Линейная алгебра и векторный анализ в Maple

1. Векторная алгебра.
2. Действия с матрицами.
3. Спектральный анализ матрицы.
4. Системы линейных уравнений. Матричные уравнения.
5. Дифференциальные операции векторного анализа: градиент, дивергенция, ротор. Лапласиан и якобиан.

Команды для решения задач линейной алгебры содержатся в пакетах **`LinearAlgebra`** и **`VectorCalculus`**, а также в пакете **`linalg`** (был единственным доступным пакетом для старых версий Maple). Данные пакеты имеют сходный набор основных команд, различия между соответствующими командами состоит в их синтаксисе. В современных версиях Maple пакет **`linalg`** считается устаревшим, рекомендуется пользоваться пакетами **`LinearAlgebra`** и **`VectorCalculus`**.

Всюду далее будет описана работа с командами пакетов **`LinearAlgebra`** и **`VectorCalculus`**, а также будут приведены аналогичные команды пакета **`linalg`** (мелким шрифтом). Примеры работы пакета Maple будут приведены в режиме интерфейса Worksheet Mode с режимами ввода Text Mode (1D-Math) и Math Mode (2D-Math).

Перед решением задач с матрицами и векторами следует загрузить интересующий пакет командой **`with(LinearAlbebra)`** для пакета **`LinearAlgebra`** или **`with(linalg)`** для пакета **`linalg`**.

§1. Векторная алгебра

Способы задания векторов.

Для последующей работы с командами пакета **`LinearAlgebra`** вектор в Maple можно задать двумя способами:

1) С помощью угловых скобок
   а) Для задания вектор-столбца используется последовательность выражений, отделенных друг от друга запятыми.

```
> <1,2,3>
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

   б) Для задания вектор-строки выражения отделяются друг от друга вертикальной чертой.

```
> <1|2|3>
```

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

2) С помощью команды-конструктора **`Vector([x1,x2,…,xn])`**, где в квадратных скобках через запятую указываются координаты вектора. Например:
   а) Вектор-столбец

```
> v1 := Vector([1, 2, 3]);
```

$$v1 := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

   б) Вектор-строка (используется опция row), например

```
> Vector[row](3, fill = 1);
```

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

Задание элементов вектора через функцию

```
> f := j -> x^j : Vector(3, f)
```

$$\begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

Использование индексирующей функции

```
> Vector(3, shape = scalar[2, 100])
```

$$\begin{bmatrix} 0 \\ 100 \\ 0 \end{bmatrix}$$

Для последующей работы с командами пакета **linalg** вектор можно задать либо с помощью команды **array**, либо с помощью команды **vector** из пакета **linalg**:

```
> a := array(1..4, [1, 2, 3, 4]);
```

$$a := \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

```
> with(linalg):
> v:=vector([1,2,3]);
```

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

Координату уже определенного вектора **v** можно получить в строке вывода, если ввести команду **v[i]** , где **i** – номер координаты. Например, первую координату заданного в предыдущем примере вектора можно вывести так:

```
> v[1];
```

1

Отрицательное целое число в качестве номера координаты указывает номер координаты с конца вектора.

```
> a := <85.3, 47.1, 59.9, 38.1>
```

$$a := \begin{bmatrix} 85.3 \\ 47.1 \\ 59.9 \\ 38.1 \end{bmatrix}$$

```
> a[-1]
```

38.1

Сложение векторов.

Сложение векторов, заданных с помощью угловых скобок <> или с помощью команды **Vector** выполняется с помощью обычного знака сложения **+**.

Сложить два вектора ***a*** и ***b*** , заданных с помощью команды **vector** пакета **linalg,** можно с помощью двух команд пакета **linalg**:

1) **evalm(a+b);**

2) **matadd(a,b).**

Команда **matadd** позволяет вычислять линейную комбинацию векторов ***a*** и ***b***: $\alpha\boldsymbol{a}+\beta\boldsymbol{b}$, где α,β – скалярные величины, если использовать формат: **matadd(a,b,alpha,beta)**.

Скалярное, векторное произведение векторов и угол между векторами

Скалярное произведение двух векторов $(\boldsymbol{a},\boldsymbol{b}) = \sum_{i=1}^{n} a_i b_i$ в n-мерном вещественном пространстве вычисляется с помощью команды **a.b** или команды **DotProduct(a,b)** пакета **LinearAlgebra**.

Соответствующая команда пакета **linalg**: **dotprod(a,b)**.

Векторное произведение двух векторов $\boldsymbol{a}\times\boldsymbol{b}$ в трехмерном евклидовом пространстве вычисляется с помощью команды **a &x b** или команды **CrossProduct(a,b)** пакета **LinearAlgebra**.

Соответствующая команда пакета **linalg**: **crossprod(a,b).**

Угол между двумя векторами $\boldsymbol{a}$ и $\boldsymbol{b}$ вычисляется с помощью команды **VectorAngle(a,b)** пакета **LinearAlgebra**.

Соответствующая команда пакета **linalg**: **angle(a,b)**.

Норма вектора.

Евклидову норму (длину) вектора $\boldsymbol{a} = (x_1,..., x_n)$, которая равна $\|\boldsymbol{a}\| = \sqrt{x_1^2 + ... + x_n^2}$, можно вычислить с помощью команды **Norm(a,2)** пакета **LinearAlgebra**. Общий вид команды:

- **Norm(A, p, c)** – p-норма матрицы или вектора**.** Второй аргумент команды задает тип нормы.
- **VectorNorm(A, p, c)** – p-норма вектора, c – (необязательные) опции для результирующего объекта

Значения параметра p (по умолчанию = infinity)

2, Euclidean или **Frobenius** – Евклидова норма

infinity – максимальный по модулю элемент

Соответствующая команда пакета **linalg**: **norm(a,2).**

Нормировать вектор $\boldsymbol{a}$: $\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ можно с помощью команды **Normalize(a)** пакета **LinearAlgebra**.

Соответствующая команда пакета **linalg**: **normalize(a).**

Нахождение базиса системы векторов. Ортогонализация системы векторов по процедуре Грамма-Шмидта.

Если имеется система n векторов $\{\boldsymbol{a}_1,\boldsymbol{a}_2,...,\boldsymbol{a}_n\}$, то с помощью команды **Basis([a1,a2,…,an])** пакета **LinearAlgebra** можно найти базис этой системы.

Соответствующая команда пакета **linalg**: **basis([a1,a2,…,an]).**

При помощи команды **GramSchmidt([a1,a2,…,an])** пакета **LinearAlgebra** можно ортогонализовать систему линейно-независимых векторов $\{\boldsymbol{a}_1,\boldsymbol{a}_2,...,\boldsymbol{a}_n\}$. Для ортонормализации следует указать опцию **normalized=true** или просто **normalized**.

Задание 1.

1. Даны два вектора: $\boldsymbol{a} = (2,1,3,2)$ и $\boldsymbol{b} = (1,2,-2,1)$. Найти $(\boldsymbol{a},\boldsymbol{b})$ и угол между $\boldsymbol{a}$ и $\boldsymbol{b}$. Для решения этой задачи наберите:

```
> with(LinearAlgebra):
```

```
> a:=Vector([2,1,3,2]); b:=Vector([1,2,-2,1]);
```

$$a := \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 2 \\ -2 \\ 1 \end{bmatrix}$$

```
> a.b; DotProduct(a,b);
> phi=VectorAngle(a,b);
```

$$0$$

$$0$$

$$\phi = \frac{1}{2}\pi$$

2. Найти векторное произведение $\boldsymbol{c} = [\boldsymbol{a}, \boldsymbol{b}]$, а затем скалярное произведение $(\boldsymbol{a}, \boldsymbol{c})$, где $\boldsymbol{a} = (2,-2,1)$, $\boldsymbol{b} = (2,3,6)$.

```
> with(LinearAlgebra):
> a:=<2,-2,1>; b:=<2,3,6>;
```

$$a := \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix}$$

$$b := \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}$$

```
> c:=a &x b;
```

$$c := \begin{bmatrix} -15 \\ -10 \\ 10 \end{bmatrix}$$

```
> c:=CrossProduct(a,b);
```

$$c := \begin{bmatrix} -15 \\ -10 \\ 10 \end{bmatrix}$$

```
> DotProduct(a,c);
```

$$0$$

3. Найти евклидову норму вектора $\boldsymbol{a} = (2,-2,1)$.

```
> with(LinearAlgebra):
> a := Vector[row]([1, 2, 3, 4, 5, 6]);
```

$$a := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

```
> Norm(a,2);
```

$$\sqrt{91}$$

4. Из системы векторов: $\boldsymbol{a}_1=(1,2,2,-1)$, $\boldsymbol{a}_2=(1,1,-5,3)$, $\boldsymbol{a}_3=(3,2,8,7)$, $\boldsymbol{a}_4=(0,1,7,-4)$, $\boldsymbol{a}_5=(2,1,12,-10)$ выделить базис и ортогонализовать его по процедуре Грамма-Шмидта:

```
> with(LinearAlgebra):
> a1:=Vector([1,2,2,-1]): a2:=Vector([1,1,-5,3]):
a3:=Vector([3,2,8,7]): a4:=Vector([0,1,7,-4]):
a5:=Vector([2,1,12,-10]):
> b:=Basis([a1,a2,a3,a4,a5]);
```

$$b := \left[\begin{bmatrix}1\\2\\2\\-1\end{bmatrix}, \begin{bmatrix}1\\1\\-5\\3\end{bmatrix}, \begin{bmatrix}3\\2\\8\\7\end{bmatrix}, \begin{bmatrix}2\\1\\12\\-10\end{bmatrix}\right]$$

```
> GramSchmidt(b);
```

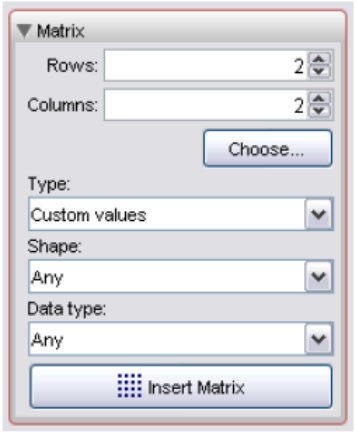
$$\left[\begin{bmatrix}1\\2\\2\\-1\end{bmatrix}, \begin{bmatrix}2\\3\\-3\\2\end{bmatrix}, \begin{bmatrix}\frac{81}{65}\\-\frac{93}{65}\\\frac{327}{65}\\\frac{549}{65}\end{bmatrix}, \begin{bmatrix}\frac{1633}{724}\\-\frac{923}{724}\\-\frac{71}{724}\\-\frac{355}{724}\end{bmatrix}\right]$$

§2. Действия с матрицами

Способы задания матриц.

Матрицу в Maple можно задать несколькими способами.

1) С помощью палитры Matrix, где можно указать количество строк и столбцов матрицы и вставить шаблон (Insert Matrix). Полученный шаблон можно заполнить значениями, используя клавишу Tab для перемещения между элементами матрицы.



2) С помощью угловых скобок
<<a11,…,an1>|<a12,…,an2>|….|<a1m,…,anm>> – матрица размера n x m, заданная по столбцам (составленная из вектор-столбцов)

<<a11|…|a1m>,<a21|….|a2m>,…,<an1|…|anm>> – матрица размера n x m , заданная по строкам (составленная из вектор-строк)

```
> A := <<1, 2, 3>|<4, 5, 6>|<7, 8, 10>|<11, 12, 13>>;
```

$$A := \begin{bmatrix} 1 & 4 & 7 & 11 \\ 2 & 5 & 8 & 12 \\ 3 & 6 & 10 & 13 \end{bmatrix}$$

```
> B := <<1|2|3>, <4|5|6>>;
```

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

3) С помощью команды-конструктора **Matrix([[a11,a12,…,a1m],[a21,a22,…,a2m],…,[an1,an2,…,anm])** или **Matrix(n, m, [[a11,a12,…,a1n], [a21,a22,…,a2m], …, [an1,an2,…,anm]])**, где **n** – число строк, **m** – число столбцов в матрице. Эти числа задавать необязательно, а достаточно перечислить элементы матрицы построчно в квадратных скобках через запятую. Например:

```
> M := Matrix([[1, 2], [3, 4]])
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> C := Matrix(3, 4, [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
```

$$C := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Задание элементов матрицы через функцию

```
> f := (i, j) → i + j; Matrix(3, f)
```

$$f := (i,j) \to i + j$$

$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

Использование опций для определения формы матрицы

```
> Matrix(3, {(1, 1) = 50, (1, 2) = 60}, fill = 1, shape = symmetric)
```

$$\begin{bmatrix} 50 & 60 & 1 \\ 60 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Определение верхнетреугольной матрицы

```
> Matrix(3, fill = 1, shape = triangular)
```

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

Определение диагональной матрицы

```
> Matrix(3, Vector([1, 2, 3]), shape = diagonal)
```

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

Другие примеры

> $Matrix(4, 3, A, fill = 87)$

$$\begin{bmatrix} 1 & 2 & 87 \\ 3 & 4 & 87 \\ 5 & 6 & 87 \\ 87 & 87 & 87 \end{bmatrix}$$

> $Matrix(2, 3, symbol = m)$

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \end{bmatrix}$$

Заполнение по столбцам (по умолчанию элементы заполяются по строкам)

> $Matrix(3, [[1, 2, 3], [4, 5, 6], [7, 8, 9]], scan = columns)$

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Для последующей работы с командами пакета **linalg** матрицу можно задать либо с помощью команды **array**, либо с помощью команды **matrix** из пакета **linalg**:

> $AA := array(1..3, 1..2, [[4, 5], [6, 7], [8, 9]]);$

$$AA := \begin{bmatrix} 4 & 5 \\ 6 & 7 \\ 8 & 9 \end{bmatrix}$$

> $with(linalg):$

> $A := matrix(2, 3, [1, 2, 3, 4, 5, 6]);$

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

> $B := matrix([[1, 2], [3, 4]]);$

$$B := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Доступ к элементам матрицы **A** можно получить в строке вывода, если ввести команду **A[i,j]** , где **i** – номер строки, **j** – номер столбца. Например:

> $M := \begin{bmatrix} 4 & 5 & 6 \\ -7 & -8 & -9 \\ 101 & 102 & 103 \end{bmatrix}:$

> $M[2, 1]$

$$-7$$

Выделение подматрицы:

> $M[1..2, 2..3]$

$$\begin{bmatrix} 5 & 6 \\ -8 & -9 \end{bmatrix}$$

В *Maple* матрицы специального вида можно генерировать с помощью дополнительных команд. В пакете **LinearAlgebra** есть следующие команды для генерирования матриц специального вида:

BandMatrix
BezoutMatrix
ConstantMatrix
DiagonalMatrix
GivensRotationMatrix
HankelMatrix

HilbertMatrix
HouseholderMatrix
IdentityMatrix
JordanBlockMatrix
RandomMatrix
ScalarMatrix
SylvesterMatrix
ToeplitzMatrix
VandermondeMatrix
ZeroMatrix

Некоторые примеры:

```
> with(LinearAlgebra);
```

Нулевая матрица

```
> ZeroMatrix(3);
```

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

Нулевой вектор

```
> ZeroVector(2);
```

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

Единичный вектор

UnitVector(i, d)задает вектор длины **d**с единицей на позиции **i**

```
> UnitVector(2, 3);
```

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

Единичная матрица

```
> IdentityMatrix(2);
```

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

Диагональная матрица

```
> DiagonalMatrix([a, b, c]);
```

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$

Случайная квадратная матрица с целыми числами

```
> RandomMatrix(3);
```

$$\begin{bmatrix} 27 & 99 & 92 \\ 8 & 29 & -31 \\ 69 & 44 & 67 \end{bmatrix}$$

```
> ConstantMatrix(4,5,2);
```

$$\begin{bmatrix} 4 & 4 \\ 4 & 4 \\ 4 & 4 \\ 4 & 4 \\ 4 & 4 \end{bmatrix}$$

```
> ConstantMatrix(n,4,outputoptions=[shape=triangular[upper]]);
```

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

```
> ScalarMatrix(x, 3, 4);
```

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

В пакете **linalg** также есть опции и команды для задания матриц специального вида. В частности диагональную матрицу можно получить командой **diag** пакета **linalg**. Например:

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

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

```
> array(1 .. 3, 1 .. 3, identity);
```

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

Определение размера матрицы

В пакете **LinearAlgebra** имеются следующие команды:

- **Dimension(A)** – размерность матрицы или вектора
- **RowDimension(A)** – число строк матрицы
- **ColumnDimension(A)** – число столбцов матрицы

В пакете **linalg:** число строк в матрице *A* можно определить с помощью команды **rowdim(A)**, а число столбцов – с помощью команды **coldim(A)**.

Операции со строками и столбцами матрицы

В пакете **LinearAlgebra** имеются следующие команды:

- **DeleteRow(A, L, outopts)** – удаление строк матрицы **A**
- **DeleteColumn(A, L, outopts)** – удаление столбцов матрицы **A**

  L – номера удаляемых строк (столбцов), могут быть в виде интервала или списка

- **Row(A, L, outopts)** – извлечение строк матрицы **A**
- **Column(A, L, outopts)** – извлечение столбцов матрицы **A**

  L – номера извлекаемых строк (столбцов), могут быть в виде интервала или списка

- **RowOperation(A, [ri,rj])** – перестановка строк **ri** и **rj** матрицы **A**
- **ColumnOperation(A, [ci,cj])** – перестановка столбцов **ci** и **cj** матрицы **A**

- **RowOperation(A, [ri,rj],expr)** – изменение строки **ri**: **ri:=ri+rj*expr**, где **expr** – число или выражение, сложение строк
- **ColumnOperation(A, [ci,cj],expr)** – изменение столбца **ci**: **ci:=ci+cj*expr**, где **expr** – число или выражение, сложение столбцов
- **RowOperation(A, r,expr)** – умножение строки **r** на выражение **expr**: **r:=r*expr**

- **ColumnOperation(A, c,expr)** – умножение столбца **c** на выражение **expr**: **c:=c*expr**

Некоторые примеры.

```
> with(LinearAlgebra):
> A := Matrix([[1,2,3],[4,5,6],[7,8,9]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Удаление строк или столбцов

```
> DeleteRow(A, 2);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> DeleteColumn(A, 2 .. 3);
```

$$\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$$

```
> DeleteRow(A, [1, 3]);
```

$$\begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

Извлечение строк или столбцов

```
> A := RandomMatrix(4);
```

$$A := \begin{bmatrix} -81 & 33 & 27 & -2 \\ -38 & -98 & -93 & -32 \\ -18 & -77 & -76 & -74 \\ 87 & 57 & -72 & -4 \end{bmatrix}$$

```
> a := Row(A, 2);
```

$$a := \begin{bmatrix} -38 & -98 & -93 & -32 \end{bmatrix}$$

```
> B := Row(A, 1 .. 3);
```

$$B := \begin{bmatrix} -81 & 33 & 27 & -2 \end{bmatrix}, \begin{bmatrix} -38 & -98 & -93 & -32 \end{bmatrix}, \begin{bmatrix} -18 & -77 & -76 & -74 \end{bmatrix}$$

Перестановка строк или столбцов

```
> RowOperation(A, [1, 3]);
```

$$\begin{bmatrix} -18 & -77 & -76 & -74 \\ -38 & -98 & -93 & -32 \\ -81 & 33 & 27 & -2 \\ 87 & 57 & -72 & -4 \end{bmatrix}$$

```
> ColumnOperation(A, [2, 3]);
```

$$\begin{bmatrix} -81 & 27 & 33 & -2 \\ -38 & -93 & -98 & -32 \\ -18 & -76 & -77 & -74 \\ 87 & -72 & 57 & -4 \end{bmatrix}$$

```
> A := Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Сложение строк: r1:=r1+r3*100

```
> RowOperation(A, [1, 3], 100);
```

$$\begin{bmatrix} 701 & 802 & 903 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Сложение столбцов: c3:=c3+c2*10

```
> ColumnOperation(A, [3, 2], 10);
```

$$\begin{bmatrix} 1 & 2 & 23 \\ 4 & 5 & 56 \\ 7 & 8 & 89 \end{bmatrix}$$

Умножение столбца: r2:=r2*10

```
> ColumnOperation(A, 2, 10);
```

$$\begin{bmatrix} 1 & 20 & 3 \\ 4 & 50 & 6 \\ 7 & 80 & 9 \end{bmatrix}$$

Сложение матриц

Сложение двух матриц одинаковой размерности осуществляется теми же командами, что и сложение векторов. В пакете **LinearAlgebra** сложение матриц выполняется с помощью обычного знака сложения **+**.

В пакете **linalg** – с помощью команд **evalm(A+B)** или **matadd(A,B)**.

Умножение матриц

Умножение матрицы на число осуществляется с помощью обычного знака умножения ***, например

$$> a := \begin{bmatrix} 93 & 43 \\ 19 & 37 \end{bmatrix}:$$

$$> 12a$$

$$\begin{bmatrix} 1116 & 516 \\ 228 & 444 \end{bmatrix}$$

Некоммутативное умножение векторов и матриц осуществляется с помощью знака **.** (точка), например

$$> a := \begin{bmatrix} 93 & 43 \\ 19 & 37 \end{bmatrix}: b := \begin{bmatrix} 48 & 20 \\ 19 & 37 \end{bmatrix}: c := \langle 23, 6 \rangle:$$

$$> a.c$$

$$\begin{bmatrix} 2397 \\ 659 \end{bmatrix}$$

- **A.B** – матричное (некоммутативное) умножение матриц и векторов

  Возведение матрицы в степень осуществляется с помощью знака возведения в степень **^**, например:

> a^3

$$\begin{bmatrix} 986548 & 613868 \\ 271244 & 187092 \end{bmatrix}$$

Также в пакете **LinearAlgebra** имеются следующие команды:

- **MatrixVectorMultiply(A, u)** – умножение матрицы **A** на вектор **u**
- **MatrixMatrixMultiply(A, B)** – умножение матрицы **A** на матрицу **B**
- Общая команда: **Multiply(A,B)** – в качестве второго аргумента можно указать матрицу или вектор.

В пакете **linalg** произведение двух матриц может быть найдено с помощью двух команд:

1) **evalm(A&*B);**
2) **multiply(A,B)**

Возведение в степень: **evalm(A^n)**– возведение матрицы **A** в степень **n**

```
> with(LinearAlgebra): A := Matrix([[1, 2], [3, 4]]); B :=
Matrix(2, [10, 20, 30, 40]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$B := \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

```
> A+B;
```

$$\begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix}$$

```
> A.B;
```

$$\begin{bmatrix} 70 & 100 \\ 150 & 220 \end{bmatrix}$$

```
> v:= <100,150>: A.v;
```

$$\begin{bmatrix} 400 \\ 900 \end{bmatrix}$$

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

$$\begin{bmatrix} 70 & 100 \\ 150 & 220 \end{bmatrix}$$

```
> MatrixVectorMultiply(A, v);
```

$$\begin{bmatrix} 400 \\ 900 \end{bmatrix}$$

```
> with(linalg):
> A:=matrix([[1,0],[0,-1]]);
> B:=matrix([[-5,1], [7,4]]);
```

$$A := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad B := \begin{bmatrix} -5 & 1 \\ 7 & 4 \end{bmatrix}$$

```
> v:=vector([2,4]);
```

$$v := [2,4]$$

```
> multiply(A,v);
```

$$[2,-4]$$

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

$$\begin{bmatrix} -5 & 1 \\ -7 & -4 \end{bmatrix}$$

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

$$\begin{bmatrix} -4 & 1 \\ 7 & 3 \end{bmatrix}$$

Определители, миноры и алгебраические дополнения. Ранг и след матрицы.

В пакете **LinearAlgebra** имеются следующие команды:

- **Determinant(A)** – вычисление определителя матрицы **A**
- **Minor(A, r, c, out, meth, outopts)** – вычисление минора M(i,j) к элементу A[i,j] матрицы *A*, **out** задает тип результате в виде **output=matrix** или/и **output=determinant** (определитель), **meth** – метод вычисления определителя
- **Rank(A)** – ранг матрицы
- **Trace(A)** – след матрицы (сумма диагональных элементов)

В пакете **linalg**:

- **minor(A,i,j)** – возвращает матрицу, полученную вычеркиванием строки **i** и столбца **j** матрицы **A**
- **det(A)** – вычисление определителя матрицы *A*

Пример использования команд пакета **LinearAlgebra**:

```
> with(LinearAlgebra):A:=Matrix([[4,0,5],[0,1,-6],[3,0,4]]);
```

$$A := \begin{bmatrix} 4 & 0 & 5 \\ 0 & 1 & -6 \\ 3 & 0 & 4 \end{bmatrix}$$

```
> Determinant(A);
```

$$1$$

```
> Minor(A,3,2);
```

$$-24$$

```
> Minor(A,3,2,output=matrix);
```

$$\begin{bmatrix} 4 & 5 \\ 0 & -6 \end{bmatrix}$$

```
> Trace(A);
```

$$9$$

```
> Rank(A);
```

$$3$$

Обратная и транспонированная (эрмитово-сопряженная) матрица.

Обратную матрицу A^{-1} , такую что $A^{-1}A=AA^{-1}=E$, где E – единичная матрица, можно вычислить следующим образом:

1) с помощью возведения в степень −1: **A^(-1)**

```
> A := Matrix([[4,0,5],[0,1,-6],[3,0,4]]):
> A^(-1)
```

$$\begin{bmatrix} 4 & 0 & -5 \\ -18 & 1 & 24 \\ -3 & 0 & 4 \end{bmatrix}$$

2) с помощью команды **MatrixInverse(A)** пакета **LinearAlgebra**

```
> with(LinearAlgebra):MatrixInverse(A);
```

$$\begin{bmatrix} 4 & 0 & -5 \\ -18 & 1 & 24 \\ -3 & 0 & 4 \end{bmatrix}$$

В пакете **linalg**:
1) **evalm(1/A);**
2) **inverse(A).**

Транспонирование матрицы A – это изменение местами строк и столбцов. Полученная в результате этого матрица называется транспонированной и обозначается A^{T}. *Эрмитово-сопряженная матрица* A^{H} – это матрица, к которой применена операция транспонирования и комплексного сопряжения элементов (в случае вещественной матрицы $A^{\mathrm{H}}= A^{\mathrm{T}}$).

Для вычисления *транспонированной* или *эрмитово-сопряженной* матрицы или вектора используется
1) возведение в степень **%T** или, соответственно, **%H**
2) команда **Transpose(A)** или, соответственно, **HermitianTranspose(A)**

```
> d:=<1,2,3>:
> d^%T
```

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

```
> with(LinearAlgebra); A := Matrix([[1, 2], [3, 4]]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> A^%T;
```

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

```
> Transpose(A);
```

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

```
> W := Matrix([[1+2*I, I], [3-4*I, -I]]);
```

$$W := \begin{bmatrix} 1+2\,\mathrm{I} & \mathrm{I} \\ 3-4\,\mathrm{I} & -\mathrm{I} \end{bmatrix}$$

```
> W^%H;
```

$$\begin{bmatrix} 1-2\,\mathrm{I} & 3+4\,\mathrm{I} \\ -\mathrm{I} & \mathrm{I} \end{bmatrix}$$

```
> HermitianTranspose(W);
```

$$\begin{bmatrix} 1-2\,\mathrm{I} & 3+4\,\mathrm{I} \\ -\mathrm{I} & \mathrm{I} \end{bmatrix}$$

Норма матрицы

Норму матрицы A можно вычислить с помощью команды **Norm(A,p)** пакета **LinearAlgebra**. Второй аргумент команды задает тип нормы. Также имеется команда

- **MatrixNorm(A, p, c)** – p-норма матрицы

Значения параметра p (по умолчанию = infinity)

1 – максимальная столбцовая норма $\|A\|_1 = \max_{j=1,m} \sum_{i=1}^{n} |a_{ij}| \quad \forall A \in C^{n\times m}$

infinity – максимальная строчная норма $\|A\|_\infty = \max_{i=1,n} \sum_{j=1}^{m} |a_{ij}| \quad \forall A \in C^{n\times m}$

2 или **Euclidean** – спектральная норма $\|A\|_2 = \sqrt{\max_i |\lambda_i(A^H A)|} \quad \forall A \in C^{n\times m}$

Frobenius – норма Фробениуса $\|A\|_F = \sqrt{\sum_{j=1}^{m}\sum_{i=1}^{n} |a_{ij}|^2} \quad \forall A \in C^{n\times m}$

Соответствующая команда пакета **linalg**: **norm(A,p)**.

Выяснение типа матрицы.

Положительная определенность

В пакете **LinearAlgebra:**

- **IsDefinite(A, q)** – проверяет положительную/отрицательную определенность матрицы **A**, параметр **q** имеет вид **query = attribute**, где attribute может иметь одно из значений:

  'positive_definite' – положительно определенная: $x^H Ax > 0 \quad \forall x \in C^n$, x^H - сопряженный транспонированный вектор, A – эрмитова матрица: $A^H = A$;

  'positive_semidefinite' – положительно полуопределенная: $x^H Ax \geq 0 \quad \forall x \in C^n$

  'negative_definite' – отрицательно определенная: $x^H Ax < 0 \quad \forall x \in C^n$

  'negative_semidefinite' – отрицательно полуопределенная: $x^H Ax \leq 0 \quad \forall x \in C^n$

  'indefinite' – неопределенная
- **IsDefinite(A)** – проверяет положительную определенность матрицы **A**, т. е. по умолчанию **query= 'positive_definite'**

linalg:

definite(A,kind) – проверяет положительную/отрицательную определенность матрицы **A**, параметр **kind** может принимать одно из следующих значений: **'positive_def', 'positive_semidef', 'negative_def'** или **'negative_semidef'**

```
> with(LinearAlgebra): A := Matrix(2, 2, [2, 1, 1, 3]);
```

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

```
> IsDefinite(A);
```

true

- Проверить *ортогональность* матрицы A ($AA^T = A^T A = I$) можно командой **IsOrthogonal(A)**, где I – единичная матрица.
- **IsUnitary(A)** – проверяет, является ли матрица A унитарной: т. е. такой, что $AA^H = A^H A = I$, где H – эрмитово сопряжение.

```
> with(LinearAlgebra):
> A := Matrix([[1, 2], [3, 4]])
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> IsOrthogonal(A)
```

false

> $Q := \left\langle \left\langle \frac{\sqrt{10}\cdot 3}{10}, -\frac{\sqrt{10}}{10} \right| \left| \frac{\sqrt{10}\, I}{10}, \frac{3\sqrt{10}\, I}{10} \right\rangle \right\rangle$

$$Q := \begin{bmatrix} \frac{3}{10}\sqrt{10} & \frac{1}{10}\,\mathrm{I}\sqrt{10} \\ -\frac{1}{10}\sqrt{10} & \frac{3}{10}\,\mathrm{I}\sqrt{10} \end{bmatrix}$$

> *IsOrthogonal*(*Q*)

false

> *IsUnitary*(*Q*)

true

Функции от матриц.

Вычисление матричной экспоненты e^{At} (exp(A*t) = I + A*t + 1/2!*A^2*t^2 + ...) возможно с помощью команды **MatrixExponential(A)**. Например:

```
> with(LinearAlgebra): T:=Matrix([[5*a,2*b],[-2*b,5*a]]);
> MatrixExponential(T);
```

$$T := \begin{bmatrix} 5a & 2b \\ -2b & 5a \end{bmatrix}$$

$$\begin{bmatrix} e^{5a}\cos(2b) & e^{5a}\sin(2b) \\ -e^{5a}\sin(2b) & e^{5a}\cos(2b) \end{bmatrix}$$

Задание 2.

1. Даны матрицы: $A = \begin{bmatrix} 4 & 3 \\ 7 & 5 \end{bmatrix}$, $B = \begin{bmatrix} -28 & 93 \\ 38 & -126 \end{bmatrix}$, $C = \begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix}$. Найти: $(AB)C$, det$A$, det$B$, det$C$, det[$(AB)C$]. Наберите:

```
> with(LinearAlgebra):
>  A:=Matrix([[4,3],[7,5]]):
>  B:=Matrix([[-28,93],[38,-126]]):
>  C:=Matrix([[7,3],[2,1]]): F:=A.B.C;
>  Det(A)=Determinant(A); Det(B)=Determinant(B);
Det(C)=Determinant(C); Det(F)=Determinant(F);
>
```

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

$$Det\left(\begin{bmatrix} 4 & 3 \\ 7 & 5 \end{bmatrix}\right) = -1$$

$$Det\left(\begin{bmatrix} -28 & 93 \\ 38 & -126 \end{bmatrix}\right) = -6$$

$$Det\left(\begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix}\right) = 1$$

$$Det\left(\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}\right) = 6$$

2. Дана матрица $A = \begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix}$, найти: det$A$, A^{-1}, A^T, det(M_{22}). Наберите:

```
> with(LinearAlgebra): A:=Matrix([[2,5,7],[6,3,4],[5,-2,-3]]);
Det(A)=Determinant(A);
> Transpose(A); MatrixInverse(A); Minor(A,2,2);
```

$$A := \begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix}$$

$$Det\left(\begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix}\right) = -1$$

$$\begin{bmatrix} 2 & 6 & 5 \\ 5 & 3 & -2 \\ 7 & 4 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 1 \\ -38 & 41 & -34 \\ 27 & -29 & 24 \end{bmatrix}$$

$$-41$$

3. Найти ранг матрицы $A = \begin{bmatrix} 8 & -4 & 5 & 5 & 9 \\ 1 & -3 & -5 & 0 & -7 \\ 7 & -5 & 1 & 4 & 1 \\ 3 & -1 & 3 & 2 & 5 \end{bmatrix}$.

```
> A:=Matrix([[8,-4,5,5,9], [1,-3,-5,0,-7], [7,-5,1,4,1], [3,-
1,3,2,5]]): rank_A=Rank(A);
```

$$rank\_A = 3$$

4. Вычислить e^T, где $T = \begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix}$.

```
> MatrixExponential(Matrix([[3,-1],[1,1]]));
```

$$\begin{bmatrix} 2\,e^2 & -e^2 \\ e^2 & 0 \end{bmatrix}$$

5. Дана матрица $A = \begin{bmatrix} 5 & 1 & 4 \\ 3 & 3 & 2 \\ 6 & 2 & 10 \end{bmatrix}$. Найти значение многочлена $P(A) = A^3 - 18A^2 + 64A$.

```
> A:=Matrix([[5,1,4],[3,3,2],[6,2,10]]):
> P_A=A^3-18*A^2+64*A;
```

$$P\_A = \begin{bmatrix} 64 & 0 & 0 \\ 0 & 64 & 0 \\ 0 & 0 & 64 \end{bmatrix}$$

§3. Спектральный анализ матрицы

Собственные числа и собственные векторы матрицы.

Из курса линейной алгебры известно, что если $A\boldsymbol{x}=\lambda\boldsymbol{x}$, то вектор $\boldsymbol{x}$ называется собственным вектором матрицы A, а число λ – собственным числом, соответствующим данному собственному вектору. Совокупность всех собственных чисел матрицы называется спектром матрицы. Если в спектре матрицы одно и тоже собственное число встречается k раз, то говорят, что кратность этого собственного числа равна k.

- **`Eigenvalues(A)`** – возвращает собственные значения матрицы **A** в виде вектор-столбца.
- **`Eigenvectors(A)`** – возвращает собственные значения матрицы **A** в виде вектор-столбца и матрицу из собственных векторов по столбцам.

Полный синтаксис:

`Eigenvectors(A, C, imp, o, outopts)` – остальные аргументы необязательны

`C` – матрица для решения обобщенной задачи на собственные значения, т. е. находятся корни det(lambda C - A)

`imp` – задает, в каком виде будут вычисляться корни характеристического многочлена; если имеет значение **`implicit=true`** или просто **`implicit`**, то будут вычисление в неявном виде RootOf

`o` – задает формат вывода результата в виде: **`output =obj`**, где **`obj`** может принимать значения **'values'** (собственные значения в виде вектор-столбца), **'vectors'** (собственные значения матрицы **A** в виде вектор-столбца и матрица из собственных векторов по столбцам), **'list' (**список из собственных значений, их кратностей и соответствующих им собственных векторов)

outopts – задает опции outputoptions конструктора для результирующего объекта, например outputoptions=[datatype=float,shape=….]

В пакете **`linalg`** для нахождения собственных чисел матрицы A используется команда **`eigenvalues(A)`.** Для нахождения собственных векторов матрицы A используется команда **`eigenvectors(A)`.** В результате выполнения этой команды будут получены собственные числа, их кратность и соответствующие им собственные векторы.

Чтобы понять, в каком виде получаются результаты выполнения команды **`eigenvectors`**, внимательно разберитесь со следующим примером: матрица $A=\begin{bmatrix}3&-1&1\\-1&5&-1\\1&-1&3\end{bmatrix}$ имеет 3 собственных вектора: $\boldsymbol{a}_1=(-1,0,1)$, отвечающий собственному числу $\lambda_1=2$ кратности 1, $\boldsymbol{a}_2=(1,1,1)$, отвечающий собственному числу $\lambda_2=3$ кратности 1, $\boldsymbol{a}_3=(1,-2,1)$, отвечающий собственному числу $\lambda_3=6$ кратности 1. Найдем их в *Maple*:

```
> A:=Matrix([[3,-1,1],[-1,5,-1],[1,-1,3]]):
> Eigenvectors(A,output=list);
```

$$\left[\left[2,\ 1,\ \left\{\begin{bmatrix}-1\\0\\1\end{bmatrix}\right\}\right],\ \left[3,\ 1,\ \left\{\begin{bmatrix}1\\1\\1\end{bmatrix}\right\}\right],\ \left[6,\ 1,\ \left\{\begin{bmatrix}1\\-2\\1\end{bmatrix}\right\}\right]\right]$$

В строке вывода перечислены в квадратных скобках собственное число, его кратность и соответствующий собственный вектор в фигурных скобках , затем следующие наборы таких же данных.

Характеристический и минимальный многочлены матрицы.

LinearAlgebra:

- **`CharacteristicPolynomial(A, lambda)`** – характеристический многочлен $P_A(\lambda) = \det(\lambda E - A)$
- **`MinimalPolynomial(A, lambda)`** – минимальный многочлен (делитель)
- **`CharacteristicMatrix(A, lambda, outopts)`** – характеристическая матрица **`lambda*I-A`**

linalg:

- **charpoly(A,lambda)** и **minpoly(A,lambda)**
- **charmat(A,lambda)**

Канонические и специальные виды матрицы.

Привести матрицу *A* к нормальной форме Жордана можно командой **`JordanForm(A)`.**

К треугольному виду матрицу *A* можно привести тремя способами:

- **`GaussianElimination(A)`** – приведение матрицы к треугольному виду методом исключения Гаусса. Полный синтаксис: **`GaussianElimination(A, m, outopts)`** – остальные аргументы необязательны

  m – параметр используемого метода в виде **`method='GaussianElimination'`** (обычный метод Гаусса, по умолчанию) или **`method='FractionFree'`** (метод Гаусса без деления, для работы с символьными матрицами, так как не производит нормировку элементов и исключает возможные ошибки, связанные с делением на нуль)

  outopts – задает опции **outputoptions** для результирующего объекта

- **`ReducedRowEchelonForm(A)`** – приведение к треугольному виду методом Гаусса-Жордана

Соответствующие команды пакета **`linalg`** имеют вид: **`jordan(A)`**, **`gausselim(A`**) – приведение к треугольному виду методом Гаусса, **`ffgausselim(A`**) – приведение к треугольному виду методом Гаусса без деления, **`gaussjord(A`**) – приведение к треугольному виду методом Гаусса-Жордана.

LU- и QR-разложение матрицы

- **`LUDecomposition(A)`** – LU-разложение матрицы: A=PLU. Полный синтаксис:

  `LUDecomposition(A, m, out, c, ip, outopts, ...)` – остальные аргументы необязательны (подробности – см. Help). Значения некоторых параметров:

  m – параметр используемого метода в виде **`method = 'GaussianElimination'`** (обычный метод Гаусса, по умолчанию), **`method = 'FractionFree'`** (метод Гаусса без деления), **`method='RREF'`**, **`method='Cholesky'`** или **`method='none'`**

  `out` – параметр выдаваемой матрицы в виде **`output = obj`**, где **`obj`** может принимать значения **`'P', 'L', 'U', 'U1', 'R', 'NAG', 'determinant', 'rank'`** или список этих значений.

  `outopts` – задает опции **`outputoptions`** для результирующего объекта

- **`QRDecomposition(A)`** – QR-разложение матрицы: A=QR. Полный синтаксис:

  `QRDecomposition(A, fs, out, c, outopts, ...)` – остальные аргументы необязательны (подробности – см. Help). Значения некоторых параметров:

fs – логический параметр в виде **fullspan='false'** (по умолчанию, неполное QR-разложение) или **fullspan='true'** (или просто **fullspan,** полное QR-разложение)

out – параметр выдаваемой матрицы в виде **output = obj**, где **obj** может принимать значения **'Q', 'R', 'NAG', 'rank'** или список этих значений.

linalg:

- **Ludecomp(A)** и **QRdecomp(A)**

Задание 3.

1. Дана матрица $U=\begin{bmatrix} 3 & 2-i \\ 2+i & 7 \end{bmatrix}$. Найти ее собственные векторы и собственные числа.

```
> with(LinearAlgebra):U:=Matrix([[3,2-I],[2+I,7]]):
> Eigenvectors(U);
> Eigenvectors(U,output=list);
```

$$\begin{bmatrix} 8 \\ 2 \end{bmatrix}, \begin{bmatrix} \frac{2}{5}-\frac{1}{5}\,\mathrm{I} & -2+\mathrm{I} \\ 1 & 1 \end{bmatrix}$$

$$\left[\left[2,1,\left\{\begin{bmatrix} -2+\mathrm{I} \\ 1 \end{bmatrix}\right\}\right],\left[8,1,\left\{\begin{bmatrix} \frac{2}{5}-\frac{1}{5}\,\mathrm{I} \\ 1 \end{bmatrix}\right\}\right]\right]$$

2. Дана матрица $A=\begin{bmatrix} 3 & -i & 0 \\ i & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$. Найти собственные векторы, собственные числа, характеристический многочлен и минимальный многочлен, Жорданову форму.

```
> A:=Matrix([[3,-I,0],[I,3,0],[0,0,4]]):
> Eigenvectors(A,output=list);
```

$$\left[\left[2,1,\left\{\begin{bmatrix} \mathrm{I} \\ 1 \\ 0 \end{bmatrix}\right\}\right],\left[4,2,\left\{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\begin{bmatrix} -\mathrm{I} \\ 1 \\ 0 \end{bmatrix}\right\}\right]\right]$$

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

$$\lambda^3-10\,\lambda^2+32\,\lambda-32$$

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

$$8-6\,\lambda+\lambda^2$$

```
> JordanForm(A);
```

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

3. Дана матрица $A=\begin{bmatrix} 1 & -3 & 4 \\ 4 & -7 & 8 \\ 6 & -7 & 7 \end{bmatrix}$. Привести матрицу A к Жордановой форме, треугольному виду (тремя способами), найти ее характеристическую матрицу. Получить LU- и QR-разложение матрицы A.

```
> A := Matrix([[1, -3, 4], [4, -7, 8], [6, -7, 7]]):
> JordanForm(A);
```

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

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

$$\begin{bmatrix} \lambda - 1 & 3 & -4 \\ -4 & \lambda + 7 & -8 \\ -6 & 7 & \lambda - 7 \end{bmatrix}$$

```
> GaussianElimination(A);
```

$$\begin{bmatrix} 1 & -3 & 4 \\ 0 & 5 & -8 \\ 0 & 0 & \frac{3}{5} \end{bmatrix}$$

```
> GaussianElimination(A,method=FractionFree);
```

$$\begin{bmatrix} 1 & -3 & 4 \\ 0 & 5 & -8 \\ 0 & 0 & 3 \end{bmatrix}$$

```
> ReducedRowEchelonForm(A);
```

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

```
> LUDecomposition(A);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 6 & \frac{11}{5} & 1 \end{bmatrix}, \begin{bmatrix} 1 & -3 & 4 \\ 0 & 5 & -8 \\ 0 & 0 & \frac{3}{5} \end{bmatrix}$$

```
> LUDecomposition(A,output=['L','U']);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 6 & \frac{11}{5} & 1 \end{bmatrix}, \begin{bmatrix} 1 & -3 & 4 \\ 0 & 5 & -8 \\ 0 & 0 & \frac{3}{5} \end{bmatrix}$$

```
> QRDecomposition(A);
```

$$\begin{bmatrix} \frac{1}{53}\sqrt{53} & -\frac{43}{3021}\sqrt{2014} & \frac{7}{57}\sqrt{38} \\ \frac{4}{53}\sqrt{53} & -\frac{79}{6042}\sqrt{2014} & -\frac{11}{114}\sqrt{38} \\ \frac{6}{53}\sqrt{53} & \frac{67}{6042}\sqrt{2014} & \frac{5}{114}\sqrt{38} \end{bmatrix},$$

$$\begin{bmatrix} \sqrt{53} & -\frac{73}{53}\sqrt{53} & \frac{78}{53}\sqrt{53} \\ 0 & \frac{3}{53}\sqrt{2014} & -\frac{169}{2014}\sqrt{2014} \\ 0 & 0 & \frac{1}{38}\sqrt{38} \end{bmatrix}$$

§4. Системы линейных уравнений. Матричные уравнения

Системы линейных уравнений и матричные уравнения.

Система линейных уравнений $A\boldsymbol{x}=\boldsymbol{b}$ может быть решена двумя способами.

Способ 1: стандартная команда **`solve`** находит решение системы линейных уравнений, записанных в развернутом виде:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_m + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}.$$

Способ 2: команда **`LinearSolve(A,b)`** из пакета **`LinearAlgebra`** (или команда **`linsolve(A,b)`** из пакета **`linalg`**) находит решение уравнения $A\boldsymbol{x}=\boldsymbol{b}$. Аргументы этой команды: A – матрица, $\boldsymbol{b}$ – вектор.

С помощью команд **`LinearSolve(A,b)`** или **`linsolve(A,b)`** можно найти решение матричного уравнения $AX=B$, если в качестве аргументов этой команды указать, соответственно, матрицы A и B.

- **`NullSpace(A, outopts)`** – поиск базиса ядра матрицы, т.е. векторов {x: Ax=0}, эквивалентно решению однородной системы уравнений
- **`GenerateEquations(A, v, B)`** – генерирование системы линейных уравнений **`Av=B`**, где **`A`** – матрица коэффициентов размера m x n, **`v`** – список неизвестных длины **`n`**, **`B`** – вектор правой части. Пример вызова: **`GenerateEquations(A,[x,y,z],b)`**
- **`GenerateMatrix(eqns, vars)`** – генерирование матрицы коэффициентов из списка (множества) уравнений **`eqns`** и списка (множества) неизвестных **`vars`**. Пример вызова: **`GenerateMatrix([eq1,e2,eq3],[x,y,z])`**

Задание 4.

1. Решить матричное уравнение: $AX=B$; где $A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$, $B=\begin{bmatrix}3 & 5\\ 5 & 9\end{bmatrix}$

```
> with(LinearAlgebra):A:=Matrix([[1,2],[3,4]]):
B:=Matrix([[3,5],[5,9]]):
> X:=LinearSolve(A,B);
```

$$X := \begin{bmatrix} -1 & -1 \\ 2 & 3 \end{bmatrix}$$

2. Дана матрица $A=\begin{bmatrix}1 & 1 & 0\\ 0 & 2 & -1\\ 1 & 3 & -1\end{bmatrix}$. Найти ее ранг, дефект: $d(A)=n-r(A)$, где n – размерность квадратной матрицы, r – ее ранг. Найти ядро A. Наберите:

```
> A:=Matrix([[1,1,0],[0,2,-1],[1,3,-1]]): r_A:=Rank(A);
> d_A:=RowDimension(A)-r_A;
> kern:=NullSpace(A); A.kern[1];
```

$$r\_A := 2$$

$$d\_A := 1$$

$$kern := \left\{ \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ 1 \end{bmatrix} \right\}$$

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

3. Сгенерировать систему однородных уравнений для матрицы из предыдущего примера и найти ее решение.

```
> s:=GenerateEquations(A,[x,y,z]);
```

$$s := [x + y = 0, 2y - z = 0, x + 3y - z = 0]$$

```
> solve(s,[x,y,z]);
```

$$[[x = -y, y = y, z = 2y]]$$

§5. Векторная алгебра в пакете VectorCalculus

Команды пакета VectorCalculus

[&x, `*`, `+`, `-`, `.`, <,>, <|>, About, AddCoordinates, ArcLength, BasisFormat, Binormal, Compatibility, ConvertVector, CrossProd, CrossProduct, Curl, Curvature, D, Del, DirectionalDiff, Divergence, DotProd, DotProduct, Flux, GetCoordinateParameters, GetCoordinates, GetNames, GetPVDescription, GetRootPoint, GetSpace, Gradient, Hessian, IsPositionVector, IsRootedVector, IsVectorField, Jacobian, Laplacian, LineInt, MapToBasis, Nabla, Norm, Normalize, PathInt, PlotPositionVector, PlotVector, PositionVector, PrincipalNormal, RadiusOfCurvature, RootedVector, ScalarPotential, SetCoordinateParameters, SetCoordinates, SpaceCurve, SurfaceInt, TNBFrame, Tangent, TangentLine, TangentPlane, TangentVector, Torsion, Vector, VectorField, VectorPotential, VectorSpace, Wronskian, diff, eval, evalVF, int, limit, series]

Ранее рассматривались команды векторной алгебры пакета **LinearAlgebra**. Здесь приведем аналогичные команды пакета **VectorCalculus**.

Способы задания векторов в пакете VectorCalculus

При подключении пакета **VectorCalculus** вектор в Maple будет представлен в виде разложения по базисным векторам текущей системы координат. При этом вектор можно задать:

3) С помощью угловых скобок (в этом случае координаты вектора можно задать в виде последовательности выражений, отделенных друг от друга запятыми или вертикальной чертой:

```
> with(VectorCalculus):
> v := <1, 2, 3>
```

$$v := e_x + 2e_y + 3e_z$$

```
> vv := <1|2|3>
```

$$vv := e_x + 2e_y + 3e_z$$

4) С помощью команды-конструктора **Vector([x1,x2,…,xn])**, где в квадратных скобках через запятую указываются координаты вектора. Например:

```
> Vector(4, [1, 2, 3, 4])
```

$$e_{x1} + 2e_{x2} + 3e_{x3} + 4e_{x4}$$

С помощью команды **SetCoordinates(v,coordsystemname)** , где **v** – вектор, **coordsystemname** – имя и обозначения для системы координат (**cartesian, cylindrical, spherical** и др.) можно задать систему координат для определенного вектора:

```
> SetCoordinates(v, cylindrical)
```

$$e_r + 2e_\theta + 3e_z$$

```
> vv;
```

$$e_x + 2e_y + 3e_z$$

5) В виде векторного поля (в этом случае базисные векторы будут обозначены чертой сверху). Для векторного поля должна быть указана система координат:

```
> SetCoordinates(spherical[r, phi, psi])
```

$$spherical_{r,\phi,\psi}$$

```
> VectorField(<r, sin(phi), cos(psi)>)
```

$$(r)\bar{e}_r + (\sin(\phi))\bar{e}_\phi + (\cos(\psi))\bar{e}_\psi$$

Скалярное, векторное произведение векторов в пакете VectorCalculus

Команды для вычисления скалярного и векторного произведения аналогичны соответствующим командам пакета **`LinearAlgebra`**.

Скалярное произведение двух векторов или векторных полей вычисляется с помощью команды **`a.b`** или команды **`DotProduct(a,b)`**.

Соответствующая команда пакета **`linalg`**: **`dotprod(a,b)`**.

Векторное произведение двух векторов или векторных полей в трехмерном пространстве вычисляется с помощью команды **`a &x b`** или команды **`CrossProduct(a,b)`**.

```
> with(VectorCalculus):
> v1 := <x, y, 1, 1>
```

$$v1 := (x)e_{x1} + (y)e_{x2} + e_{x3} + e_{x4}$$

```
> v2 := <3, 4, 5, 6>
```

$$v2 := 3e_{x1} + 4e_{x2} + 5e_{x3} + 6e_{x4}$$

```
> DotProduct(v1, v2)
```

$$11 + 3x + 4y$$

```
> DotProduct(v2, v1)
```

$$11 + 3x + 4y$$

```
> a := <2,-2, 1>; b := <2, 3, 6>;
```

$$a := 2e_x - 2e_y + e_z$$

$$b := 2e_x + 3e_y + 6e_z$$

```
> CrossProduct(a, b)
```

$$-15e_x - 10e_y + 10e_z$$

```
> a &x b
```

$$-15e_x - 10e_y + 10e_z$$

Норма вектора в пакете VectorCalculus

Команда для вычисления нормы вектора или векторного поля аналогична соответствующей команде пакета **`LinearAlgebra`**. Общий вид команды:

- **`Norm(A, p)`** – p-норма вектора**.** Второй аргумент команды задает тип нормы.

  Значения параметра p (по умолчанию = `infinity`)

  `2`, `Euclidean` или **`Frobenius`** – Евклидова норма

  `infinity` – максимальный по модулю элемент

Соответствующая команда пакета **`linalg`**: **`norm(a,2)`**.

Нормировать вектор или векторное поле $\boldsymbol{a}$: $\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ можно с помощью команды **`Normalize(a)`** пакета **`VectorCalculus`**.

§6. Векторный анализ в пакете VectorCalculus

Приведем определения основных дифференциальных операций векторного анализа и команды *Maple* для их вычисления. Команды векторного анализа содержатся в пакете **`VectorCalculus`**, а также в подпакетах **`VectorCalculus`** пакета **`Student`** и **`Vectors`** пакета **`Physics`**. Здесь рассмотрим команды пакета **`VectorCalculus`**.

1. Градиент, дивергенция, ротор.

Градиент скалярной функции многих переменных: $f(x1,x2,\ldots,xn):\Re^n\to\Re$ – это вектор, координатами которого являются частные производные по соответствующим переменным: $\text{grad } f(x1,x2,\ldots,xn)=\left(\frac{\partial f}{\partial x1},\frac{\partial f}{\partial x2},\ldots,\frac{\partial f}{\partial xn}\right)$.

В *Maple* градиент функции многих переменных вычисляется командой **`Gradient(f,[x,y,z])`**, где здесь и в дальнейшем **`f`** – функция, **`[x,y,z]`** – набор переменных, от которых она зависит. Синонимами данной команды являются команды векторного дифференциального оператора (набла-оперетора) **`Del(f,[x,y,z])`** и **`Nabla(f,[x,y,z])`**

```
> with(VectorCalculus):
> g1 := Gradient(x^2 + y^2, [x, y])
```

$$g1 := 2\,x\bar{e}_x + 2\,y\bar{e}_y$$

Для набла-оператора можно использовать шаблон:

```
> SetCoordinates('spherical'[r, phi, theta]):
> g2 := ∇(r^2 phi)
```

$$g2 := 2\,r\,\phi\bar{e}_r + (r)\bar{e}_\phi$$

Команда вычисления градиента в пакете **`linalg`**: **`grad(f,[x,y,z],c)`**, где **`f`** – функция, **`[x,y,z]`** – набор переменных, от которых она зависит. Параметр **`c`** позволяет вычислять данную дифференциальную операцию в различных криволинейных координатах (по умолчанию используется прямоугольная декартова система координат). Этот параметр может указываться во всех имеющихся в *Maple* дифференциальных операциях. Для вычисления дифференциальной операции в цилиндрических координатах следует записать **`coords=cylindrical`**, в сферических координатах – **`coords=spherical`**.

Дивергенцией вектор-функции $\boldsymbol{F}(x,y,z)$ называется функция (скалярная), вычисляемая по правилу: $\text{div}\boldsymbol{F}(x,y,z)=\frac{\partial F_x}{\partial x}+\frac{\partial F_y}{\partial y}+\frac{\partial F_z}{\partial z}$. В *Maple* дивергенция векторного поля вычисляется командой **`Divergence(f)`**. Если объект F является вектором, его нужно записать как векторное поле **`vectorfield`**.

```
> with(VectorCalculus):
> F := VectorField(<x^2, y^2, z^2>, 'cartesian'[x, y, z])
```

$$F := (x^2)\bar{e}_x + (y^2)\bar{e}_y + (z^2)\bar{e}_z$$

```
> Divergence(F)
```

$$2\,x + 2\,y + 2\,z$$

Для дивергенции можно использовать шаблон набла-оператора и скалярного умножения векторов:

```
> ∇•F
```

$$2\,x + 2\,y + 2\,z$$

Для объекта вектор дивергенцию вычислить нельзя, его нужно записать как векторное поле:

```
> v := Vector([x^2, y^2, z^2])
```

$$v := (x^2)e_x + (y^2)e_y + (z^2)e_z$$

```
> Divergence(v)
Error, (in VectorCalculus:-Divergence) the input Vector must be a vector field
> v := VectorField(v, 'cartesian'[x, y, z])
```

$$v := (x^2)\bar{e}_x + (y^2)\bar{e}_y + (z^2)\bar{e}_z$$

```
> Divergence(v)
```

$$2x + 2y + 2z$$

Команда вычисления градиента в пакете **linalg**: **diverge(F,[x,y,z],c)**, где **F** – вектор-функция, **[x,y,z]** – набор переменных, от которых она зависит.

Ротором вектор-функции $\boldsymbol{F}(x, y, z)$ в трехмерном пространстве называется вектор с координатами: $\mathrm{rot}\boldsymbol{F} = \left[\left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right), \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right), \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\right]$. В *Maple* ротор векторного поля в 3D вычисляется командой **Curl(f)**. Если объект F является вектором, его нужно записать как векторное поле **vectorfield**.

```
> with(VectorCalculus):
> F := VectorField(⟨y, -x, 0⟩, 'cartesian'[x, y, z])
```

$$F := (y)\bar{e}_x - x\bar{e}_y$$

```
> Curl(F)
```

$$-2\bar{e}_z$$

Для ротора можно использовать шаблон набла-оператора и векторного умножения векторов:

```
> ∇ × F
```

$$-2\bar{e}_z$$

Для объекта вектор дивергенцию вычислить нельзя, его нужно записать как векторное поле:

```
> v := Vector([y, -x, 0])
```

$$v := (y)e_x - xe_y$$

```
> Curl(v)
Error, (in VectorCalculus:-Curl) the input Vector must be a vector field
> v := VectorField(v, 'cartesian'[x, y, z])
```

$$v := (y)\bar{e}_x - x\bar{e}_y$$

```
> Curl(v)
```

$$-2\bar{e}_z$$

Команда вычисления ротора в пакете **linalg**: **curl(F,[x,y,z],c)**.

2. Лапласиан и якобиан

Лапласиан скалярной функции многих переменных: $f(x1,x2,...,xn):\Re^n \to \Re$ – это оператор, действующий по правилу: $\Delta f = \frac{\partial^2 f}{\partial x_1{}^2} + \frac{\partial^2 f}{\partial x_2{}^2} + ... + \frac{\partial^2 f}{\partial x_n{}^2}$.

В *Maple* лапласиан функции многих переменных или векторного поля вычисляется командой **`Laplacian(f,[x,y,z])`**, где здесь и в дальнейшем **`f`** – функция, **`[x,y,z]`** – набор переменных, от которых она зависит.

```
> with(VectorCalculus):Laplacian(x^2+y^2+z^2,[x,y,z])
```

$$6$$

```
> Laplacian(x1^2+x2^2+x3^2+x4^2,[x1,x2,x3,
    x4])
```

$$8$$

В случае, когда объект F является векторным полем, он переводится в декартову систему координат и лапласиан применяется к каждой компоненте векторного поля.

```
> F:=VectorField(<x^2,y^2,z^2>,'cartesian'[x,y,z])
```

$$F := (x^2)\bar{e}_x + (y^2)\bar{e}_y + (z^2)\bar{e}_z$$

```
> Laplacian(F)
```

$$2\bar{e}_x + 2\bar{e}_y + 2\bar{e}_z$$

```
> SetCoordinates(spherical[r,phi,psi]):
> V:=VectorField(<r,sin(phi),cos(psi)>)
```

$$V := (r)\bar{e}_r + (\sin(\phi))\bar{e}_\phi + (\cos(\psi))\bar{e}_\psi$$

```
> simplify(Laplacian(V))
```

$$\frac{2(-2\sin(\phi)\cos(\phi)+\sin(\psi))}{r^2\sin(\phi)}\bar{e}_r + \frac{2(\cos(\phi)\sin(\psi)-\sin(\phi)+\sin(\phi)\cos(\phi)^2)}{r^2\sin(\phi)^2}\bar{e}_\phi - \frac{2\cos(\psi)}{r^2\sin(\phi)^2}\bar{e}_\psi$$

Команда вычисления ротора в пакете **`linalg`**: **`laplacian(f,[x,y,z],c)`**.

Матрицей Якоби вектор-функции ***F***(*x*, *y*, *z*) называется

$$J = \begin{bmatrix} \frac{\partial F_x}{\partial x} & \frac{\partial F_y}{\partial x} & \frac{\partial F_z}{\partial x} \\ \frac{\partial F_x}{\partial y} & \frac{\partial F_y}{\partial y} & \frac{\partial F_z}{\partial y} \\ \frac{\partial F_x}{\partial z} & \frac{\partial F_y}{\partial z} & \frac{\partial F_z}{\partial z} \end{bmatrix}$$

В *Maple* матрица Якоби для вектора или списка алгебраических выражений вычисляется командой **`Jacobian(f,[x,y,z])`**, где **`f`** – функция, **`[x,y,z]`** – набор переменных, от которых она зависит. В случае, когда число переменных равно количеству компонент функции, в команде **`Jacobian`** можно указать третий аргумент **`determinant`** для вычисления определителя матрицы (якобиана): **`Jacobian(f,[x,y,z],'determinant')`**

```
> with(VectorCalculus):
> Jacobian([r*cos(t),r*sin(t),r^2*t],[r,t])
```

$$\begin{bmatrix} \cos(t) & -r\sin(t) \\ \sin(t) & r\cos(t) \\ 2rt & r^2 \end{bmatrix}$$

```
> Jacobian([r*cos(t), r*sin(t), r^2*z], [r, t, z],
      determinant)
```

$$\begin{bmatrix} \cos(t) & -r\sin(t) & 0 \\ \sin(t) & r\cos(t) & 0 \\ 2rz & 0 & r^2 \end{bmatrix}, \cos(t)^2 r^3 + \sin(t)^2 r^3$$

```
> F := VectorField(<x^2, y^2, z^2>, 'cartesian'[x, y, z])
```

$$F := (x^2)\bar{e}_x + (y^2)\bar{e}_y + (z^2)\bar{e}_z$$

```
> Jacobian(F, determinant)
```

$$\begin{bmatrix} 2x & 0 & 0 \\ 0 & 2y & 0 \\ 0 & 0 & 2z \end{bmatrix}, 8xyz$$

Команда вычисления матрицы Якоби в пакете **linalg**: **jacobian(F,[x,y,z])**.

Задание 5

1. Дана функция $u(x,y) = \operatorname{arctg}\frac{y}{x}$. Найти $\operatorname{grad} u(x,y)$. Какие углы составляет grad u с осями координат? Найти производную функции $u(x,y)$ по направлению вектора $\boldsymbol{q}$=[1,1].

```
> restart;
> with(VectorCalculus):
> u := arctan(VectorCalculus[`*`](y, 1/x));
> Gradient(u, [x, y]);
```

$$-\frac{y}{x^2\left(1+\frac{y^2}{x^2}\right)}\bar{e}_x + \left(\frac{1}{x\left(1+\frac{y^2}{x^2}\right)}\right)\bar{e}_y$$

```
> g := simplify(%);
```

$$g := -\frac{y}{x^2+y^2}\bar{e}_x + \left(\frac{x}{x^2+y^2}\right)\bar{e}_y$$

```
> e1 := Vector([1, 0]); e2 := Vector([0, 1]);
```

$$e1 := e_x$$

$$e2 := e_y$$

```
> with(LinearAlgebra):
> alpha1 := simplify(VectorAngle(g, e1));
```

$$\alpha 1 := \pi - \arccos\left(\frac{\left|\frac{y^2}{(x^2+y^2)^2}\right|(x^2+y^2)}{y\sqrt{\left|\frac{y^2}{(x^2+y^2)^2}\right| + \left|\frac{x^2}{(x^2+y^2)^2}\right|}}\right)$$

```
> assume(x::'real', y::'real');
> alpha1 := simplify(alpha1);
```

$$\alpha 1 := \pi - \arccos\left(\frac{y\sim}{\sqrt{x\sim^2 + y\sim^2}}\right)$$

```
> alpha2 := simplify(VectorAngle(g, e2));
```

$$\alpha 2 := \arccos\left(\frac{x\sim}{\sqrt{x\sim^2 + y\sim^2}}\right)$$

Косинусы этих углов являются направляющими косинусами $\mathrm{grad}\,u(x, y)$. Убедимся, что сумма их квадратов равна единице.

```
> simplify(cos(alpha1)^2+ cos(alpha2)^2);
```

$$1$$

Производная функции u по направлению $\boldsymbol{q}$ равна скалярному произведению градиента этой функции на нормированный вектор $\boldsymbol{q}$: $\frac{\partial u}{\partial \boldsymbol{q}} = (\mathrm{grad}u, \boldsymbol{e})$, где $\boldsymbol{e} = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}$ – нормированный вектор $\boldsymbol{q}$ (норма Евклидова).

```
> q := Vector([1, 1]);
```

$$q := e_x + e_y$$

```
> e := Normalize(q, 2);
```

$$e := \begin{bmatrix} \frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} \end{bmatrix}$$

```
> udq := simplify(DotProduct(g, e));
```

$$udq := \frac{1}{2}\,\frac{\sqrt{2}\,(-y\sim + x\sim)}{x\sim^2 + y\sim^2}$$

2. Дана вектор-функция $\boldsymbol{F}(x, y, z)=[x^2yz,\ xy^2z,\ xyz^2]$. Найти $\mathrm{div}\boldsymbol{F}$ и $\mathrm{rot}\boldsymbol{F}$.

```
> restart;
> with(VectorCalculus):
> SetCoordinates(cartesian[x, y, z]);
```

$$cartesian_{x, y, z}$$

```
> F := VectorField(<x^2*y*z, x*y^2*z, x*y*z^2>);
```

$$F := (x^2 y z)\bar{e}_x + (x y^2 z)\bar{e}_y + (x y z^2)\bar{e}_z$$

```
> Divergence(F)
```

$$6 x y z$$

```
> Curl(F)
```

$$(x z^2 - x y^2)\bar{e}_x + (x^2 y - y z^2)\bar{e}_y + (y^2 z - x^2 z)\bar{e}_z$$

3. При каком значении параметра a функция $u=x^3+axy^2$ удовлетворяет уравнению Лапласа $\Delta u=0$?

```
> restart;
> with(VectorCalculus):
> u := x^3+a*x*y^2;
```

$$u := x^3 + a x y^2$$

```
> Delta_u := Laplacian(u, [x, y]);
```

$$Delta\_u := 6 x + 2 a x$$

```
> solve(Delta_u=0,a);
```

$$-3$$

4. Найти матрицу Якоби и ее определитель для вектор-функции $\boldsymbol{v}=[x, y/x]$.

```
> restart;
> with(VectorCalculus):
> v := Vector([x, y/x]);
```

$$v := (x)e_x + \left(\frac{y}{x}\right)e_y$$

```
> Jacobian(v, [x, y],determinant);
```

$$\begin{bmatrix} 1 & 0 \\ -\frac{y}{x^2} & \frac{1}{x} \end{bmatrix}, \frac{1}{x}$$