5. Матричные и векторные вычисления

Среда Maple позволяет выполнять все стандартные операции, определенные в линейной алгебре. Они становятся доступны при подключении библиотеки linalg:

```
> with(linalg):
```

Для определения матрицы (вектора) используются команды **matrix** **(vector)**.

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);
a:=vector([2,x^2,4,5.3,alpha]);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}$$

$$a := [2, x^2, 4, 5.3, \alpha]$$

Известно, что в Maple (как и во многих других программных продуктах) под матрицей понимается двумерный массив, индекс которого изменяется от единицы до любого целого числа. Следовательно, матрицу также можно задать следующим образом:

```
> AA:=array(1..2,1..3);
```

$$AA := \mathrm{array}(1\,..\,2, 1\,..\,3, [])$$

Но массив вида array(1..2,0..3) матрицей не является, т. к. второй индекс изменяется от нуля.

Имеются богатые возможности для формирования матриц специального вида. Например, сформировать единичную матрицу можно следующим образом:

```
> E:=array(identity,1..3,1..3): evalm(E);
```

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

Для получения диагональной матрицы используют команду **diag(vec)**, где **vec** - вектор, расположенный на главной диагонали. Заметим, что если **vec** - единичный вектор, то полученная матрица так же будет единичной.

```
> De:=diag(1,2,3);
```

$$De := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Матрица из случайных чисел генерируется командой **randmatrix(n,m,opt)**, где **n,m** - размерность матрицы, а **opt** - параметр, определяющий тип матрицы (**symmetric, antisymmetric, unimodular** и др.).

```
> randmatrix(4,4,symmetric);
```

$$\begin{bmatrix} -85 & -55 & -35 & 79 \\ -55 & -37 & 97 & 56 \\ -35 & 97 & 50 & 49 \\ 79 & 56 & 49 & 63 \end{bmatrix}$$

Команда **blockmatrix** определяет блочную, а команда **hilbert** - гильбертову матрицы.

Чтобы узнать количество строк или столбцов необходимо выполнить следующие операции:

```
> rowdim("); coldim(A);
```

$$4$$

$$3$$

В библиотеке “linalg” предусмотрены различные преобразования над матрицами: транспонирование, вычисление обратной матрицы, сопряженной матрицы, взятие минора, вычисление ядра и др. Для этого используются соответствующие команды: **transpose(A), inverse(A), adjoint(A), minor(A,n,m), kernel(A)**.

```
> AT:=transpose(A);  minor(A, 2,1);
```

$$AT := \begin{bmatrix} 1 & 4 & 7 \\ 1 & 1 & 1 \\ 1 & 6 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 9 \end{bmatrix}$$

Для того, чтобы выделить часть матрицы (вектора) существуют команды

submatrix(A,i1..i2,j1..j2) и subvector(a,i1..i2), а если необходимо выделить i-тую строчку или i-ый столбец используют **row(A,i)** или **col(A,i)** соответственно.

```
> submatrix(A,1..2,2..3); row(",2);
```

$$\begin{bmatrix} 1 & 1 \\ 1 & 6 \end{bmatrix}$$

$$[1, 6]$$

Строки или столбцы удаляются командами **delcols(A,i1..i2)** или **delrows(A,i1..i2)** соответственно.

Склеить несколько матриц горизонтально можно используя функцию **concat(A1,..,A2)**, а вертикально - **stack(A1,..,A2)**:

```
> S:=concat(A,E,De);
```

$$S := \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 6 & 0 & 1 & 0 & 0 & 2 & 0 \\ 7 & 1 & 9 & 0 & 0 & 1 & 0 & 0 & 3 \end{bmatrix}$$

Для вычисления определителя, ранга, числа обусловленности, следа матрицы используются следующие команды соответственно:

```
> det(A); rank(S); cond(A); trace(A);
```

Результат будет иметь следующий вид:

6;

3;

136/3;

11.

Команда **multiply(A1,..,An)** служит для перемножения матриц (векторов). Следует заметить, что операции "+", "-", "*" можно выполнить и традиционным способом, при этом знак "*" используется при умножении скалярной функции или числа на матрицу, а когда в произведении участвуют только вектора или матрицы, то он заменяется на "&*":

```
> Su:=evalm(A&*De+E-x*multiply(A,E));
```

$$Su := \begin{bmatrix} 2-x & 2-x & 3-x \\ 4-4\,x & 3-x & 18-6\,x \\ 7-7\,x & 2-x & 28-9\,x \end{bmatrix}$$

Команды **addrow(A,i,j,m) и addcol(A,i,j,m)** служат для умножения i-той строки (столбца) на скалярный множитель m и прибавления к j-той строке (столбцу).

```
> addrow(Su,1,2,10);
```

$$\begin{bmatrix} 2-x & 2-x & 3-x \\ 24-14\,x & 23-11\,x & 48-16\,x \\ 7-7\,x & 2-x & 28-9\,x \end{bmatrix}$$

Т.е. первая и третья строки остаются без изменений, а на место второй строки записывается выражение m*row(A,i) + row(A,j).

```
> row2:=evalm(10*row(Su,1)+row(Su,2));
```

row2 := [24 - 14 x, 23 - 11 x, 48 - 16 x]

Вычислим матрицу **B,** исходя из формулы: B=A-E* λ , используя команду `diag(1,1,1)` для создания единичной матрицы 3x3.

```
> B:=evalm(A-lambda*diag(1,1,1));
```

$$B := \begin{bmatrix} 1-\lambda & 1 & 1 \\ 4 & 1-\lambda & 6 \\ 7 & 1 & 9-\lambda \end{bmatrix}$$

Найдем определитель матрицы B:

```
> f:=det(B);
```

$$f := 6 - 2\,\lambda + 11\,\lambda^2 - \lambda^3$$

Это выражение будет являться характеристическим полиномом матрицы A. Его можно найти, выполнив всего одну команду:

```
> charpoly(A,lambda);
```

$$2\,\lambda - 11\,\lambda^2 + \lambda^3 - 6$$

Теперь найдем собственные значения матрицы A, решая ее характеристическое уравнение относительно λ .

```
> res:=evalf(solve(f,lambda),4);
```

$$res := 10.87,\ .066 + .7395\,I,\ .066 - .7395\,I$$

Последний результат можно найти, проделав лишь одну операцию над матрицей A:

```
> evalf(eigenvals(A),4);
```

$$10.87,\ .066+.7395\,I,\ .066-.7395\,I$$

Получим собственные векторы матрицы A :

```
> evalf(eigenvects(A,'radical'),4);
```

$$[10.87,\ 1.,\ \{[.27,\ 1.,\ 1.46]\}\ ],$$
$$[.066+.7395\,I,\ 1.,\ \{[-.2180-1.063\,I,\ 1.,\ -.01168+.8315\,I]\}\ ],$$
$$[.066-.7395\,I,\ 1.,\ \{[-.2180+1.063\,I,\ 1.,\ -.01168-.8315\,I]\}\ ]$$

Здесь результат выдается в форме:

[num,r,{vect}],

где **num** – собственное значение матрицы;

r – кратность собственного значения;

vect – собственный вектор;

'*radical*' – ключ, определяющий режим нахождения всех собственных значений.

С помощью библиотеки **linalg** можно привести матрицу к различным специальным формам.

Команда **gausselim(A)** используется для приведения к треугольному виду.

Алгоритм гауссова исключения без деления реализуется функцией **ffgausselim(A)**.

```
> ffgausselim(A);
```

$$\begin{bmatrix} 1 & 0 & -\frac{33}{7}-\frac{4}{7}x^2+\frac{22}{7}x \\ 0 & 1 & -\frac{2}{7}x^2+\frac{1}{7}+\frac{4}{7}x \\ 0 & 0 & x^3-\frac{19}{3}x^2+\frac{91}{6}x-\frac{85}{6} \end{bmatrix}$$

К треугольному виду привести матрицу можно так же при помощи алгоритма Гаусса-Жордана - **gaussjord**.

Команда **jordan(M)** приводит матрицу к жордановой форме, а **hermite(A,x)** - к эрмитовой форме, элементы которой зависят от переменной **x.**

```
> hermite(Su,x);
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -3 & 2 \\ 0 & 0 & 6 \end{bmatrix}$$

Можно воспользоваться так же дифференциальными операторами. Например, что бы вычислить градиент (дивергенцию) функции **`f(F)`**, зависящей от переменных вектора **x**, необходимо воспользоваться командами **`grad(f,x)`** или **`diverge(F,x)`**.

Функция **`curl(v,x)`** определяет ротор трехмерного вектора **`v`** по трем переменным вектора **`x`**.

Матрицу Якоби для вектора **`v`** по переменным вектора **`x`** вычисляют при помощи команды **`jacobian(v,x)`**, а лапласиан функции **`f`** по переменным вектора **`x`** - **`laplacian(f,x)`**. Например:

```
> f:=x^3+y^2+cos(z+x);
```

$$f := x^3 + y^2 + \cos(z + x)$$

```
> gr:=grad(f,[x,y,z]); laplacian(f,[x,y,z]);
```

$$gr := [3\,x^2 - \sin(z + x),\, 2\,y,\, -\sin(z + x)]$$

$$6\,x - 2\cos(z + x) + 2$$

```
> diverge(gr,[x,y,z]); jacobian(gr,[x,y,z]);
curl(gr,[x,y,z]);
```

$$6\,x - 2\cos(z + x) + 2$$

$$\begin{bmatrix} 6\,x - \cos(z + x) & 0 & -\cos(z + x) \\ 0 & 2 & 0 \\ -\cos(z + x) & 0 & -\cos(z + x) \end{bmatrix}$$

$$[0, 0, 0]$$

Для решения матричного уравнения AX=B используется команда **`linsolve(A,B)`**, где **`A`** - матрица, **`X, B`** - матрицы или векторы.

```
> A:=matrix([[-1,2],[-3,4]]):
  B:=matrix([[5,-6],[7,-8]]):
```

```
> linsolve(A, B);
```

$$X := \begin{bmatrix} 3 & -4 \\ 4 & -5 \end{bmatrix}$$

Приведем основные из доступных операций с кратким описанием:

команда	описание
addcol(A,i,j,m)	Умножение **j**-го столбца матрицы **A** на скалярный множитель **m** и прибавление к **i**-му столбцу матрицы **A**
addrow(A,i,j,m)	Умножение **j**-ой строки матрицы **A** на скалярный множитель m и прибавление к **i**-ой строке матрицы **A**
adjoint(A)	Определение сопряженной матрицы **A**
blockmatrix	Определение блочной матрицы
charpoly(A,lambda)	Определение характеристического многочлена матрицы **A** относительно **lambda**
col(A,i)	Выделение **i**-го столбца матрицы **A**
coldim(A);	Определение числа столбцов матрицы **A**
concat(A1,..,A2)	Склеивание нескольких матриц (с **A1** по **A2**) горизонтально
cond(A)	Определение числа обусловленности матрицы **A**
curl(v,x)	Определение ротора трехмерного вектора **v** по трем переменным вектора **x**
delcols(A,i1..i2)	Удаление строк с номерами **i1** по **i2**
delrows(A,i1..i2)	Удаление столбцов с номерами **i1** по **i2**
det(A)	Нахождение определителя **A**
diag(vec)	Определение диагональной матрицы, где **vec** - вектор, расположенный на главной диагонали
diverge(F,x)	Определение дивергенции функции **F**, зависящей от набора переменных вектора **x**
eigenvals(A)	Определение собственных значений матрицы **A**
eigenvects(A)	Определение собственных векторов матрицы **A**
ffgausselim(A)	Применение алгоритма гауссова исключения без деления матрицы **A**
gausselim(A)	Приведение матрицы **A** к треугольному виду
gaussjord(A)	Приведение матрицы **A** к треугольному виду при помощи алгоритма Гаусса-Жордана

команда	описание
grad(f,x)	Определение градиента функции **f**, зависящей от набора переменных вектора **x**
hermite(A,x)	Приведение матрицы **A** к эрмитовой форме, элементы которой зависят от переменной **x**
hilbert(A)	Определение гильбертовой матрицы **A**
inverse(A)	Нахождение обратной матрицы **A**
jacobian(A,x)	Определение матрицы Якоби для вектора **v** по переменным вектора **x**
jordan(A)	Приведение матрицы **A** к жордановой форме
kernel(A)	Определение ядра матрицы **A**
laplacian(f,x)	Определение лапласиана функции **f** по переменным вектора **x**
linsolve(A,B)	Решение матричного уравнения AX=B, где **A** - матрица, **X,B** - матрица или вектор.
matrix(A)	Определение матрицы **A**
minor(A, i, j)	Распечатка минора матрицы **A**, отвечающего элементу, стоящему в **i**-ой строке, **j**-ом столбце
multiply(A1,..,A2)	Перемножение матриц с **A1** по **A2**
randmatrix(n, m, opt)	Определение матрицы из случайных чисел, где **n,m** - размерность матрицы, а **opt** - параметр, определяющий тип матрицы (*symmetric, antisymmetric, unimodular* и др.)
rank(A)	Определение ранга матрицы **A**
row(A,i)	Выделение **i**-ой строчки матрицы **A**
rowdim(A)	Определение числа строк матрицы **A**
stack(A1,..,A2)	Склеивание нескольких матриц (с **A1** по **A2**) вертикально
submatrix(A,i1..i2,j1..j2)	Выделение части матрицы (с **i1** по **i2** элемента строки, с **j1** по **j2** элемента столбца)
subvector(a,i1.. i2)	Выделение части вектора (с **i1** по **i2** элемента)
trace(A)	Определение следа матрицы **A**
transpose(A)	Транспонирование матрицы **A**
vector(B)	Определение вектора **B**