Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ЧИСЕЛ И СОБСТВЕННЫХ ВЕКТОРОВ

Отчет по лабораторной работе №3 по дисциплине
«Численные методы»

Выполнил:

Обучающийся гр. ______439-3______
(группа)

________________ __А. С. Мазовец__
(подпись) *(И. О. Фамилия)*

« _____ » ___________________ 2021г.
(дата)

Проверил:

______ассистент кафедры АСУ______
(должность, ученая степень, звание)

________________ ___А. Е. Косова___
(подпись) *(И. О. Фамилия)*

________ « _____ » ___________________ 2021г.
(оценка) *(дата)*

Томск 2021

1 Введение

1.1 Цели

При помощи метода Данилевского необходимо реализовать поиск собственных чисел и векторов, а также найти их кратность.

1.2 Входные данные

m – тип задачи (1 – поиск собственных чисел, 2 – векторов);

n – порядок матрицы;

```
1| a11..a1n
2| a21..a2n
3| .  .   . .
4| an1..ann
```

1.3 Выходные данные

P – матрица Фробениуса;

λi – i-е собственное число;

$|A-\lambda iE|$ – проверка i-го собственного числа (при m=1);

xi – i-й собственный вектор (при m = 2);

$Axi-\lambda ixi$ – проверка i-го собственного вектора (при m = 2);

ki – кратность i-го собственного числа/вектора;

2 Теория

2.1 Определение

Собственным числом матрицы А называется такое число λ, которое обращает в ноль определитель:

$$|A-\lambda E|=0 \quad (2.1)$$

где Е – единичная матрица.

Собственным вектором матрицы А называется такой ненулевой вектор χ, что выполняется:

$$Ax=\lambda x \quad (2.2)$$

У квадратной матрицы размерности n имеется n собственных чисел и векторов. Некоторые из них могут быть кратными. Таким образом, квадратная матрица размерности n имеет m различных собственных чисел λi и соответствующих им собственных векторов xi кратности ki . При этом:

$$\sum_{i=1}^{m} k_i=n, i=1,2\ldots n, 1\leq m\leq n. \quad (2.3)$$

2.2 Метод Данилевского

Суть его состоит в том, что исходная матрица А преобразуется в подобную ей матрицу Фробениуса Р, имеющую следующий вид:

$$\begin{pmatrix} p1 & p2 & \cdots & p3 & p4 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \quad (2.4)$$

Делается это при помощи следующего преобразования подобия:

$$P=S^{-1}AS \quad (2.5)$$

где $S=M_{n-1}M_{n-2}\ldots M_1, S^{-1}=M_{n-1}^{-1}M_{n-2}^{-1}\ldots M_1^{-1}$

Матрицы М строятся следующим образом:

$$M_k:\begin{cases} m_{ij}=e_{ij}, i=1,2\ldots n, j=1,2\ldots n, i\neq k; \\ m_{kj}=\dfrac{-a_{k+1,j}^{n-k-1}}{a_{k+1,k}^{n-k-1}}, j=1,2\ldots n, j\neq k; \\ m_{kk}=\dfrac{1}{a_{k+1,j}^{n-k-1}}. \end{cases} \quad (2.5)$$

$$M_k^{-1}:\begin{cases} m_{ij}=e_{ij}, i=1,2\ldots n\; j=1,2\ldots n, i\neq k; \\ m_{kj}=a_{k+1,j}^{n-k-1}, j=1,2\ldots n. \end{cases} \quad (2.6)$$

Далее для матрицы P строится характеристический полином:

$$D(\lambda)=det(P-\lambda E)=(-1)^n(\lambda^n-p_1\lambda^{n-1}-p_2\lambda^{n-2}-\ldots-p_n) \quad (2.7)$$

Это полином имеет n корней λ1, λ2, …, λn. Некоторые из них могут быть кратными, при этом выполняется соотношение. Необходимо не только найти все корни полинома, но и определить их кратность.

2.3 Вычисление собственных векторов

Далее для каждого собственного числа вычисляется соответствующий ему собственный вектор. Если yi – это собственный вектор матрицы P, соответствующий собственному числу λi, то

$$x_i=S_{yi}, i=1,2\ldots n. \quad (2.8)$$

При этом собственный вектор матрицы P выглядит следующим образом:

$$Y_i=\begin{pmatrix} \lambda_i^{n-1} \\ \lambda_i^{n-2} \\ \ldots \\ \lambda_i \\ 1 \end{pmatrix} \quad (2.9)$$

2.4 Определение кратности собственных чисел и векторов

Корень уравнения ξ имеет кратность k, если не только функция в точке ξ принимает нулевое значение, но и k–1 ее производных:

$$f^{(i)}(\xi)=0, i=0,1,2\ldots k-1 \quad (2.10)$$

При i = 0 имеем саму функцию. Таким образом, получаем к нулей функции и ее производных.

3 Результат работы

Пример выполнения программы для матрицы:

$$A = \begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & 1.6 & 2 \end{pmatrix} \quad (3.1)$$

Входные данные:

```
⎡    2.200,   1.000,   0.500,   2.000 ⎤
⎢    1.000,   1.300,   2.000,   1.000 ⎥
⎢    0.500,   2.000,   0.500,   1.600 ⎥
⎣    2.000,   1.000,   1.600,   2.000 ⎦
```

Вывод программы:

```
⎡    2.200,   1.000,   0.500,   2.000 ⎤
⎢    1.000,   1.300,   2.000,   1.000 ⎥
⎢    0.500,   2.000,   0.500,   1.600 ⎥
⎣    2.000,   1.000,   1.600,   2.000 ⎦

⎡    1.575,   0.688,   0.312,   1.375 ⎤
⎢   -1.500,   0.050,   1.250,  -1.500 ⎥
⎢    1.450,   4.125,   4.375,   2.810 ⎥
⎣    0.000,   0.000,   1.000,   0.000 ⎦

⎡    1.333,   0.167,  -0.417,   0.907 ⎤
⎢   -4.327,   4.667,   7.143,  -5.013 ⎥
⎢    0.000,   1.000,   0.000,   0.000 ⎥
⎣    0.000,   0.000,   1.000,   0.000 ⎦

⎡    6.000,   0.200, -12.735,   2.762 ⎤
⎢    1.000,   0.000,   0.000,  -0.000 ⎥
⎢    0.000,   1.000,   0.000,   0.000 ⎥
⎣    0.000,   0.000,   1.000,   0.000 ⎦

Matrix S
⎡   -0.231,   1.079,   1.651,  -1.159 ⎤
⎢    0.081,  -0.137,  -1.641,  -0.274 ⎥
⎢    0.238,  -1.263,  -0.413,   0.370 ⎥
⎣    0.000,   0.000,   0.000,   1.000 ⎦

Frobenius matrix P
⎡    6.000,   0.200, -12.735,   2.762 ⎤
⎢    1.000,  -0.000,  -0.000,  -0.000 ⎥
⎢    0.000,   1.000,  -0.000,   0.000 ⎥
⎣    0.000,  -0.000,   1.000,   0.000 ⎦

λ1          =  -1.420
|A - λ1E|   = -5.491E-5
```

```
x1           = [  -0.666,   1.548,  -2.272,   1.000 ]
Ax1 - λ1x1  = [ -1.269E-5, 4.462E-6, 1.308E-5, 0.000E0 ]
k1           = 1

λ2           =   0.223
|A - λ2E|    = 4.805E-5
x2           = [  -0.740,  -0.645,   0.218,   1.000 ]
Ax2 - λ2x2  = [ 1.111E-5, -3.904E-6, -1.144E-5, 0.000E0 ]
k2           = 1

λ3           =   1.545
|A - λ3E|    = -2.682E-5
x3           = [   3.116,  -2.837,  -2.406,   1.000 ]
Ax3 - λ3x3  = [ -6.199E-6, 2.179E-6, 6.387E-6, 4.441E-16 ]
k3           = 1

λ4           =   5.652
|A - λ4E|    = -3.677E-4
x4           = [   0.897,   0.753,   0.690,   1.000 ]
Ax4 - λ4x4  = [ -8.498E-5, 2.987E-5, 8.755E-5, -6.217E-15 ]
k4           = 1
```

3.1 Проверка

Вычислим собственные значения матрицы A используя формулы 2.1 — 2.7.

1-й этап:

$$A=\begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & \underline{1.6} & 2 \end{pmatrix}$$

$$S_1=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-2}{1.6} & \frac{-1}{1.6} & \frac{1}{1.6} & \frac{2}{1.6} \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1.25 & -0.625 & 0.625 & -1.25 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S_1^{-1}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 1 & 1.6 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2-й этап:

$$P_1=S_1^{-1}\cdot A\cdot S_1=\begin{pmatrix} 1.575 & 0.6875 & 0.3125 & 1.375 \\ -1.5 & 0.05 & 1.25 & -1.5 \\ 1.45 & \underline{4.125} & 4.375 & 2.81 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$S_2=\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{-1.45}{4.125} & \frac{1}{4.125} & \frac{-4.375}{4.125} & \frac{-2.81}{4.125} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ -0.3515 & 0.2424 & -1.0606 & -0.6812 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S_2^{-1}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1.45 & 4.125 & 4.375 & 2.81 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3-й этап:

$$P_2=S_2^{-1}\cdot P_1\cdot S_2=\begin{pmatrix} 1.3333 & 0.1667 & -0.4167 & 0.9067 \\ \underline{-4.3267} & 4.6667 & 7.1433 & -5.0133 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$s_3=\begin{pmatrix} \frac{1}{-4.3267} & \frac{-4.6667}{-4.3267} & \frac{-7.1433}{-4.3267} & \frac{5.0133}{-4.3267} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} -0.2311 & 1.0786 & 1.651 & -1.1587 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S_3^{-1}=\begin{pmatrix} -4.3267 & 4.6667 & 7.1433 & -5.0133 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица Фробениуса:

$$P=S_3^{-1}\cdot P_2\cdot S_2=\begin{pmatrix} 6 & 0.2 & -12.735 & 2.7616 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Полином характеристического уравнения матрицы А:

$$\lambda^4-6\lambda^3-0.2\lambda^2+12.735\lambda-2.7616=0$$

Корни полинома:

$\lambda_1=-1.4201$ $\lambda_2=0.2226$ $\lambda_3=1.5454$ $\lambda_4=5.652$

Найдем собственные векторы матрицы А:

$$S=S_1\cdot S_2\cdot S_3=\begin{pmatrix} -0.2311 & 1.0786 & 1.651 & -1.1587 \\ 0.0812 & -0.1367 & -1.641 & -0.2739 \\ 0.2381 & -1.2628 & -0.4132 & 0.3696 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$x_1 = S \cdot y_1 = S \cdot \begin{pmatrix} -2.8638 \\ 2.0166 \\ -1.4201 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.6663 \\ 1.548 \\ -2.2723 \\ 1 \end{pmatrix} \quad x_2 = S \cdot y_2 = S \cdot \begin{pmatrix} 0.011 \\ 0.0496 \\ 0.2226 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.7402 \\ -0.6451 \\ 0.2176 \\ 1 \end{pmatrix}$$

$$x_3 = S \cdot y_3 = S \cdot \begin{pmatrix} 3.691 \\ 2.3883 \\ 1.5454 \\ 1 \end{pmatrix} = \begin{pmatrix} 3.1157 \\ -2.8365 \\ -2.4059 \\ 1 \end{pmatrix} \quad x_4 = S \cdot y_4 = S \cdot \begin{pmatrix} 180.5568 \\ 31.9455 \\ 5.652 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.8975 \\ 0.7531 \\ 0.69 \\ 1 \end{pmatrix}$$

4 Вывод

Реализовал и освоил метод Данилевского для нахождения собственных значений и собственных векторов матрицы. Сделал проверку полученных результатов.

5 Листинг программы

5.1 Модуль Methods.hs

```haskell
module Methods where

import Control.Applicative
import Control.Monad
import Prelude hiding ( break )
import Data.List hiding ( break )

type Matrix = [ Vector ]
type Vector = [ Double ]


matrixE :: Matrix
matrixE = do
    prior <- inits $ repeat 0
    break <- return $ repeat 0
    return $ prior ++ 1 : break

mulVector :: Matrix -> Vector -> Vector
mulVector a b = join $ mulMatrix a (transpose [b])

mulScal :: Matrix -> Double -> Matrix
mulScal a b = map (map (*b)) a

mulVector' :: Vector -> Matrix -> Vector
mulVector' a b = join $ mulMatrix (transpose [a]) b

mulMatrix :: Matrix -> Matrix -> Matrix
mulMatrix a b = do
```

```
    a' <- a
    b' <- return $ map (sum . zipWith (*) a') $ transpose b
    return b'

subMatrix :: Matrix -> Matrix -> Matrix
subMatrix = zipWith (zipWith (-))

subVector :: Vector -> Vector -> Vector
subVector = zipWith (-)


gershgorin :: Matrix -> (Double, Double)
gershgorin m = (\e -> (minimum e, maximum e)) . concat $ do
    l <- zipWith (\l i -> (l, i)) m [0..]
    let center (l', i') = l' !! i'
    let low_c  (l', i') = sum $ take i' l'
    let high_c (l', i') = sum $ drop (i' + 1) l'
    let rad    l'       = sum $ sequence [high_c, low_c] l'
    return $ sequence [(+) <$> center <*> rad, (-) <$> center <*> rad] l


self :: Matrix -> Double -> Vector
self m eps = map (\x -> (sum x) / (fromIntegral $ length x)) $
    groupBy (\a b -> abs (a - b) <= (eps * 100)) $
    filter (>= (fst $ gershgorin m) - 0.5) $ do
    let selfV = 1 : map (*(-1)) (head $ frobenius m)
     lamda <- [(fst $ gershgorin m), (fst $ gershgorin m) + eps .. (snd $ gershgorin m)]
     let root = sum $ zipWith (*) (iterate (*lamda) 1) (reverse selfV)
     return $ if (abs root) <= (eps * 100) then lamda else ((fst $ gershgorin m) - 1)


selfVec :: Matrix -> Double -> [Vector]
selfVec m eps = selfVec' $ map (\x -> take (length $ self m eps) $ iterate (*x) 1) (self m eps)
  where
    selfVec' vs = do
        v <- vs
        return $ mulVector (matrixS m) (reverse v)

reductions :: Matrix -> Double -> [Int]
reductions m eps = map red $ self m eps
  where
    red = length . filter ((<= eps * 100) . abs) . decodered
    selfV = reverse $ 1 : map (*(-1)) (head $ frobenius m)
     kd     = take (length selfV) $ zipWith (\a b -> (a, b)) (repeat 1) (iterate (+1) 0) :: [(Double, Double)]
    stepd kd' = do
        (k, d) <- kd'
```

```
        return $ if d == 0 then (k*d, 0) else (k*d, d-1)
    kds   = take (length selfV) $ iterate stepd kd
    decodekd  l = map (map (\(k, d) -> k * (l ** d))) kds
    decodered l = map (sum . zipWith (*) selfV) (decodekd l)


frobenius :: Matrix -> Matrix
frobenius a = (matrixS' a) `mulMatrix` a `mulMatrix` (matrixS a)


convsA :: Matrix -> [Matrix]
convsA = scanl step <*> liftA2 zip convsM' convsM
  where
    step a0 (m', m) = m' `mulMatrix` a0 `mulMatrix` m


-- | Matrix M{k}
convsM  :: Matrix -> [Matrix]
convsM  a = zipWith3 (\p j b -> p ++ j : b) priorsE mkjs breaksE
  where
    -- m[i,j]
    priorsE = drop 2 . reverse $ inits matrixE'
    breaksE = tail . reverse $ tails matrixE'

    -- m[k,j]
    mkjs = zipWith3 (\p j b -> (negate . (*j) <$> p) ++ j : (negate . (*j) <$> b))
        priorsMkj mkks breaksMkj
      where
          priorsMkj = map reverse . zipWith drop [2..] $ reverse <$> ak1s
          breaksMkj = map reverse . zipWith take [1..] $ reverse <$> ak1s

    -- m[k,k]
    mkks = map ((1/) . head) . zipWith drop [1..] $ reverse <$> ak1s
    ak1s = map head . zipWith drop [0..] $ reverse <$> convsA a

    matrixE' = take (length a) <$> take (length a) matrixE


-- | Matrix M{k} inversed
convsM' :: Matrix -> [Matrix]
convsM' a = zipWith3 (\p j b -> p ++ j : b) priorsE mkjs breaksE
  where
    -- m[i,j]
    priorsE = drop 2 . reverse $ inits matrixE'
    breaksE = tail . reverse $ tails matrixE'
```

```

    -- m[k,j]
    mkjs = map head . zipWith drop [0..] $ reverse <$> convsA a

    matrixE' = take (length a) <$> take (length a) matrixE


matrixS  :: Matrix -> Matrix
matrixS  = foldl1 (\acc x -> acc `mulMatrix` x) . convsM


matrixS' :: Matrix -> Matrix
matrixS' = foldl1 (\acc x -> acc `mulMatrix` x) . reverse . ConvsM'
```

5.2 Модуль Main.hs

```
import Methods
import qualified M
import Text.Printf
import Data.List hiding ( break )

main :: IO ()
main = do
      let m     = [[2.2,1,0.5,2],[1,1.3,2,1],[0.5,2,0.5,1.6],[2,1,1.6,2]]
    let eps = 1E-5
    putStrLn $ showMatrixIters "%7.3F" $ convsA m
    putStrLn "\nMatrix S"
    putStrLn $ showMatrix "%7.3F" $ matrixS m
    putStrLn "\nFrobenius matrix P"
    putStrLn $ showMatrix "%7.3F" $ frobenius m
    putStrLn ""

    putStrLn $ do
        (lamda, vector, koef, i) <- zipWith4 (\a b c d -> (a, b, c, d)) (self m eps) (selfVec m eps) (reductions m eps) (iterate (+1) (1 :: Int))
        printf "λ%d          = %7.3F" i lamda ++ printf "\n" ++
          printf "|A - λ%dE|    = %7.3E" i (M.det (subMatrix m (mulScal matrixE lamda))) ++ printf "\n" ++
           printf "x%d            = " i ++ showMatrix "%7.3F" [vector] ++ printf "\n" ++
             printf "Ax%d - λ%dx%d   = " i i i ++ showMatrix "%7.3E" [subVector (mulVector m vector) (map (*lamda) vector)] ++ printf "\n" ++
         printf "k%d          = %d" i koef ++ printf "\n" ++
         printf "\n"

    return ()

```

```

showMatrix :: String -> Matrix -> String
showMatrix _ []     = "[]"
showMatrix _ [[]]   = "[]"
showMatrix f (l:[]) = ("[ "++) . (++" ]") . intercalate ", " $ map (printf f) l
showMatrix f m      = showPrior ++ showTail ++ showBreak
  where
    showPrior = ("⎡ "++) . (++" ⎤\n") . intercalate ", " . map (printf f) $ head m
    showTail  = concat . map (("⎢ "++) . (++" ⎥\n") . intercalate ", " . map (printf f)) . init $ tail m
    showBreak = ("⎣ "++) . (++" ⎦") . intercalate ", " . map (printf f) $ last m


showMatrixIters :: String -> [Matrix] -> String
showMatrixIters f ms = intercalate "\n\n" $ map (showMatrix f) ms
```