ПРОГРАММИРОВАНИЕ, 2023, № 1, с. 38–47

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 517.36

ВЫЧИСЛЕНИЕ УНИМОДУЛЯРНЫХ МАТРИЦ СТЕПЕННЫХ ПРЕОБРАЗОВАНИЙ

© 2023 г. А. Д. Брюно[a,*] (ORCID: 0000-0002-7465-1258),
А. А. Азимов[b,] (ORCID: 0000-0002-7799-2525)**

[a]*Институт прикладной математики им. М.В. Келдыша РАН,*
125047 Москва, Миусская пл., д. 4, Россия

[b]*Самаркандский государственный университет,*
140104 Самарканд, Университетский бульвар, д. 15, Узбекистан

**E-mail: abruno@keldysh.ru*

***E-mail: Azimov_Alijon_Akhmadovich@mail.ru*

Поступила в редакцию 01.08.2022 г.
После доработки 13.08.2022 г.
Принята к публикации 07.09.2022 г.

Здесь указан алгоритм решения следующей задачи. Пусть в n-мерном вещественном пространстве задано $m < n$ целочисленных векторов. Их линейная оболочка образует линейное подпространство L в $\mathbb{R}^n$. Требуется вычислить такую унимодулярную матрицу, что линейное преобразование с ней переводит подпространство L в координатное. Также приведены программы, реализующие эти алгоритмы, и степенные преобразования, для которых они предназначены.

DOI: 10.31857/S013234742301003X, **EDN:** GRRDJF

1. ВВЕДЕНИЕ

Напомним, что матрица называется унимодулярной, если она квадратная, все ее элементы – целые числа и ее определитель равен ± 1. Ее обратная матрица также унимодулярна.

Будем векторы обозначать как векторы-строки: $X = (x_1,\ldots,x_n)$. $[x]$ – это целая часть вещественного числа x.

Задача 1. Пусть в n-мерном вещественном пространстве $\mathbb{R}^n$ задано m, $(m < n)$ целочисленных векторов $A_1,\ldots,A_m$. Их линейная оболочка

$$L = \left\{ X = \sum_{j=1}^{m} \lambda_j A_j, \lambda_j \in \mathbb{R}, j = 1,\ldots,m \right\} \qquad (1.1)$$

образует линейное подпространство в $\mathbb{R}^n$. Требуется вычислить такую унимодулярную матрицу α, что преобразование

$$X\alpha = Y$$

переводит подпространство L в координатное подпространство

$$M = \{Y : y_{n-l+1} = \ldots = y_n = 0\},$$

где $l = \dim L$.

Здесь указан алгоритм решения этой задачи, и составлена программа его реализующая. Это некоторое обобщение алгоритма цепной дроби [1], который напоминается в разделе 2. В разделе 3 описан алгоритм Эйлера [2], который обобщает алгоритм Евклида (т.е. цепной дроби) на n-мерный целочисленный вектор. В разделе 4 дается решение задачи 1, а в разделах 5, 6 – программы, соответствующие разделам 2, 3, 4. В разделе 7 рассмотрены степенные преобразования. Для вычисления их унимодулярных матриц развиты все эти алгоритмы и программы.

2. АЛГОРИТМ ЕВКЛИДА И ЦЕПНАЯ ДРОБЬ

Задача 2. Пусть заданы 2 целых положительных числа a_1 и a_2. Надо найти их наибольший общий делитель (НОД).

Для этого используется **алгоритм Евклида**. Пусть $a_1 \geq a_2$; делим a_1 на a_2 с остатком:

$$a_1 = b_1 \cdot a_2 + a_3, \qquad (2.1)$$

где $b_1 = [a_1/a_2]$ и a_3 – целые числа, $0 \leq a_3 < a_2$. Если $a_3 = 0$, то НОД равен a_2. Если $a_3 \neq 0$, то делим с остатком a_2 на a_3:

$$a_2 = b_2 \cdot a_3 + a_4, \qquad (2.2)$$

где $b_2 = [a_2/a_3]$ и где $0 \le a_4 < a_3$. Если $a_4 = 0$, то НОД равен a_3. Если $a_4 \ne 0$, то продолжаем процедуру, пока не получим нулевой остаток $a_{k+1} = 0$.

Тогда НОД – это a_k. Эту процедуру можно записать в виде цепной дроби

$$\frac{a_1}{a_2} = b_1 + \cfrac{1}{b_2 + \cfrac{1}{b_3 + \cfrac{1}{\ldots + \cfrac{1}{b_{k-1}}}}}. \qquad (2.3)$$

Она применима к любому вещественному числу β и дает, вообще говоря, бесконечное разложение. Только для рациональных чисел $\beta = a_1/a_2$ оно конечно. Для квадратичных иррациональностей β оно периодично [1]. Если в цепной дроби (2.3) отбросить окончание, начинающееся с b_{l+1}, и свернуть полученную цепную дробь в рациональное число, то оно называется *подходящей дробью*.

Задача 3. Пусть заданы 2 целых положительных числа a_1 и a_2. Надо вычислить такую унимодулярную матрицу α, что $(a_1, a_2)\alpha = (a_k, 0)$ или $(0, a_k)$, где целое число $a_k > 0$.

Деление с остатком (2.1) можно записать в виде умножения на матрицу $(a_1, a_2)\begin{pmatrix} 1 & 0 \\ -b_1 & 1 \end{pmatrix} = (a_3, a_2)$ или $(a_1, a_2)\beta_1 = (a_3, a_2)$, где $\beta_1 = \begin{pmatrix} 1 & 0 \\ -b_1 & 1 \end{pmatrix}$, а деление с остатком (2.2) есть $(a_3, a_2)\begin{pmatrix} 1 & -b_2 \\ 0 & 1 \end{pmatrix} = (a_3, a_4)$ или $(a_3, a_2)\beta_2 = (a_3, a_4)$, где $\beta_2 = \begin{pmatrix} 1 & -b_2 \\ 0 & 1 \end{pmatrix}$. Последний шаг алгоритма Евклида есть либо

$$(a_k, a_{k-1})\begin{pmatrix} 1 & -b_{k-1} \\ 0 & 1 \end{pmatrix} = (a_k, 0),$$

либо

$$(a_{k-1}, a_k)\begin{pmatrix} 1 & 0 \\ -b_{k-1} & 1 \end{pmatrix} = (0, a_k).$$

Поэтому искомая матрица

$$\alpha = \beta_1\beta_2\cdots\beta_{k-1},$$

где

$$\beta_j = \begin{pmatrix} 1 & 0 \\ -b_j & 1 \end{pmatrix}, \qquad (2.4)$$

если j нечетно,

$$\beta_j = \begin{pmatrix} 1 & -b_j \\ 0 & 1 \end{pmatrix}, \qquad (2.5)$$

если j четно.

Поскольку все матрицы β_j унимодулярны, то их произведение α также унимодулярная матрица. Она дает решение задачи 3.

Отметим, что

$$\alpha^{-1} = \beta_{k-1}^{-1}\beta_{k-2}^{-1}\cdots\beta_2^{-1}\beta_1^{-1}$$

и согласно (2.4) и (2.5) $\beta_j^{-1} = \begin{pmatrix} 1 & 0 \\ b_j & 1 \end{pmatrix}$ или $\begin{pmatrix} 1 & b_j \\ 0 & 1 \end{pmatrix}$, т.е. содержит неотрицательные элементы. Поэтому в матрице α^{-1} все элементы неотрицательны.

Пример 1. Пусть $a_1 = 17$, $a_2 = 5$. Тогда $b_1 = [17/5] = 3$, $a_3 = 2$, $\alpha_1 = \begin{pmatrix} 1 & 0 \\ -3 & 1 \end{pmatrix}$,

$$b_2 = \left[\frac{5}{2}\right] = 2, \quad a_4 = 1, \quad \alpha_2 = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix},$$

$$b_3 = \left[\frac{2}{1}\right] = 2, \quad a_5 = 0, \quad \alpha_3 = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}.$$

Матрица $\alpha = \alpha_1\alpha_2\alpha_3 = \begin{pmatrix} 1 & 0 \\ -3 & 1 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ -17 & 7 \end{pmatrix}$.

Раскладывая в цепную дробь

$$\frac{17}{5} = 3 + \frac{1}{2 + \frac{1}{2}}, \qquad (2.6)$$

получим, что последняя подходящая дробь – это $3 + 1/2 = 7/2$. Поэтому в матрице α второй столбец состоит из $-2, 7$.

Здесь было предложено решение задачи 3 для $a_1, a_2 > 0$. Если $a_1, a_2 < 0$, то надо взять матрицу α для вектора $(-a_1, -a_2)$. Если $a_1 \cdot a_2 < 0$, то надо взять матрицу $\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix}$ для вектора $(|a_1|, |a_2|)$. Тогда матрица

$$\alpha = \begin{pmatrix} \alpha_{11}\operatorname{sign} a_1 & \alpha_{12}\operatorname{sign} a_2 \\ \alpha_{21}\operatorname{sign} a_1 & \alpha_{22}\operatorname{sign} a_2 \end{pmatrix}$$

является унимодулярной и аннулирует одну из координат вектора (a_1, a_2).

Здесь предполагалось, что $|a_1| \ge |a_2|$. Если это не так, то надо предварительно сделать перестановку координат

$$(a_1, a_2)\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = (a_2, a_1).$$

Похожее изложение см. в п. 1.9, § 1, гл. I [3].

3. АЛГОРИТМ ЭЙЛЕРА И ОБОБЩЕНИЕ ЦЕПНОЙ ДРОБИ

Задача 4. Пусть задан n-мерный целочисленный вектор $A = (a_1, a_2, \ldots, a_n)$. Надо найти такую n-мерную унимодулярную матрицу α, что вектор $A\alpha = C = (c_1, \ldots, c_n)$ содержит только одну координату c_n, отличную от нуля.

Для ее решения Эйлер [2] предложил следующий алгоритм. Пусть для начала все координаты вектора A неотрицательны. С помощью перестановки $A\alpha_0 = (\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n)$ упорядочим координаты

$$\tilde{a}_j \le \tilde{a}_{j+1}, \quad j = 1, \ldots, n-1.$$

Здесь α_0 – унимодулярная матрица перестановки. Пусть $\tilde{a}_k$ – наименьшее из чисел $\tilde{a}_j$, отличное от нуля.

Пусть

$$b_j = \left[\frac{\tilde{a}_j}{\tilde{a}_k}\right], \quad j = 1, \ldots, n.$$

При этом $b_1 = \cdots = b_{k-1} = 0$, $b_k = 1$. Делаем преобразование

$$d_j = \tilde{a}_j - b_j\tilde{a}_k, \quad 1 \le j \le n, \quad j \ne k, \quad d_k = \tilde{a}_k. \quad (3.1)$$

Ему соответствует унимодулярная матрица α_1, у которой на диагонали стоят единицы, в k-й строке стоят

$$0, 0, \ldots, 0, 1, -b_{k+1}, \ldots, -b_n,$$

т.е.

$$\tilde{A}\alpha_1 = D = (d_1, \ldots, d_n).$$

Теперь упорядочиваем компоненты вектора D с помощью унимодулярной матрицы-перестановки β_0 так, что $D\beta_0 = \tilde{D} = (0, \ldots, 0, \tilde{d}_k, \ldots, \tilde{d}_n)$, где $\tilde{d}_j \le \tilde{d}_{j+1}$.

Пусть $\tilde{d}_l$ – наименьшее из $\tilde{d}_j$, отличное от нуля, и $e_j = [\tilde{d}_j/\tilde{d}_l]$, $j = 1, \ldots, l$. Делаем преобразование

$$f_j = \tilde{d}_j - e_j\tilde{d}_l, \quad 1 \le j \le n, \quad j \ne l, \quad f_l = \tilde{d}_l,$$

и так далее. На каждом шаге максимум координат вектора убывает и является n-й координатой. Поэтому через конечное число шагов получаем вектор с одной ненулевой координатой – последней. Ее величина – это НОД всех исходных координат $a_1, \ldots, a_n$. Каждый шаг состоит из матрицы-перестановки и треугольной матрицы с единичной диагональю:

$$A\alpha_0\alpha_1\beta_0\beta_1\gamma_0\gamma_1 \ldots \omega_0\omega_1 = A\alpha = C = (0, \ldots, 0, c_n).$$

Матрица

$$\alpha = \alpha_0\alpha_1\beta_0\beta_1\gamma_0\gamma_1 \cdots \omega_0\omega_1 \quad (3.2)$$

является решением задачи 4.

Если не все координаты a_j исходного вектора A одного знака, то сначала упорядочиваем их по модулю

$$|\tilde{a}_j| \le |\tilde{a}_{j+1}|$$

и полагаем

$$b_j = [|\tilde{a}_j| / |\tilde{a}_k|] \operatorname{sign} \tilde{a}_j \operatorname{sign} \widetilde{a_k}.$$

Замечание 1. Умножая справа матрицу α на унимодулярную матрицу-перестановку, можно из вектора C получить вектор, у которого все координаты кроме одной равны нулю, а единственная ненулевая координата расположена на любом месте.

Пример 2. Пусть $n = 4$ и $A = (5, 2, 4, 3)$. Упорядочиваем координаты вектора A: $A \cdot \begin{pmatrix} 0&0&0&1 \\ 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \end{pmatrix} = \tilde{A} = (2, 3, 4, 5)$, где $\begin{pmatrix} 0&0&0&1 \\ 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \end{pmatrix} = \alpha_0$, имеем $k = 1$. Получаем

$$b_2 = \left[\frac{3}{2}\right] = 1, \quad b_3 = \left[\frac{4}{2}\right] = 2, \quad b_4 = \left[\frac{5}{2}\right] = 2.$$

Поэтому

$$\alpha_1 = \begin{pmatrix} 1&-1&-2&-2 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix},$$

$$\tilde{A}\alpha_1 = (2, 1, 0, 1) = D.$$

Упорядочиваем координаты вектора D:

$$D\begin{pmatrix} 0&0&0&1 \\ 0&1&0&0 \\ 1&0&0&0 \\ 0&0&1&0 \end{pmatrix} = (0, 1, 1, 2) = \tilde{D}, \text{ где } \begin{pmatrix} 0&0&0&1 \\ 0&1&0&0 \\ 1&0&0&0 \\ 0&0&1&0 \end{pmatrix} = \beta_0.$$

Здесь $l = 2$. Получаем $e_1 = 0$, $e_2 = 1$, $e_3 = 1$, $e_4 = 2$ и

$$\beta_1 = \begin{pmatrix} 1&0&0&0 \\ 0&1&-1&-2 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}.$$

$\tilde{D}\beta_1 = (0,1,0,0)$. Наконец, $\gamma_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ и

$\tilde{D}\beta_1\gamma_0 = (0,0,0,1) = C$.

Здесь

$$\alpha = \alpha_0\alpha_1\beta_0\beta_1\gamma_0 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & -1 & 3 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 1 \end{pmatrix}. \qquad (3.3)$$

Пусть заданный вектор A является нормалью к некоторому линейному многообразию. Тогда после преобразования с помощью матрицы α получим, что этот вектор содержит первые $n-1$ нулевые координаты. Следовательно, все векторы исходного многообразия после преобразования будут иметь последней координатой ноль.

Алгоритм Эйлера обобщает алгоритм цепной дроби только для целочисленных векторов. Для произвольных вещественных векторов это обобщение искали все крупные математики XIX века. Но безуспешно. В [4] предложено такое обобщение цепной дроби для n-мерного вектора, которое дает последовательность наилучших приближений и периодично, если все координаты исходного вектора являются корнями многочлена степени n с целыми коэффициентами.

4. РЕШЕНИЕ ЗАДАЧИ 1

Пусть заданы целочисленные векторы

$$\begin{aligned} A_1 &= (a_{11}, a_{12}, \ldots, a_{1n}), \\ A_2 &= (a_{21}, a_{22}, \ldots, a_{2n}), \\ &\ldots \\ A_m &= (a_{m1}, a_{m2}, \ldots, a_{mn}) \end{aligned} \qquad (4.1)$$

($m < n$) и линейное пространство (1.1).

Первым делом проверяем, что среди них нет одинаковых. Если есть, то оставляем из них только один, а остальные отбрасываем. Поэтому будем считать, что все векторы (4.1) разные. Теперь применяем алгоритм Эйлера к вектору A_1, т.е. вычисляем матрицу α_0, такую, что $A_1\alpha_0 = C_1 = c_nE_n$, где c_n – целое число и E_k – это k-й единичный вектор.

Пусть $A_j\alpha_0 = C_j = (c_{j1}, \ldots, c_{jn})$, $j = 2, \ldots, m$. Полагаем $A_j^1 = (c_{j1}, \ldots, c_{jn-1})$, $j = 2, \ldots, m$. Применяем алгоритм Эйлера к $(n-1)$-мерному вектору A_2^1. Получаем $A_2^1\alpha_1 = C_2^1 = (0,0,\ldots,c_{n-1}^1)$, где α_1 – это $(n-1)$-мерная квадратная матрица. Пусть $A_j^1\alpha_1 = C_j^1 = (c_{j1}^1, \ldots, c_{jn-1}^1)$, $j = 3, \ldots, m$. Применяем алгоритм Эйлера к $(n-2)$-мерному вектору C_3^1 и т. д. В итоге получаем последовательность матриц $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ убывающих размерностей n, $n-1$, ..., $n-m+1$. Образуем блочные квадратные матрицы $\beta_j = \begin{pmatrix} \alpha_j & 0 \\ 0 & I_{j+1} \end{pmatrix}$, $j = 0, \ldots, n-m$, размерности n, где I_{j+1} – это единичные матрицы размерности $j+1$. Положим

$$\gamma = \beta_0\beta_1\cdots\beta_{m-1}.$$

Тогда $A_j\gamma = (0,0,\ldots,0,w_{j,n-j+1},\ldots,w_{j,n}) = W_j$, $j = 1, \ldots, m$. Матрица γ дает решение задачи 1.

Замечание 2. Умножая справа матрицу γ на матрицу-перестановку $\begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix}$, можно из набора векторов $W_1, \ldots, W_m$ получить треугольную матрицу $U = (u_{jk})$, $j = 1, \ldots, m$, $k = 1, \ldots, n$, у которой $u_{jk} = 0$, если $k > j$.

Пример 3 (продолжение примера 2). Пусть $n = 4$, $m = 2$, заданы векторы $A_1 = (5,2,4,3)$, $A_2 = (7,8,9,3)$. Согласно примеру 2 матрица α из (3.3) приводит вектор A_1 к виду (0, 0, 0, 1). Имеем

$$A_2 \cdot \alpha = (7,8,9,3)\begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & -1 & 3 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 1 \end{pmatrix} = (-7,-4,18,-5) = C_2,$$

образуем вектор $A_2^1 = (-7,-4,18)$. К этому трехмерному вектору применяем алгоритм Эйлера. Упорядочиваем его координаты по модулю

$$A_2^1\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (-4,-7,18).$$

Имеем

$$d_2 = \left[\frac{7}{4}\right] = 1, \quad d_3 = -\left[\frac{18}{4}\right] = -4.$$

Получаем

$$(-4,-7,18)\begin{pmatrix} 1 & -1 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (-4,-3,2).$$

Упорядочиваем координаты по модулю

$$(-4,-3,2)\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}=(2,-3,-4).$$

Имеем

$$d_2=-\left[\frac{3}{2}\right]=-1,\quad d_3=-\left[\frac{4}{2}\right]=-2.$$

Получаем

$$(2,-3,-4)\begin{pmatrix}1&1&2\\0&1&0\\0&0&1\end{pmatrix}=(2,-1,0).$$

Упорядочиваем

$$(2,-1,0)\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}=(0,-1,2).$$

Имеем

$$d_3=-\left[\frac{2}{1}\right]=-2,$$

получаем

$$(0,-1,2)\begin{pmatrix}1&0&0\\0&1&2\\0&0&1\end{pmatrix}=(0,-1,0).$$

Упорядочиваем

$$(0,-1,0)\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}=(0,0,-1).$$

Теперь

$$\alpha_1=\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}1&-1&4\\0&1&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\begin{pmatrix}1&1&2\\0&1&0\\0&0&1\end{pmatrix}\times$$
$$\times\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\cdot\begin{pmatrix}1&0&0\\0&1&2\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}=\begin{pmatrix}0&2&1\\9&10&3\\2&3&1\end{pmatrix}. \qquad (4.2)$$

Полагаем

$$\beta_1=\begin{pmatrix}\alpha_1&0\\0&1\end{pmatrix}=\begin{pmatrix}0&2&1&0\\9&10&3&0\\2&3&1&0\\0&0&0&1\end{pmatrix}.$$

Теперь

$$\gamma=\alpha\beta_1=\begin{pmatrix}0&1&0&0\\-2&-1&3&-1\\1&0&0&0\\0&-1&-2&1\end{pmatrix}\cdot\begin{pmatrix}0&2&1&0\\9&10&3&0\\2&3&1&0\\0&0&0&1\end{pmatrix}=$$
$$=\begin{pmatrix}9&10&3&0\\-3&-5&-2&-1\\0&2&1&0\\-13&-16&-5&1\end{pmatrix}.$$

Проверяем $A_1\gamma=(0,0,0,1)$, $A_2\gamma=(0,0,-1,-5)$. Итак,

$$\begin{pmatrix}A_1\\A_2\end{pmatrix}\gamma=\begin{pmatrix}0,0,0,1\\0,0,-1,-5\end{pmatrix}.$$

5. ПРОГРАММЫ ВЫЧИСЛЕНИЯ ЦЕПНОЙ ДРОБИ

Алгоритмы вычисления цепных дробей реализованы в различных системах. Здесь опишем основные процедуры в двух системах компьютерной алгебры (СКА): в проприетарной `Maple` и в открытой `sympy`. Пакет `NumberTheory` в СКА Maple [5] позволяет получать разложение в цепную дробь рациональных, алгебраических и трансцендентных чисел, а также полиномов или элементарных функций от одной переменной. В пакете sympy [7] такой функционал реализован только для рациональных чисел или квадратичных иррациональностей. Если требуется работа с цепными дробями для других иррациональностей или трансцендентных чисел, то следует использовать открытую СКА `Sage` [6].

Для работы с рациональным числом в виде цепной дроби достаточно трех основных процедур: [1)]

1) преобразование в цепную дробь;

2) получение элементов цепной дроби;

3) получение рациональных приближений.

В СКА `Maple` соответствующие действия обеспечивают процедуры `ContinuedFraction`, `Term` и `Convergent`, а в sympy функционал пунктов 1 и 2 обеспечивает процедура `continued_fraction`, а пункта 3 – процедура *continued_fraction_convergents*.

Ниже приведем примеры реализации вычисления унимодулярных 2 × 2 матриц в соответствии с алгоритмами п. 1.9 книги [3]. Эти алгоритмы используют цепные дроби.

Первый алгоритм (см. [3], стр. 28–30) использует слагаемые разложения рационального числа p/q, а итоговая унимодулярная матрица получается путем последовательного умножения верхне- или нижнетреугольных матриц, которые составляются на основе элементов разложения цепной

дроби. Его реализация для Maple и для sympy приведена на листингах 1 и 2 соответственно.

Листинг 1

```
UniMod1:=proc (p:: integer, q:: integer)
description "Compute unimodular 2 × 2-
    matrix with the help of continued
    fraction";
uses NumberTheory, LinearAlgebra,
    ListTools;
local pabs:=abs (p), qabs:=abs (q), gcdpq
    :=igcd (pabs, qabs), cf_terms,
k,M, alpha:=DiagonalMatrix ([1, 1]);
if gcdpq!=1 then pabs:=pabs/gcdpq;
    qabs:=qabs/gcdpq; end if;
cf_terms:= Term(ContinuedFraction (
    pabs/qabs), all);
for k in [seq] ([i, cf_terms [i]], i =1..
    numelems (cf_terms)) do
M:=DiagonalMatrix ([1, 1]);
if type (k [1], even) then M [2, 1]:= -k
    [-1] else M [1, 2]:= -k[-1] end if;
alpha:=M. alpha;
end do;
return Matrix ([sign (p) *Column(alpha
    , 1), sign (q) *Column(alpha, 2)]);
[-1] else M [1, 2]:= =k[=1] end if;
alpha:=M. alpha;
end proc;
```

Листинг 2

```
import sympy as sym
from sympy import Rational, eye, Matrix
from sympy.ntheory.continued_fraction\
import continued_fraction_convergents,\
continued_fraction_iterator,\
continued_fraction

def UniMod1(p, q):
"""
Construct 2 × 2 unimodular matrix
for fraction p/q.
First variant
"""
r - Rational (p, q)
cfr - continued_fraction (r)
Mlst - [eye (2.1) for k in range (len (cfr))]
for k, m in enumerate(cfr):
if k%2=-1: Mlst [k] [1,0]=-m
else: Mlst [k] [0,1]= -m
alpha = eye (2.1)
for M in Mlst [:: -1]: alpha*-M
return alpha
```

Пример работы процедуры UniMod1 для пары чисел 5, 17 примера 1 приведен на листинге 3.

Листинг 3

```
>r := [5, 17]: UniMod2(op(r1)); %.Vector (r1);
```

$$\begin{bmatrix} 7 & -2 \\ -17 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Второй алгоритм (см. [3], стр. 30–31) использует только само рациональное число p/q и его последнюю подходящую дробь p_{n-1}/q_{n-1}. Его реализация приведена на листингах 4 и 5 соответственно.

Листинг 4

```
UniMod2:=proc (p:: integer, q:: integer)
description "Compute unimodular 2 × 2=
    matrix with the help of continued
    fraction. Second variant"; uses
    NumberTheory;
local pabs:=abs (p), qabs:=abs (q), gcdpq
    :=igcd (pabs, qabs), cf, cf_conv,
gamma, rho, Sigma, alpha;
if gcdpq!=1 then pabs:=pabs/gcdpq;
    qabs:=qabs /gcdpq; end if;
cf:= ContinuedFraction (pabs/qabs);
cf_conv:=Convergent (cf, all);
gamma:=cf_conv [=1];
rho:=cf_conv [=2];
Sigma:=numer (gamma=rho);
alpha:=Matrix ([[s i gn (p) *Sigma*denom(
    rho),=sign (q) *Sigma*numer (rho)],
[-sign (p) *denom(gamma), sign (q) *numer (
    gamma)]]);
return alpha;
end proc;
```

Листинг 5

```
import sympy as sym
```

```
from sympy import Rational, eye, Matrix
from sympy.ntheory.continued_fraction\
import continued_fraction_convergents,\
continued_fraction_iterator,\
continued_fraction

def UniMod2(p, q):
"""
Construct 2 × 2 unimodular matrix
for fraction p/q.
Second variant
"""
r = Rational (p, q)
cfr = continued_fraction (r)
cfr_conv = \
list (continued_fraction_convergents (cfr))
gamma = cfr1_conv [=1]
rho = cfr1_conv [=2]
sigma=(gamma=rho). numerator
alpha = Matrix ([[sigma* rho. denomina-
tor, \
=sigma * rho.numerator], \
[=gamma.denominator, gamma. numerator]])
return alpha
```

6. РЕАЛИЗАЦИЯ АЛГОРИТМА ЭЙЛЕРА И РЕШЕНИЯ ЗАДАЧИ 1

Для имплементации алгоритма Эйлера, описанного в разделе 3, был реализован набор процедур в СКА Maple, листинги которых представлены ниже вместе с кратким их описанием. Отметим, что целочисленный вектор в СКА Maple может быть представлен в двух различных формах: в виде списка (перечня чисел в квадратных скобках) или в виде вектора-строки (вектора-столбца) пакета LinearAlgebra (Vector[row] или Vector[column] соответственно). Если имя процедуры содержит цифру 2, то это означает, что входной целочисленный вектор может быть задан в одном из двух указанных видов.

Процедура MakePermute2 представлена на листинге 6. Она строит перестановочную матрицу по заданному вектору A. Результат работы процедуры – упорядоченный вектор и перестановочная матрица α_0. Порядок сортировки элементов вектора задается параметром sorting, но по умолчанию элементы упорядочиваются по возрастанию. В начале работы процедура проверяет, что вектор A не является нульмерным (строка 4 листинга 6).

Листинг 6

```
1  MakePermute2:= proc (A:: {Vector, list
      }, sorting:='<')
   uses LinearAlgebra;
   local Asort, Aind, Aper, i, nA:=
      numelems (A);
   if nA=0 then error ("Zero dimensional
      vector!"); end if;
   Aind:= sort (abs~(A), sorting, output
      = [permutat ion]);
6  Asort:= A[Aind];
   Aper:= Matrix (nA, nA, fill = 0);
   for i to numelems (Aind) do
   Aper [i, Aind [i]]:= 1;
   end do;
11 if type (A, Vector [row]) then
   return Asort, Transpose (Aper);
   else
   return Asort, Aper;
   end if;
16 end proc;
```

Вторая процедура MakeUnimod2 (листинг 7) для упорядоченного по возрастанию вектора A строит унимодулярную матрицу α_1, реализующую один шаг (3.1) алгоритма Эйлера.

Листинг 7

```
1  MakeUnimod2:= proc (As:: {Vector, list
     })
   uses ListTools, LinearAlgebra;
   local M, i, Amin, nminpos, ncol, absAs
     :=abs~(As),nA;
4  nA:=numelems (As);
   if nA=0 then error ("Zero dimensional
     vector!"); end if;
   M:= DiagonalMatrix ([seq] (1, k = 1..
     nA));
   Amin, nminpos:= FindMinimalElement (
     convert (absAs, list), position);
   Amin:= As [nminpos];
9  if Amin = 0 then
   ncol:= [Sear chAl l] (0, convert (absAs,
      list)) [=1] + 1;
   else
   ncol:= nminpos;
   end if;
14 for i from ncol+1 to nA do
   M[i, ncol]:= =t runc (As [i] /As [ncol]);
   end do;
   if type (As, Vector [row]) then
```

```
   return LinearAlgebra:=Transpose (M);
19 else
   return M;
   end if;
   end proc;
```

Рекурсивная процедура Unimodr2 (листинг 8) вычисляет по исходному вектору A унимодулярную матрицу α, которая преобразует A в вектор C с единственной последней ненулевой координатой. При первом ее вызове второй параметр Uni не указывается. В этом случае он полагается равным единичной матрице (строка 7 листинга). При последующих вызовах в процедуру передается унимодулярная матрица, вычисленная на предыдущих вызовах. Для своей работы процедура Unimod2 использует описанные выше процедуры MakePermute2 и MakeUnimod2. Условие окончания рекурсии – это получение вектора, у которого все элементы, кроме одного, равны нулю (строка 16 листинга).

Листинг 8

```
1  Unimodr2:= proc (A:: {Vector, list},
      Uni:: Matrix)
   uses LinearAlgebra, ListTools;
3  local Alen, Avec, Alst, As, Aper, M,
      Mperm, Auni, res, _;
   Alen:= numelems (A)
   if Alen=0 then error ("Zero
      dimensional vector!"); end if;
   if nargs = 1 then
   res:= DiagonalMatrix ([seq] (1, k = 1
      .. Alen));
8  else
   res:= Uni;
   end if;
   if type (A, list) then
   Alst:=A; Avec:=convert (A, Vector [
      column]);
13 elif type (A, Vector) then
   Alst:=convert (A, list); Avec:=convert (A
      , Vector [column]);
   end if;
   if Occurrences (0, Alst) = Alen = 1
      then
   _, Mperm:= MakePermute2 (A);
18 if type (A, list) then
   return convert (Mperm. Vector (A), list),
      Mperm. res;
   elif type (A, Vector [row]) then
   return A.Mperm, res.Mperm;
   else
23 return Mperm. Avec, Mperm. res;
   end if;
   end if;
   As, Aper:= MakePermute2 (A);
   M:= MakeUnimod2 (As);
28 if type (A, list) then
   Auni:= M. Aper;
   return Unimodr2 (convert (Auni. (Vector (
      A)), list), Auni. res);
   elif type (A, Vector [row]) then
   Auni:= Aper.M;
33 return Unimodr2 (A. Auni, res. Auni);
   el se
   Auni:= M. Aper;
   return Unimodr2 (Auni.A, Auni. res);
   end if;
38 end proc;
```

Ниже приведен листинг 9 решения примера 2 для вектора $A = (5,2,4,3)$.

Листинг 9

```
>A:= [5, 2, 4, 3]: Arow:=Vector [row] (A):
>Unimodr (Arow);
```

$$[0\ 0\ 0\ 1],\begin{bmatrix} 0 & 1 & 0 & 0 \\ -2 & -1 & 3 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 1 \end{bmatrix}$$

Решение задачи 1 в соответствии с алгоритмом раздела 4 реализовано в рекурсивной процедуре UniSys, приведенной на листинге 10. Процедура получает исходный набор целочисленных векторов A_j, $j = 1,\ldots,m$ в виде списка Vlst. Если исходный список пуст (строка 5), либо число векторов превышает их размерность (строка 8), либо векторы являются линейно зависимыми (строки 9–15), либо векторы из набора имеют разный тип (строки 16–24) или разную размерность (строки 25–28), то процедура UniSys завершает свою работу. Если список состоит из одного вектора, то вызывается процедура Unimodr2, вычисляется матрица α – и на этом работа процедуры окончена. В противном случае осуществляется повторный вызов процедуры UniSys для набора векторов A_j^1, $j = 2,\ldots,m$, из которого исключен первый вектор, к оставшимся векторам применена унимодулярная матрица α. При этом размерность векторов A_j^1 уменьшена на единицу, а матрица α передается в виде параметра Uni для повторного вызова процедуры. При успешном окончании ра-

боты процедура UniSys возвращает итоговую матрицу γ, решающую задачу 1.

Листинг 10

```
UniSys:=proc (Vlst:: list, Uni:: Matrix)
uses LinearAlgebra;
local res, dlst,UM,_,Vdim, nVlst, Vtcol
   :=t rue,Mtmp;
nVl st:=numelems (Vl s t);
if nVl st=0 then error ("Initial list
   is empty!"); end if;
Vdim:=Dimension (Vlst [1]);
if type (Vlst [1], Vector [row]) then
   Vtcol:= false; end if;
if nVlst>Vdim then error ("Too many
   vector sofdimens ion ",Vdim); end
   if;
if Vtcol then
Mtmp:=Matrix (Vlst);
if Rank(Mtmp)<nVl st then error ("
   Vectors are not independent ! ");
   end if;
else
Mtmp:=<op (Vlst)>;
if Rank(Mtmp)<nVl st then error ("
   Vectors are not independent!");
   end if;
end if;
if Vtcol then
if not evalb ('and' (op ([seq] (type (V,
   Vector [column]),V in Vl s t)))) then
error ("All elements should be Vectors
   ");
end if;
else
if not evalb ('and' (op ([seq] (type (V,
   Vector [row]),V in Vl s t)))) then
error ("All elements should be Vectors
   ");
end if;
end if;
dlst:=[seq] (Dimension (V),V in Vl st);
if ListTools [Occur rences] (dlst [1],
   dlst) <> nVlst then
error ("All vectors should be o f the
   same size");
end if;
if nargs = 1 then
res:= DiagonalMatrix ([seq] (1, k = 1..
   Vdim));
else
res:= Uni;
end if;
_,UM:=Unimodr2 (Vlst [1]);
if nVl st=1 then
if Vtcol then return UM. res; else
   return res.UM; end if;
end if;
if Vtcol then
return DiagonalMatrix ([UniSys2 ([seq] (
   SubVector (UM.V, [1.. Vdim=1]), V in
   Vl s t [2.. =1])), 1]).UM;
else
return UM. DiagonalMatrix ([UniSys2 ([
   seq] (SubVector (V.UM, [1.. Vdim=1]),
   V in Vlst [2.. =1])), 1]);
end if;
end proc;
```

Ниже приведен листинг 11 решения примера 3 для векторов $A = (5,2,4,3)$ и $B = (7,8,9,3)$.

Листинг 11

```
>A:= [5, 2, 4, 3]: Arow:=Vector [row] (A):
>B:= [7–9, 3]: Brow:=Vector [row] (B):
>UniSys ([Arow,Brow]);
```

$$[0\ 0\ 0\ 1], \begin{bmatrix} 9 & 10 & 3 & 0 \\ -3 & -5 & -2 & -1 \\ 0 & 2 & 1 & 0 \\ -13 & -16 & -5 & 1 \end{bmatrix}$$

7. СТЕПЕННЫЕ ПРЕОБРАЗОВАНИЯ

Пусть задан многочлен

$$f(X) = \sum f_Q X^Q, \quad Q \in \mathbf{S}, \tag{7.1}$$

где $X = (x_1,\ldots,x_n) \in \mathbb{R}^n$ или $\mathbb{C}^n$, $Q = (q_1,\ldots,q_n) \in \mathbb{Z}^n$, $Q \geq 0$, f_Q – постоянные коэффициенты из $\mathbb{R}$ или $\mathbb{C}$, $\mathbf{S} = \mathbf{S}(f)$ – носитель многочлена f. Пусть $\mathcal{F}$ – алгебраическое многообразие $f(X) = 0$ и точка $X = X^0 \in \mathcal{F}$.

Если X^0 – простая точка, т.е. хотя бы одна из производных $\partial f/\partial x_j$ в этой точке X^0 отлична от нуля, то, по теореме о неявной функции, вблизи точки X^0 многообразие $\mathcal{F}$ описывается уравнением

$$\Delta x_j = \varphi(\Delta x_1,\ldots,\Delta x_{j-1},\Delta x_{j+1},\ldots,\Delta x_n), \tag{7.2}$$

где $\Delta x_k = x_k - x_k^0$, φ – сходящийся степенной ряд от своих аргументов.

Если точка X^0 – не простая, то согласно [8, 9] можно искать ветви многообразия $\mathcal{F}$, проходящие через точку X^0, в виде параметрических разложений

$$\Delta x_j = \varphi_j(\xi_1,\ldots,\xi_{n-1}), \quad i = 1,\ldots,n, \tag{7.3}$$

где ξ_k – малые параметры, а φ_j – сходящиеся степенные ряды. Для этого строится выпуклая оболочка Γ носителя $\mathbf{S}$ в пространстве $\mathbb{R}^n$. Тогда Γ – это многогранник, граница которого $\partial\Gamma$ состоит из (обобщенных) граней $\Gamma_j^{(d)}$ размерностей d, $0 \le d < n$. Здесь j – это номер грани. Поскольку все вершины $\Gamma_j^{(0)}$ многогранника Γ целочисленны, то каждая грань $\Gamma_j^{(d)}$ имеет $n - d$ целочисленных линейно независимых нормалей $N_{j1}^{(d)},\ldots,N_{jn-d}^{(d)} \in \mathbb{R}_*^n$, т.е. лежащих в пространстве $\mathbb{R}_*^n$, двойственном (сопряженном) пространству $\mathbb{R}^n$.

Кроме того, каждой грани $\Gamma_j^{(d)}$ соответствуют граничное множество

$$D_j^{(d)} = \left\{Q \in S \cap \Gamma_j^{(d)}\right\}$$

и укороченная сумма

$$\hat{f}_j^{(d)}(X) = \sum f_Q X^Q \quad \text{по} \quad Q \in D_j^{(d)}. \tag{7.4}$$

Теорема ([8, следствие § 3, гл. II] и теорема 3.1 [9]). *Для грани $\Gamma_j^{(d)}$ существует степенное преобразование*

$$\ln Y = \ln X \cdot \alpha, \tag{7.5}$$

где $\ln Y = (\ln y_1,\ldots,\ln y_n)$, $\ln X = (\ln x_1,\ldots,\ln x_n)$ с унимодулярной матрицей α, которое переводит укороченную сумму (7.4) *в многочлен g от d координат, т.е.*

$$\hat{f}_j^{(d)}(X) = Y^T g(y_1,\ldots,y_d), \tag{7.6}$$

где $T = (t_1,\ldots,t_n) \in \mathbb{Z}^n$.

Но в [8, 9] не было указано, как вычислять унимодулярную матрицу α. Это сделано в настоящей работе. А именно: если $n = 2$, то с помощью цепной дроби раздела 2, если $d = n - 1$, то с помощью алгоритма Эйлера, если $n > 2$ и $d < n - 1$, то с помощью алгоритма раздела 4, решающего задачу 1.

СПИСОК ЛИТЕРАТУРЫ

1. *Хинчин А.Я.* Цепные дроби. М.: Физматгиз, 1961.
2. *Euler L.* De relatione inter ternas pluresve quantitates instituenda // 1785, All Works 591.
3. *Брюно А.Д.* Локальный метод нелинейного анализа дифференциальных уравнений. М.: Наука, 1979. 252 с.
4. *Брюно А.Д.* Вычисление основных единиц числовых колец с помощью обобщенной цепной дроби // Программирование. 2019. № 2. С. 17–31. https://doi.org/10.1134/S0132347419020055
5. *Thompson I.* Understanding Maple. Cambridge University Press, 2016. 228 p.
6. *The Sage Developers.* SageMath, the Sage Mathematics Software System (Version 9.1.1). 2020. https://doi.org/10.5281/zenodo. 4066866. https://www.sagemath.org.
7. *Meurer A., Smith C.P.,* [et al.]. SymPy: symbolic computing in Python // PeerJ Computer Science. 2017. V. 3. e103. ISSN 2376–5992. DOI: . URL: https://doi.org/10.7717/ peerj-cs.103.
8. *Брюно А.Д.* Степенная геометрия в алгебраических и дифференциальных уравнениях. М.: Физматлит, 1998. 288 с.
9. *Брюно А.Д., Батхин А.Б.* Разрешение алгебраической сингулярности алгоритмами степенной геометрии // Программирование. 2012. № 2. С. 11–28.