatrix) {
  n, m := A.Shape()

  Q = matrix.MakeRealMatrix(n, m)
  for i := 0; i < n; i++ {
    Q.SetEl(i, i, 1)
  }
```

```
  for i := 0; i < n-1; i++ {
    var norm float64
    for j := i; j < m; j++ {
      norm += math.Pow(A.GetEl(j, i), 2)
    }
    norm = math.Pow(norm, 0.5)

    v := matrix.MakeRealMatrix(n, 1)
    vT := matrix.MakeRealMatrix(1, n)
    v.SetEl(i, 0, A.GetEl(i, i)+sign(A.GetEl(i, i))*norm)
    vT.SetEl(0, i, A.GetEl(i, i)+sign(A.GetEl(i, i))*norm)
    for j := i + 1; j < m; j++ {
      v.SetEl(j, 0, A.GetEl(j, i))
      vT.SetEl(0, j, A.GetEl(j, i))
    }

    H := matrix.MakeRealMatrix(n, m)
    for i := 0; i < n; i++ {
      H.SetEl(i, i, 1)
    }
    c := -2 / vT.MulByR(v).GetEl(0, 0)
    H = H.Add(v.MulByR(vT).MulByConstant(c))
    Q = Q.MulByR(H)
    A = H.MulByR(A)

  }
  R = A

  return Q, R
}

func calcL(A *matrix.RMatrix, i int) (l_0, l_1 complex128) {
  aii := A.GetEl(i, i)
  aioi := A.GetEl(i+1, i)
  aiio := A.GetEl(i, i+1)
  aioio := A.GetEl(i+1, i+1)

  a := complex(1, 0)
  b := complex(aii+aioio, 0)
  c := complex(aii*aioio-aiio*aioi, 0)

  d := cmplx.Pow(b*b-4*a*c, 0.5)
  l_0 = (-b - d) / (2 * a)
  l_1 = (-b + d) / (2 * a)

  return l_0, l_1
}

func getL(A *matrix.RMatrix, pl []complex128, eps float64) ([]complex128, bool) {
  ok := true
```

```
  n, _ := A.Shape()
  l := make([]complex128, n)
  for i := 0; i < n; i++ {
    // Комплексно сопряжонный
    if i+1 < n && math.Abs(A.GetEl(i+1, i)) > eps {
      l_0, l_1 := calcL(A, i)

      l[i] = l_0
      l[i+1] = l_1

      ok = ok && (cmplx.Abs(l[i]-pl[i]) < eps && cmplx.Abs(l[i+1]-pl[i+1]) < eps)
      fmt.Println(ok, l[i]-pl[i], l[i+1]-pl[i+1])
      i++

    } else {
      l[i] = complex(A.GetEl(i, i), 0)
      ok = ok && (cmplx.Abs(l[i]-pl[i]) < eps)
      fmt.Println(ok, l[i]-pl[i])
    }
  }
  return l, ok
}

func main() {
  var eps float64
  fmt.Scan(&eps)

  A := utils.ReadRMatrix()
  n, _ := A.Shape()
  l := make([]complex128, n)
  isRun := true
  for i := 0; isRun; i++ {
    Q, R := doQR(A)
    A = R.MulByR(Q)

    var ok bool
    l, ok = getL(A, l, eps)
    isRun = !ok
    fmt.Println(l, isRun)

    fmt.Println("------- it:", i)
    utils.PrintMatrix(A)
  }

  fmt.Println(l)
}
```

Вывод программы

```
0.001
3 3
-9 -9 -3
-9 0 -2
-5 -1 -4
Ответ:
3 3
-15.9965   0.9235  -1.0076
  0.0001   5.5368   1.7701
  0.0000   0.0003  -2.5403
[(-15.996545203784159+0i) (5.536810412737502+0i) (-2.540265208953363+0i)]
```