РЕШЕНИЕ АЛГЕБРАИЧЕСКИХ ЗАДАЧ В ПРОГРАММЕ MATHEMATICA

Нилуфар Вайитовна Жураева
Кафедра информатики и информационных технологий
Чирчикский государственный педагогический университет
E-mail: mina-uzb@mail.ru. Тел: +998903747418
Нигора Нурмухаммедовна Файзиева
Магистр кафедры информатики и информационных технологий
Чирчикский государственный педагогический университет
E-mail: nigorafayziyeva1999@gmail.com. Тел: +998998504611

Аннотация: *В статье представлены краткое описание и ход выполнения различных алгебраических расчетов в программе Mathematica и рассмотрены порядок правильной записи ввода данных и математических формул.*

Ключевые слова: *матрица, умножение матриц, решение алгебраических систем уравнений, обращение матриц, определитель матрицы.*

Abstract: *The article provides a brief description and progress of various algebraic calculations in the Mathematica program and discusses the procedure for correctly recording data input and mathematical formulas.*

Key words: *matrix, matrix multiplication, solution of algebraic systems of equations, matrix inversion, matrix determinant.*

Запись матриц. Матрица в языке Mathematica записывается как список, составленный из строк этой матрицы. Список, элементы которого сами являются списками, называется вложенным списком (nested list). Таким образом, матрица

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

изображается как {{a,b},{c,d}}. Именно такая форма используется при вводе матриц, внутри себя система переводит это выражение в полную форму List[List[a,b],List[c,d]].

Симметрия между строками и столбцами восстанавливается при помощи команды Transpose, переводящей матрицу в транспонированную матрицу, строки которой совпадают со столбцами исходной:

```
In[179]:= Transpose[{{a,b},{c,d}}]
Out[179]={{a,c},{b,d}}
```

По умолчанию команда Transpose переставляет два верхних уровня вложенности списков. Таким образом, например, команда Transpose, примененная к списку матриц будет переставлять строки этих матриц между собой, а вовсе не транспонировать сами эти матрицы. Для одновременного транспонирования списка матриц нужно применять Transpose к элементам этого списка при помощи команды Map:

```
In[180]:= Transpose[{{{a,b},{c,d}},{{e,f},{g,h}}}]
Out[180]={{{a,b},{e,f}},{{c,d},{g,h}}}
In[181]:= Map[Transpose,{{{a,b},{c,d}},{{e,f},{g,h}}}]
Out[181]={{{a,c},{b,d}},{{e,g},{f,h}}}
```

Из описанного представления матриц и того, что было сказано выше об операциях над векторами следует, что при действиях над матрицами все обычные операции тракутются как покомпонентные:

```
In[182]:= {{a,b},{c,d}}+{{e,f},{g,h}}
Out[182]={{a+e,b+f},{c+g,d+h}}
In[183]:= {{a,b},{c,d}}*{{e,f},{g,h}}
Out[183]={{a*e,b*f},{c*g,d*h}}
```

Таким образом, * (Times) это умножение матриц по Адамару, а вовсе не обычное произведение матриц, которое обозначается . (Dot). Более спорным представляется применение того же правила к скалярам, которые при этом отождествляются не с кратными единичной, а с кратными пробной матрицы, состоящей из одних единиц! Изобразим для примера пробную матрицу степени 3:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

В свете этого соглашения нас не должен удивлять следующий результат:

```
In[184]:= {{{a,b},{c,d}}+x,{{a,b},{c,d}}*x}
Out[184]={{{a+x,b+x},{c+x,d+x}},{{a*x,b*x},{c*x,d*x}}}
```

• **Генерация векторов и матриц.** Основной командой генерации векторов, матриц и других списков в языке Mathematica является команда Table. Вызванная с одним итератором команда Table[f[i],{i,m,n}] порождает список значений функции f в точках i = m,m + 1,... ,n. Вот типичный пример использования этой команды:

```
In[185]:= Table[x^i/i!,{i,0,5}]
Out[185]={1,x,x^2/2,x^3/6,x^4/24,x^5/120}
```

А вот забавная вариация на тему этого примера, детально обсуждаемая Анри Пуанкаре в “Новых методах небесной механики”:

```
In[186]:= Table[N[10^i/i!],{i,1,30}]
Out[186]={10., 50., 166.667, 416.667, 833.333, 1388.89, 1984.13,
2480.16, 2755.73, 2755.73, 2505.21, 2087.68, 1605.9,
1147.07, 764.716, 477.948, 281.146, 156.192, 82.2064,
41.1032, 19.5729, 8.89679, 3.86817, 1.61174, 0.644695,
0.24796, 0.0918369, 0.0327989, 0.01131, 0.00376999}
```

Из этой таблицы видно, что отношение 10i/i! вначале очень быстро растет (“расходится в смысле астрономов”), а потом очень быстро убывает (“сходится в смысле математиков”).

Вызванная с двумя итераторами команда

Table[f[i,j],{i,k,l},{j,m,n}] порождает вложенный список значений функции двух аргументов f в парах (i, j), где i = k, k + 1, . . . , l, j = m, m + 1, . . . , n. Этот список будет организован как матрица, причем итератор i считается внешним, а j — внутренним, иными словами, i нумерует строки, а j — позиции внутри строк. Таким образом, строки этой матрицы имеют вид (fim, . . . , fin).

Еще раз обратите внимание на следующие два ключевых момента в этом определении:

◦ матрица трактуется как строка, составленная из строк,

◦ внутренние итераторы пишутся последними.

Иными словами, если мы дадим два определения матрицы Forward

```
In[187]:= forwaa[n ]:=Table[If[j==i+1,1,0],{i,1,n},{j,1,n}]
In[188]:= forwbb[n ]:=Table[If[j==i+1,1,0],{j,1,n},{i,1,n}]
```

отличающиеся лишь порядком итераторов, то получившиеся матрицы будут транспонированы друг к другу. Вот как, например, выглядят матрицы forwaa[4] и forwbb[4] в традиционной математической нотации

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Иными словами, только одна из этих матриц в действительности окажется матрицей Forward, вторая будет матрицей Backward! Ну а, скажем, упомянутая в предыдущем пункте пробная матрица задается командой

test[n]:=Table[1,{i,1,n},{j,1,n}] .

В системе Mathematica существует большое количество других команд генерации списков и мтариц специального вида. Одной из таких команд является DiagonalMatrix[{d1,...,dn}], которая порождает диагональную матрицу с диагональными элементами d1,... ,dn. Однако в тех простых ситуациях, которые мы здесь рассматриваем, вполне достаточно команды Table. Кроме того, как мы обсуждаем ниже, матрицы можно создавать из векторов или, наоборот, более глубоких списков при помощи структурных команд таких, как Partition или Flatten.

• **Части матриц**. Для выделения частей списков в языке Mathematica используется специальный вид скобок — двойные квадратные скобки [[]], являющиеся сокращением команды Part. При этом в соответствии с общими правилами спецификации уровня, детально обсуждаемыми в Модуле 2, x[[i]] — или, что то же самое, Part[x,i] — обозначает i-ю часть списка x; x[[-i]] — i-ю часть с конца; x[[i,j]] — j-ю часть его i-части и т.д. При этом сами номера позиций тоже могут задаваться списком, а если мы хотим выбрать все части на каком-то уровне, то спецификацией All.

Приведем несколько примеров применения этих соглашений

◦ x[[i,j]] — элемент x в позиции (i,j);
◦ x,[[{i,j},{k,l}]] — подматрица порядка 2 матрицы x, стоящая на пересечении строк с номерами i,j и столбцов с номерами k, l;
◦ x[[i]] — i-я строка матрицы x;
◦ x[[All,j]] — j-й столбец матрицы x;
◦ Tr[x,List] — главная диагональ матрицы x;

Так, например, x[[1]] и x[[-1]] обозначают, соответственно, первую и последнюю строку матрицы x, а x[[All,1]] и x[[All,-1]] — ее первый и последний столбец. Еще раз обратите внимание на отсутствие здесь симметрии между строками и столбцами! Столбец матрицы является строкой транспонированной матрицы, поэтому j-й столбец матрицы x можно породить также посредством Transpose[x][[j]].

Разумеется, также и главную диагональ матрицы можно определять тысячей других способов, например, напрашивающимся

Table[x[[i,i]],{i,1,Length[x]}].

Однако встроенная команда Tr[x,List] не только изящнее, но и работает быстрее, хотя разница во времени становится по настоящему заметной только для матриц порядка несколько десятков тысяч.

• **Просмотр матриц.** Чтобы увидеть матрицу не как вложенный список, а в традиционной записи, надо применить к ней одну из команд TableForm или MatrixForm. Скажем, следующая строка определяет одну из самых важных в самой математике и ее приложениях матриц — матрицу Вандермонда V (x1,... ,xn). Собственно говоря, вся теория определителей построена исключительно для анализа этого примера:

In[189]:= vandermonde[x ,n]:=Table[x[j]^i,{i,0,n-1},{j,1,n}]

Посмотрим теперь, как выглядит матрица vandermonde[y,5]. Это делается при помощи команды MatrixForm. Вычислив

In[190]:= MatrixForm[vandermonde[y,5]]

мы увидим примерно следующий результат:

$$V(y_1, y_2, y_3, y_4, y_5) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ y_1 & y_2 & y_3 & y_4 & y_5 \\ y_1^2 & y_2^2 & y_3^2 & y_4^2 & y_5^2 \\ y_1^3 & y_2^3 & y_3^3 & y_4^3 & y_5^3 \\ y_1^4 & y_2^4 & y_3^4 & y_4^4 & y_5^4 \end{pmatrix}$$

Разумеется, здесь мы воспроизводим не то, что будет фактически отображено в записной книжке Mathematica, а уже облагороженный TEX'овский аутпут типографского качества, получающийся с помощью применения поверх MatrixForm команды экспорта TeXForm.

Мы сами обычно не печатаем форматные команды TableForm, Matrix-

Form и т.д. заранее, а применяем их к уже имеющемуся выражению в постфиксной форме (задним числом). Это делается при помощи оператора

MatrixForm

В первом приближении различие между командами TableForm и MatrixForm такое же, как между matrix и pmatrix в TEX'e. Иными словами, TableForm[x] выводит элементы матрицы x в виде таблицы, а MatrixForm[x] —в традиционной математической записи с использованием круглых скобок.

Комментарий. В действительности, кроме этого очевидного отличия, между командами TableForm и MatrixForm имеется большое количество тонких различий в

том, как они по умолчанию трактуют горизонтальное (слева, сверху, по центру, по десятичной точке, по фиксированному символу и т.д.) и вертикальное (по верху, по низу, по центру, по базовой линии, по оси) выравнивание строк и столбцов и индивидуальных элементов внутри строки или столбца. Однако для большинства пользователей, которые интересуются результатом вычисления, а не получением типографского продукта профессионального или полупрофессионального качества, подобные тонкости не имеют большого значения.

• **Матричные единицы**. Самыми важными с точки зрения профессионального математика матрицами являются стандартные матричные единицы eij, у которых в позиции (i,j) стоит 1 и 0 во всех остальных местах. Матрицы eij, 1 ≤ i,j ≤ n, образуют базис M(n,K) как векторного пространства над K. Более того, это мультипликативный базис, в том смысле, что произведение двух базисных элементов либо равно 0, либо снова представляет собой базисный элемент: eijehk = δjheik. Таким образом, eijehk = 0, если j 踊= h, в то время как eijejk = eik. Вот одно из возможных определений этих матриц в языке Mathematica. Конечно, теперь мы должны явно указывать не только i,j, но и порядок n:

In[192]:= e[n ,i ,j]:=Table[If[h==i,1,0]*If[k==j,1,0],
{h,1,n},{k,1,n}]

Еще раз обратите внимание на несколько уже встречавшихся нам принципиальных моментов:

◦ использование Blank и := SetDelayed для определения функции;

◦ использование == Equal в уравнениях i==h и j==k;

◦ использование условного оператора If[condition,x,y], возвращающего x, если condition==True и y, если condition==False.

Комментарий. Как мы обсуждаем в дальнейшем, в общем случае намного разумнее вызывать оператор If с четырьмя аргументами, в формате If[condition,x,y,z], явным образом предписывая, что следует делать в случае, когда система не может решить, выполняется условие condition или нет. Однако здесь мы используем этот оператор в чисто иллюстративных целях и только в простейшей ситуации сравнения небольших целых чисел, когда у системы не должно быть никаких сомнений, равны они или нет. В действительности вместо If[i==h,1,0] мы могли бы воспользоваться встроенной функцией KroneckerDelta, но это чуть длиннее.

Посмотрим теперь, как выглядят стандартные матричные единицы степени 3. Применив к ним форматную команду MatrixForm

In[193]:= Map[MatrixForm,Flatten[Table[e[3,i,j],{i,1,3},{j,1,3}],1]]

мы увидим следующее:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Стоит подчеркнуть здесь один момент, не встречавшийся до сих пор. Дело в том, что команда Table[e[3,i,j],{i,1,3},{j,1,3}] создает вовсе не список из девяти стандартных матричных единиц, а вложенный список list, состоящий из трех списков в каждом из которых три матричные единицы! Таким образом, применение команды MatrixForm непосредственно к элементам этого списку даст нам совершенно не то, что хотелось.

Команда Flatten[list,1] служит как раз для того, чтобы убрать один лишний вложенный уровень в list, а именно, верхний, и создать список из девяти матричных единиц (каждая из которых, конечно, сама является матрицей, т.е. вложенным списком!!) Применение команды Flatten[list] не достигло бы этой цели, так как убрало бы все вложенные уровни, создав

список из 81 нуля и единицы.

Линейная алгебра

В этом пункте мы опишем как проводить некоторые простейшие матричные вычисления. Как правило, для наглядности мы будем приводить не фактический аутпут системы Mathematica, а традиционную форму матриц, которая получается, если применить к получающимся матрицам MatrixForm, а потом экспортировать результат в TEX'овском формате посредством TeXForm.

• **Решение систем линейных уравнений**. Решение систем линейных уравнений осуществляется при помощи функций LinearSolve и NullSpace. А именно, функция LinearSolve ищет частное решение неоднородной системы линейных уравнений, а функция NullSpace — общее решение однородной системы.

Функция Linear Solve вызывается в различных форматах, в зависимости от того, нужно ли нам решить одну линейную систему ax = b с данной матрицей a или (как весьма часто бывает в приложениях!) несколько систем с одной и той же матрицей и различными правыми частями. В первом случае функция LinearSolve вызывается в формате LinearSolve[a,b] в то время как во втором случае — в формате LinearSolve[a][b].

Разница состоит в том, что во втором случае система записывает факторизацию матрицы a, которая используется для решения системы ax = b.

Если матрица a не обратима, то при вызове команды LinearSolve в таком формате выдается сообщение:

LinearSolve: The matrix bla-bla-bla is singular
so a factorization will not be saved.

Приведем пример использования функции LinearSolve. Следующие команды определяют случайную матрицу порядка m на n с целыми коэффициентами между −100 и 100 и случайный вектор высоты n с целыми компонентами в том же интервале:

In[195]:= rama[m ,n]:=Table[Random[Integer,{-100,100}],
{i,1,m},{j,1,n}]
In[196]:= cora[n]:=Table[Random[Integer,{-100,100}],{i,1,n}]

А именно, вызванная в формате Random[Integer,{k,l}], команда Random генерирует случайное целое число x, k ≤ x ≤ l, естественно, каждый раз новое.

Следующая команда фактически генерирует случайную 4×4-матрицу и случайный вектор высоты 4:

In[197]:= nnn=4;aaa=rama[nnn,nnn];bbb=cora[nnn];

Вот так выглядит типичный получающийся при этом результат

$$a = \begin{pmatrix} 66 & -83 & 1 & 63 \\ -25 & -10 & -49 & 99 \\ 74 & 74 & -31 & -6 \\ -92 & 62 & 44 & 70 \end{pmatrix}, \qquad b = \begin{pmatrix} 13 \\ 19 \\ -79 \\ -8 \end{pmatrix}.$$

Определитель такой случайной целочисленной матрицы грандиозен (например, в данном случае он равен 123258840). Поэтому применение команды LinearSolve дает

In[198]:= LinearSolve[aaa][bbb]
Out[198]={-1056071/2054314,-3961079/6162942,
-597458/3081471,-202933/2054314}

В то же время команда NullSpace возвращает какую-то фундаментальную систему решений уравнения ax = 0. Вот, к примеру, что получается при применении этой команды к уже встречавшейся нам в предыдущем параграфе матрице test[4]:

In[199]:= NullSpace[test[4]]
Out[199]={{-1,0,0,1},{-1,0,1,0},{-1,1,0,0}}

Часто хочется увидеть общее решение неоднородной системы, совмещающее ее частное решение и общее решение соответствующей неоднородной системы. Для

этого нужно свести вместе результаты работы команд LinearSolve и NullSpace. Традиционную форму такого общего решения

можно породить, например, при помощи следующей конструкции:

```
In[200]:= gensolution[g ,b ]:=StringForm["``+``",
MatrixForm[LinearSolve[g,b]],
Table[a[i],{i,1,Length[NullSpace[g]]}].
Map[MatrixForm,NullSpace[g]]]
```

Здесь использована одна из важнейших команд форматирования вывода StringForm. Мы часто пользуемся этой командой в тех случаях, когда нам нужно включить результат вычисления в текст. Эта команда служит для включения результатов вычисления в текстовый объект и вызывается вследующем формате:

StringForm["ccc``ccc``ccc...",expression1,expression2,...] где, как мы уже знаем, заключенный в двойные кавычки "..." объект рассматривается как стринг, обозначает произвольную комбинацию знаков, а каждая пара аксанов `` заменяется на значение очередного выражения expression.

Теперь вычисление

In[201]:= gensolution[test[4],{1,1,1,1}] даст нам следующий результат

$$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+a_1\begin{pmatrix}-1\\0\\0\\1\end{pmatrix}+a_2\begin{pmatrix}-1\\0\\1\\0\end{pmatrix}+a_3\begin{pmatrix}-1\\1\\0\\0\end{pmatrix}.$$

• **Умножение матриц.** В отличие от покомпонентного умножения *или, в полной форме Times, обычное матричное умножение представляющее композицию линейных отображений, обозначается . или, в полной форме Dot. Таким образом, x.y или Dot[x,y] выражает произведение матриц x и y. Например,

```
In[202]:= {{a,b},{c,d}}.{{e,f},{g,h}}
Out[202]={{a*e+b*g,a*f+b*h},{c*e+d*g,c*f+d*h}}
```

Чтобы проиллюстрировать матричные вычисления на содержательном примере, свяжем с каждой функцией f матрицу Тэйлора:

```
In[203]:= taylor[f ,x ,n ]:=Table[If[i<=j,D[f,{x,j-i}]/(j-i)!,0],{i,1,n+1},{j,1,n+1}]
```

Взглянем на матрицу Тэйлора степени 5 — с точки зрения анализа, где нумерация производных начинается с нуля, эта матрица является матрицей Тэйлора порядка 4:

$$T(f,4)=\begin{pmatrix} f(x) & f'(x) & f''(x)/2 & f'''(x)/6 & f''''(x)/24 \\ 0 & f(x) & f'(x) & f''(x)/2 & f'''(x)/6 \\ 0 & 0 & f(x) & f'(x) & f''(x)/2 \\ 0 & 0 & 0 & f(x) & f'(x) \\ 0 & 0 & 0 & 0 & f(x) \end{pmatrix}$$

Вычислим теперь произведение двух матриц Тэйлора:

In[204]:= Simplify[taylor[f[x],x,4].taylor[g[x],x,4]]//MatrixForm

Мы не воспроизводим результат этого вычисления ввиду его громоздкости, но у каждого, кто когда-нибудь видел формулу Лейбница дифференцирования произведения, в этот момент закрадываются сильные подозрения.

После следующего вычисления

In[205]:= Timing[Simplify[taylor[f[x],x,50].taylor[g[x],x,50]]==
taylor[f[x]*g[x],x,50]]
Out[205]={3.034,True}

эти подозрения незамедлительно переходят в уверенность. В самом деле, две матрицы степени 51 не могут совпасть по случайной причине! Это значит, что от нас в курсе анализа скрывали нечто весьма существенное, а именно то, что многочлен Тэйлора n-го порядка произведения двух функций равен произведению их многочленов Тэйлора того же порядка:

T(fg, n) = T(g, n)T(g, n).

Разумеется, многочлен Тэйлора n-го порядка следует здесь понимать с точностью до бесконечно малых более высокого порядка или, как сказал бы алгебраист, по модулю x^{n+1}.

То, что мы сейчас увидели — это как раз типичный пример грамотного использования Mathematica. А именно, опытный пользователь спрашивает у системы то, что он действительно хочет узнать, в данном случае, равна ли матрица Тэйлора функции fg произведению матриц Тэйлора функций f и g. Но для этого совершенно не обязательно фактически смотреть на сами эти матрицы или их произведение!! Неумелое использование, которым грешат не только начинающие, но и многие авторы учебных текстов, состоит в том, чтобы показывать то, что с точки зрения окончательной цели является промежуточным результатом. Например, выводить на экран матрицу taylor[f[x],x,50].taylor[g[x],x,50].

• **Обращение матриц**. Обратная к g матрица g−1 вычисляется при помощи функции Inverse. Прежде всего проверим, что мы правильно понимаем, в каком формате вызывается Inverse:

```
In[206]:= Inverse[{{a,b},{c,d}}]
Out[206]={{d/(-b*c+a*d),-b/(-b*c+a*d)},
{-c/(-b*c+a*d),a/(-b*c+a*d)}}
```

Проиллюстрируем теперь использование этой функции на содержательном примере. Следующая трехдиагональная матрица является одной из самых знаменитых и важных в математике. Чистым математикам она известна как матрица Картана, а прикладным — как матрица конечных разностей:

```
In[207]:= cartan[n ]:=Table[Which[i==j,2,Abs[i-j]==1,-1,True,0],
{i,1,n},{j,1,n}]
```

Обратите внимание на использование для задания этой матрицы условного оператора Which. Встречавшийся нам ранее оператор If[test,x,y,z], выбирает одну из трех возможностей x,y,z в зависимости от трех значений истинности теста test, в следующем порядке: True, False, Undecided. В отличие от него оператор Which вызывается в формате

```
Which[test1,x1,test2,x2,test3,x3,...]
```

Этот оператор поочередно оценивает истинность каждого из тестов test1, test2, test3, ... и возвращает xi для первого из них, который принимает значение True. В нашем примере последний тест всегда выдает значение True, так что коэффицент матрицы в позиции (i,j) равен 0, если пара (i,j) не проходит ни одного из предыдущих тестов. Примерно так же работает и переключатель Switch, но, в отличие от условного оператора Which, он вызывается в другом формате и проверяет не прохождение теста, а совпадение выражения с одной из предписанных форм или его соответствие предписанному паттерну. Так, скажем, Switch[Mod[i-j,4],0,a,1,b,2,c,3,d] принимает значение a,b,c или d в соответствии с тем, чему равен вычет $i - j$ по модулю 4.

Посмотрим для примера на матрицу cartan[8]:

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

Обратная к этой матрице хорошо известна и очень часто используется.

Изобразим для примера 9*Inverse[cartan[8]]:

$$\begin{pmatrix} 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\ 6 & 12 & 18 & 15 & 12 & 9 & 6 & 3 \\ 5 & 10 & 15 & 20 & 16 & 12 & 8 & 4 \\ 4 & 8 & 12 & 16 & 20 & 15 & 10 & 5 \\ 3 & 6 & 9 & 12 & 15 & 18 & 12 & 6 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

• **Определитель матрицы.** Определитель матрицы g вычисляется при помощи функции Det. Как всегда, прежде всего стоит проверить на совсем простом примере, правильно ли мы понимаем ее использование:

In[208]:= Det[{{a,b},{c,d}}]

Out[208]= -b*c+a*d

Теория определителей была придумана для анализа единственного примера — определителя Вандермонда det(V (x1,. .. ,xn)). Так вот с него и начнем. Вычисление Length[Det[vandermonde[y,5]]] показывает, что Mathematica считает, что этот определитель является суммой 120 слагаемых Однако применяя к тому же определителю команду Simplify, мы получим уже вполне осмысленный результат:

In[209]:= Simplify[Det[vandermonde[y,5]]]

Out[209]= (y[1]-y[2])*(y[1]-y[3])*(y[2]-y[3])*(y[1]-y[4])*(y[2]-y[4])*

```
(y[3]-y[4])*(y[1]-y[5])*(y[2]-y[5])*(y[3]-y[5])*(y[4]-y[5])
```

Команда Minors[x] порождает все дополнительные миноры матрицы x. Однако упорядочивает она их лексикографически по порядку включенных в них строк и столбцов (а вовсе не исключенных, как это делается

в элементарных учебниках линейной алгебры!) Следующее вычисление показывает дополнительные миноры в матрице степени 3:

```
In[210]:= fancy1=Partition[CharacterRange["a","i"],3]
Out[210]={{a,b,c},{d,e,f},{g,h,i}}
In[211]:= Minors[fancy1]
Out[211]={{-b*d+a*e,-c*d+a*f,-c*e+b*f},
{-b*g+a*h,-c*g+a*i,-c*h+b*i},
{-e*g+d*h,-f*g+d*i,-f*h+e*i}}
```

СПИСОК ЛИТЕРАТУРЫ:

1. Вавилов Н. А.Математика для нематематика. Москва. 2021 стр 132-147.
2. Саченков О.А. «математика методичка» 2016 г.
3. Климов В. С., Ухалов А. Ю. «Решение задач математического анализа с использованием систем компьютерной математики», Ярославль, 2014г
4. https://cyberleninka.ru/article/n/o-reshenii-zadach-matematicheskoy-fiziki-v-sisteme-wolfram-mathematica