Кирил Василев Голов, фак. № 62360,
СИ I курс, Група 5

Алгоритми за решение на дом. раб. №3 (УП).

Задача ②.

I. Алгебричен алгоритъм за решаване на матричното уравнение $XA = (-3)A^tC - 5X$.

Правим преобразувания на уравнението. Следните са еквивалентни:

1) $XA = (-3)A^tC - 5X$

2) $XA + 5X = (-3)A^tC$

3) $X(A + 5E) = (-3)A^tC$

4) Нека $A + 5E = S$ и $(-3)A^tC = T$.

5) Тогава, $XS = T$.

$X = TS^{-1}$, където

S^{-1} е обратната на S матрица.

II. Алгоритъм за прилагане на горното в програма на C++.

1) Въвеждаме дадените матрици от клавиатурата.

2) Транспонираме A. (вж. Алгоритъм VII).

3) Умножаваме $A^t . C$ (вж. Алгоритъм VIII).

4) Умножаваме $A^t . C$ с (-3).
$T = (-3) A^t C$
(вж. Алгоритъм IX).

5) Умножаваме $E * 5$. (вж. Алгоритъм IX).

6) Събираме $A + 5E = S$.
(вж. Алгоритъм X).

7) Намираме обратната на S матрица.
(вж. Алгоритъм III).

8) Ако S е необратима матрица, връщаме резултат матрица X с елементи единствено нули, за да сигнализираме грешка.

9) Умножаваме T с S^{-1} отдясно.
(вж. Алгоритъм VIII)

10) Получихме X.

III Алгоритъм за намиране на обратна матрица.

Вход: матрица $A = (a_{ij})_{n \times n}$

Изход: матрица $A^{-1}_{n \times n}$.

1) Пресмятаме det (A) и проверяваме дали е 0. (вж. Алгоритъм VI)

а) Ако е 0, спираме алгоритъма и извеждаме грешка. (Матрицата е необратима).

б) Ако не е 0, продължаваме.

2) Намираме матрица $X = (A_{ji})_{n \times n}$, където A_{ji} са адюнгираните количества на елементите a_{ij} от A. (вж. Алгоритъм IV).

3) $A^{-1} = \frac{1}{\det A} \cdot X$, т.е. необходимо е единствено да разделим всеки елемент на X на $\det A$.

4) Получихме A^{-1}.

IV Алгоритъм за намиране на $X = (A_{ji})_{n \times n}$.

Вход: матрица $A = (a_{ij})_{n \times n}$

Изход: матрица $X = (A_{ji})_{n \times n}$.

За всяко i от 0 до n:
  За всяко j от 0 до n:
    Намираме поддетерминантата Δ_{ij}.
    (вж. Алгоритъм V).

А) Ако $i+j$ е четно,
$A_{ji} = \Delta_{ij}$.

Б) Ако $i+j$ е нечетно,
$A_{ji} = (-1)\Delta_{ij}$.

Получихме матрицата $X = (A_{ji})_{n \times n}$.

V. Алгоритъм за намиране на подопределителя

Вход: матрица $A=(a_{ij})_{n\times n}$, ред p, стълб q

Изход: (число) Δ_{pq}.

1) Премахваме от матрицата A всички елементи от ред p и стълб q.

2) Намираме детерминантата на тази матрица от ред $n-1$. (вж. Алгоритъм VI).

3) Получихме Δ_{pq}.

VI. Рекурсивен алгоритъм за намиране на детерминанта на матрица.

Ще използваме развитието на детерминантата на матрицата по нулевия (първия) ред (правило на Лаплас).

Вход: матрица $A=(a_{ij})_{n\times n}$, ред n

Изход: (число) $\det A$.

1) (Дъно на рекурсията)
Ако $n=1$, връщаме A_{00} (единствен елемент).

2) В противен случай, нека $\det A = 0$.
Тогава,
за всяко i от 0 до n:
$\det A = \det A + a_{0i} \cdot A_{0i}$ или всъщност
$\det A = \det A + a_{0i} \cdot (-1)^{0+i} \cdot \Delta_{0i}$,

където Δ_{0i} се пресмята рекурсивно чрез същия алгоритъм до достигане на ред $n = 1$.

3) Получихме $\det A$.

VII. Алгоритъм за транспониране на матрица.

Вход: матрица $A = (a_{ij})_{n \times n}$

Изход: матрица $A^t_{n \times n} = B = (b_{ij})_{n \times n}$

За всяко i от 0 до n:
За всяко j от 0 до n:
$b_{ij} = a_{ji}$.

VIII. Алгоритъм за умножение на матрици

Вход: матрица $A = (a_{ik})_{n \times n}$, $B = (b_{kj})_{n \times n}$

Изход: матрица $C = A \cdot B = (c_{ij})_{n \times n}$

За всяко i от 0 до n:
  За всяко j от 0 до n:
    $c_{ij} = 0$
    За всяко k от 0 до n:
      $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

IX. Алгоритъм за умножение на матрица с число.

Вход: матрица $A = (a_{ij})_{n \times n}$, число x

Изход: матрица $B = xA = (b_{ij})_{n \times n}$.

За всяко i от 0 до n:
  За всяко j от 0 до n:
    $b_{ij} = a_{ij} \cdot x$.

X. Алгоритъм за събиране на матрици

Вход: матрици $A = (a_{ij})_{n \times n}$, $B = (b_{ij})_{n \times n}$

Изход: матрица $C = A + B = (c_{ij})_{n \times n}$.

За всяко i от 0 до n:
  За всяко j от 0 до n:
    $c_{ij} = a_{ij} + b_{ij}$.

Задача ③.

I Алгоритъм за намиране на базис на лин. пространство $U+W$, където $U = l(a_1, \dots, a_n)$, а W е пространството от решения на следната ХСЛУ:

$$(*)\begin{cases} b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 = 0 \\ \dots\dots\dots\dots\dots\dots\dots \\ b_{m1}x_1 + b_{m2}x_2 + b_{m3}x_3 + b_{m4}x_4 = 0 \end{cases}$$

като $U, W \leq \mathbb{R}^4$.

Вход: числа m, n
m на брой вектора $\{b_1, \dots, b_m\}$
n на брой вектора $\{a_1, \dots, a_n\}$

Изход: базис на $U+W$.

1) Въвеждаме необходимите данни от клавиатурата.

2) Извършваме елементарни преобразувания на матрицата с вектор-редове $\{b_1, \dots, b_m\}$. (вж. Алгоритъм II)

3) Намираме ФСР на (*). Лин. обвивка на получените вектори съвпада с W. (вж. Алгоритъм III).

За базис на $U+W$, трябва да комбинираме векторите съответно от линейните обвивки на U и W.

4) За осъществяване на горното, първо правим елементарни преобразувания на матрицата с вектор-редове $\{a_1, \ldots, a_m\}$.
(вж. Алгоритъм II).

5) Към ненулевите редове, получени след горното преобразуване, прибавяме векторите от ФСР на (*). (точка 3).

6) Правим за последен път елементарните преобразувания на матрицата, получена в предишната точка.

(вж. Алгоритъм II).

7) Получените ненулеви редове са базис на $L+W$. Поради имплементацията на Алгоритъм II обаче е възможно векторите да съдържат голям брой дробни числа.

8) Стремим се да открием подходящи числа, с които да умножим / разделим вектор-редовете, така че да получим матрица с предимно цели числа.

(вж. Алгоритъм IV).

9) Получените ненулеви вектори извеждаме на екрана.

II. Алгоритъм за елементарни преобразувания на матрица (метод на Гаус).

Вход: матрица $A_{n \times n}$

Изход: матрица $B_{n \times n} \sim A_{n \times n}$

Опитваме се да приведем матрицата в горно триъгълен вид.

1) Обхождаме елементите по главния диагонал:

1.1.) Търсим по-голям елемент по абсолютна стойност от текущия в същата колона, но само под него.

1.2.) Ако открием такъв, разменяме местата на текущия ред с реда, съдържащ максималния елемент.

1.3.) Нулираме елементите под текущия, като умножим текущия ред по $-\frac{y}{x}$ и го прибавим към необходимия, където y е елементът, който желаем да нулираме, а x е текущият елемент. Предварително сме проверили, че $x \neq 0$.

2) Получихме матрица $B \sim A$ в горно-триъгълен вид.

III. Алгоритъм за намиране на ФСР на ХСЛУ, чиято матрица предварително е била приведена в горнотриъгълен вид.

Вход: матрица $B_{n \times n}$.

Изход: матрица $S_{n \times n}$ – ФСР.

1) Чрез Алгоритъм V пресмятаме $r(B) = m$. $n = 4$ (в случая) е предварително зададената размерност на пространството. Тогава, ще имаме $n-m$ на брой вектора в S (ФСР), и общо $n-m$ на брой параметри, когато решаваме ХСЛУ.

2) Обхождаме матрицата B, започвайки от последния й ненулев ред и най-дясната колона.

а) Ако текущият елемент не е 0, не е записан като параметър или изчислима променлива, записваме го като такава.

б) Всички елементи вляво записваме като параметри, освен ако не са нули.

в) В противен случай, продължаваме обхождането.

3) Получихме масив с местата на необходимите ни параметри (напр. 0110 – трябват ни параметри за x_2 и x_3) и масив с местата на изчислимите променливи (напр. 1001 – изчислими са x_1 и x_4).

4) Задаваме стойности на параметрите във ФСР, като за всеки параметър записваме само на едно място стойност, различна от 0 (в случая 1).

5) Изчисляваме стойностите на изчислимите променливи съгласно зададените за параметрите стойности, съответно за всеки вектор от ФСР.

6) Получихме п-т на брой вектори, представляващи ФСР на ХСЛУ.

IV. Алгоритъм за намиране на подходящ множител / делител за „опростяване" на матрица

Вход: матрица $A_{n \times m}$

Изход: матрица $B_{n \times m} \sim A$.

Следното изпълняваме за всеки ред от A:

1) Пресмятаме броя на дробните елементи от реда.

2) Ако този брой е 0, преминаваме към следващия ред.

3) В противен случай,

За всяко i от 2 до 25:

3.1.) Умножаваме текущия ред по i.

3.2.) Отново пресмятаме броя на дробните елементи.

3.3.) Ако броят е по-малък от предходния, запазваме това умножение.

3.4.) Ако сме достигнали 0 на броя дробни елементи, прекратяваме този цикъл.

4.) За всяко i от 25 до 2:

4.1.) Делим текущия ред на i.

4.2) Пресмятаме броя на дробните елементи.

4.3.) Ако няма дробни елементи, сме открили НОД на всички елементи от реда (≤ 25), затова него запишем делителя и спрем цикъла.

5) Получихме матрица $B_{n \times n} \sim A$ с броя на дробните елементи в нея по-малък или равен на изхода.

V. Алгоритъм за намиране на ранг на преобразувана матрица

Вход: матрица $A_{n \times n}$, преобразувана чрез м. Гаус

Изход: (число) $r(A)$

1) Нека $r(A) = 0$.

2) За всеки ред от A:
ако редът не е нулев, $r(A) = r(A) + 1$.

3) Получихме $r(A)$ – броят на ненулевите редове в A.

За конкретна имплементация погледнете кода, който ще ви бъде изпратен по e-mail.

За въпроси: kirilgolov1@gmail.com.