ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ЮФУ
РОСТОВ-НА-ДОНУ·ТАГАНРОГ

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК
имени И.И. Воровича

Пакеты научных вычислений

Лекция 5

Линейная алгебра и векторный анализ в Maple

Обзор пакетов линейной алгебры и векторного анализа.
Структура матрицы и вектора.
Основные матричные и векторные операции.
Решение задач линейной алгебры.
Векторный анализ

Наседкина А. А.

Пакеты линейной алгебры и векторного анализа

- Обзор пакетов линейной алгебры и векторного анализа
- Определение векторов и матриц (команды из стандартной библиотеки)
- Пакет linalg (устаревший): обзор команд, определение векторов и матриц
- Пакет LinearAlgebra (основной): обзор команд
- Пакет VectorCalculus: обзор команд

Обзор пакетов линейной алгебры

- Небольшая часть команд линейной алгебры находится в стандартной библиотеке (**Vector**, **Matrix** и некоторые другие)
- Основная часть команд линейной алгебры находятся в специальных пакетах

- Пакет **linalg** был единственным пакетом линейной алгебры в старых версиях Maple (до 6-й), в современных версиях считается устаревшим. Недостатки:
  - необходимость использования команды **evalm**
  - ограниченные возможности для работы с числовыми матрицами

- Пакет **LinearAlgebra** (появился в 6-й версии) – основной пакет линейной алгебры для работы с матрицами и векторами, удобен для матричных вычислений
- Пакет **VectorCalculus** (появился в 8-й версии) – команды для векторного анализа

- Подпакеты пакета Student: **Student[LinearAlgebra]**, **Student[VectorCalculus]**

Определение векторов: команды из стандартной библиотеки

Как задать вектор?

- С помощью угловых скобок:
  <a1,a2,…,an> – вектор-столбец
  <a1|a2|…|an> – вектор-строка
- С помощью команды-конструктора **Vector** (большие возможности, см. Help)
  Vector([a1,a2,…,an]) – вектор-столбец
  Vector[row]([a1,a2,…,an]) – вектор-строка

```
> v1 := <1, 2, 3>; whattype(v1)
```

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

$$Vector_{column}$$

```
> w1 := <1|2|3>; whattype(w1)
```

$$w1 := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$Vector_{row}$$

```
> v2 := Vector([4, 5, 6]); whattype(v2)
```

$$v2 := \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$Vector_{column}$$

```
> w2 := Vector[row]([4, 5, 6]); whattype(w2)
```

$$w2 := \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

$$Vector_{row}$$

Команда Vector и ее аргументы

Vector[o](n, init, ro, sym, sh, st, dt, f, a, o) – все аргументы необязательны

o – ориентация вектора: row (строка) или column (столбец – по умолчанию), либо можно задать последний параметр **o** в виде: **orientation=name**

n – число элементов вектора

init – значения элементов вектора, могут задаваться функцией, процедурой, списком, массивом и др.

ro – булево выражение в виде **readonly=true** или **false**, определяет, можно ли изменять элементы вектора

sym – задает символьные значения элементов вектора в виде **symbol=name**

sh – задает одну или несколько индексирующих функций для элементов вектора в виде **shape=name** или **shape=list**

st – задает способ хранения элементов в виде **storage=name** (пр.: sparse)

dt – задает тип данных элементов в виде **datatype=name**

f – заполняет незаданные элементы вектора в виде **fill=value**, где **value** – значение типа **datatype**

a – задает дополнительные свойства вектора в виде **attributes=list**

Команда Vector: примеры

По умолчанию элементы вектора - нули

```
> Vector(2)
```

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

```
> Vector(1..2, 1.25)
```

$$\begin{bmatrix} 1.25 \\ 1.25 \end{bmatrix}$$

```
> Vector(4, symbol = v, orientation = row)
```

$$\begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}$$

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

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

```
> v := Vector(4, [5, 8]) + Vector(4, fill = 10)
```

$$v := \begin{bmatrix} 15 \\ 18 \\ 10 \\ 10 \end{bmatrix}$$

```
> s := {1 = 28, 2 = -I + 1} : Vector(2, s)
```

$$\begin{bmatrix} 28 \\ 1 - I \end{bmatrix}$$

Задание элементов 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])
```

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

Запрет изменения элементов вектора

```
> v := Vector[row]([1, 2, 3], readonly = true) :
> v[1] := 10
Error, cannot assign to a read-only
Vector
```

Определение матриц: команды из стандартной библиотеки

Как задать матрицу?

- С помощью угловых скобок:
  <<a11,…,an1>|<a12,…,an2>|….|<a1m,…,anm>> – матрица размера n x m, заданная по столбцам
  <<a11|…|a1m>,<a21|….|a2m>,…,<an1|…|anm>> – матрица размера n x m , заданная по строкам
- С помощью команды-конструктора **Matrix** (большие возможности, см. Help)
  Matrix([[a11,a12,…,a1m],[a21,a22,…,a2m],…,[an1,an2,…,anm]])

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

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

Matrix

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

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

Matrix

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

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

Команда Matrix и ее аргументы

Matrix(r, c, init, ro, sym, sc, sh, st, o, dt, f, a) – все аргументы необязательны
r – число строк матрицы
c – число столбцов матрицы
init – значения элементов матрицы, могут задаваться функцией, процедурой, списком, матрице, вектором и др.
ro – булево выражение в виде **readonly=true** или **false**, определяет, можно ли изменять элементы матрицы
sym – задает символьные значения элементов матрицы в виде **symbol=name**
sc – определяет способ заполнения матрицы элементами **init** в виде **scan=name** или **scan=list** (например, **scan=columns** – заполнение по столбцам)
sh – задает одну или несколько индексирующих функций для элементов матрицы в виде **shape=name** или **shape=list** (пр.: diagonal, triangular[upper], triangular[lower], symmetric, identity и т. д.)
st – задает способ хранения элементов в виде **storage=name** (sparse, diagonal и др.)
o – задает способ хранения структуры матрицы: order= C_order (хранение по строкам) или order= Fortran_order (хранение по столбцам – по умолчанию)
dt – задает тип данных элементов в виде **datatype=name**
f – заполняет незаданные элементы матрицы в виде **fill=value**, где **value** – значение типа **datatype**
a – задает дополнительные свойства матрицы в виде **attributes=list**

Команда Matrix: примеры

Задание квадратной матрицы
По умолчанию все элементы - нули

```
> Matrix(2)
```

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

Задание размеров матрицы, заполнение элементами

```
> Matrix(1..2, 1..3, 50)
```

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

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

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 0 & 0 \end{bmatrix}$$

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

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

```
> 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)
```

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

```
> s := {(1, 1) = 55, (1, 2) = 66, (2, 1) = 77};
  Matrix(2, 3, s)
```

$$s := \{(1, 1) = 55, (1, 2) = 66, (2, 1) = 77\}$$

$$\begin{bmatrix} 55 & 66 & 0 \\ 77 & 0 & 0 \end{bmatrix}$$

Указание 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 единичной матрицы

```
> Matrix(3, shape = identity)
```

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

Определение верхнетреугольной матрицы

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

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

Команда Matrix: примеры (продолжение)

Определение нижнетреугольной матрицы

```
> Matrix(3, [[1], [2, 3], [4, 5, 6]], shape = triangular[lower])
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

Определение диагональной матрицы

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

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

Задание элементов через функцию

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

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

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

Заполнение по столбцам (по умолчанию элементы заполняются по строкам)

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

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

Матрица из одного столбца не является вектором, нужна конвертация типа

```
> B := Matrix(2, 1, [30, 40]); whattype(M)
```

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

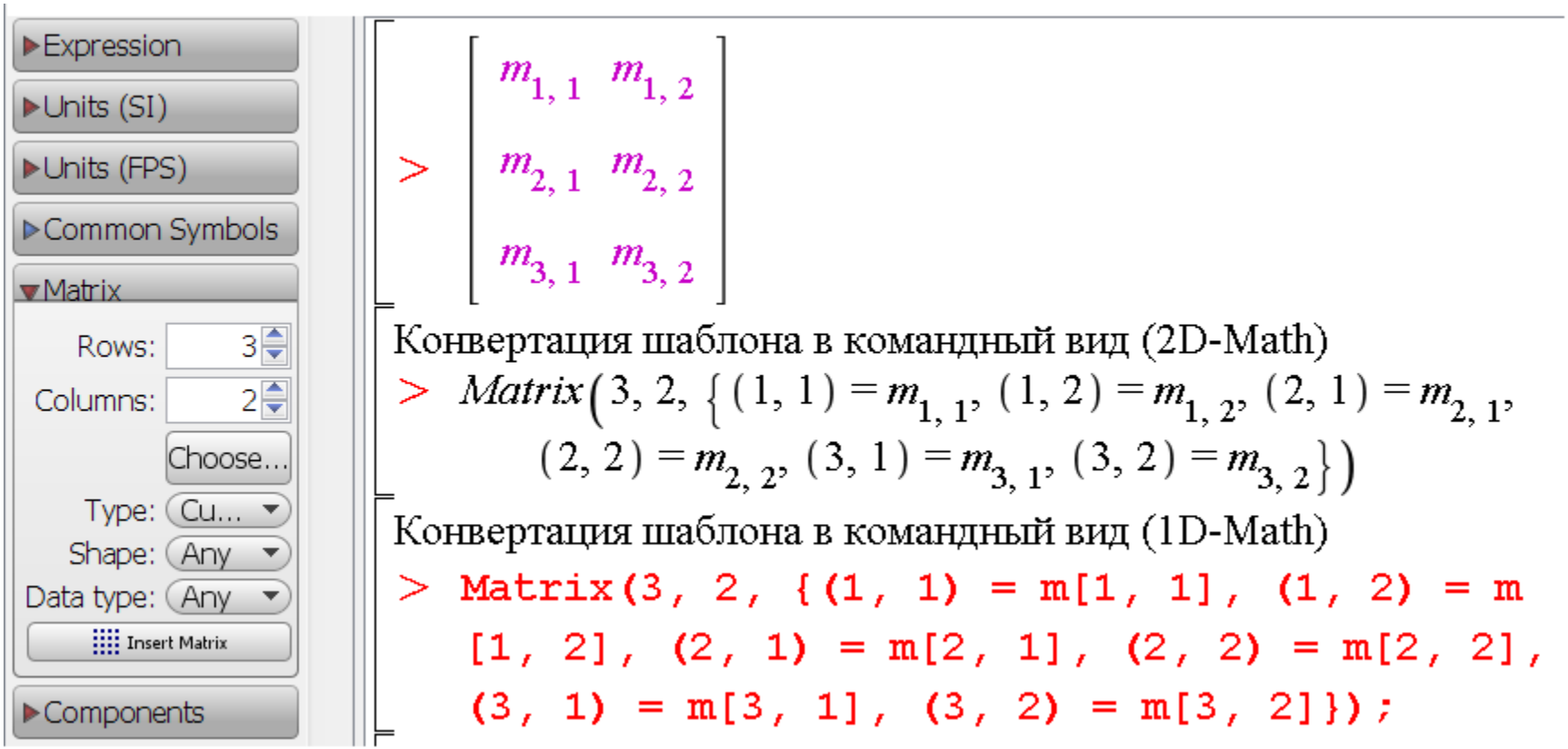
$$Matrix$$

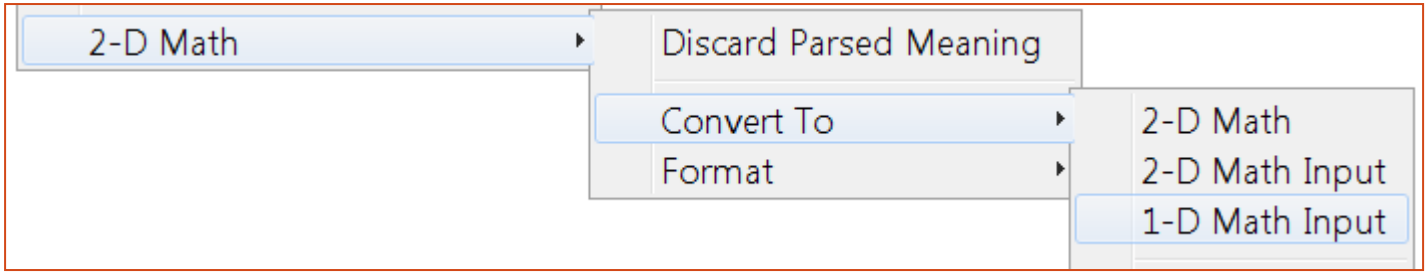
```
> b := convert(B, Vector); whattype(b)
```

$$b := \begin{bmatrix} 30 \\ 40 \end{bmatrix}$$

$$Vector_{column}$$

Задание матриц с помощью шаблонов





Просмотр больших матриц

- Матрицы и векторы размером более 10 не выводятся на экран, просмотр элементов осуществляется в браузере

```
> with(LinearAlgebra):
> RandomMatrix(15)
```

15 x 15 Matrix

Data Type: anything

Storage: rectangular

Order: Fortran_order

Double-click

Browse Matrix

Table | Image | Options

	1	2	3	4	5	6	7
1	44	90	83	-29	20	-94	35
2	92	-41	-45	9	-46	27	-26
3	73	-79	68	81	35	18	-86
4	-39	9	58	35	-54	18	50
5	62	45	-43	80	-17	63	-94
6	11	-10	-85	20	-25	86	-97
7	61	-5	-85	39	78	-51	-38
8	28	47	19	-35	23	51	-36
9	-48	-54	25	26	-67	38	-69
10	-63	-72	17	-74	28	-38	69
11	27	-79	81	13	-81	-19	-15
12	58	75	89	32	-36	-55	2
13	2	-85	92	48	-88	71	-88

Insert | Export | Done

Пакет linalg (устаревший): обзор команд

- **linalg[command](arguments)**
- **with(linalg): command(arguments)**

Команды для задания матриц и векторов: **matrix** и **vector**, использование аналогично использованию команд Matrix и Vector

```
> with(linalg)
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol,
    addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat,
    charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto,
    crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals,
    eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim,
    fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad,
    hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis,
    inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve,
    matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace,
    orthog, permanent, pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim,
    rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector,
    sumbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent,
    vectdim, vector, wronskian]
```

Задание векторов и матриц в пакете linalg

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

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

true

symbol

```
> l := [1, 2, 3]: type(l, vector); whattype(l)
```

false

list

```
> lv := convert(l, vector); whattype(lv)
```

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

symbol

```
> lV := convert(l, Vector); whattype(lV)
```

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

$Vector_{column}$

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

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

```
> type(A, matrix); whattype(A)
```

true

symbol

```
> AM := convert(A, Matrix); whattype(AM)
```

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

Matrix

Команда array для задания векторов и матриц

Для задания векторов и матриц для последующей работы с командами пакета **linalg** можно использовать команду **array**

```
> d := array(1..2, [4, 5]); type(d, vector); whattype(d)
```

$$d := \begin{bmatrix} 4 & 5 \end{bmatrix}$$

true

symbol

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

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

true

symbol

С помощью array можно задавать матрицы специального вида

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

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

```
> array( 1..2, 1..2, diagonal);
```

$$\begin{bmatrix} \text{`?`}_{1,1} & 0 \\ 0 & \text{`?`}_{2,2} \end{bmatrix}$$

Пакет LinearAlgebra : обзор команд

- **LinearAlgebra[command](arguments)**
- **with(LinearAlgebra): command(arguments)**

```
> with(LinearAlgebra)
```

[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, LA_Main, LUDecomposition, LeastSquares, LinearSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]

Пакет VectorCalculus: обзор команд

- **VectorCalculus[command](arguments)**
- **with(VectorCalculus): command(arguments)**

```
> with(VectorCalculus)
[&x, `*`, `+`, `-`, `.`, <,>, <|>, About, AddCoordinates, ArcLength, BasisFormat,
    Binormal, Compatibility, ConvertVector, CrossProd, CrossProduct, Curl,
    Curvature, D, Del, DirectionalDiff, Divergence, DotProd, DotProduct, Flux,
    GetCoordinateParameters, GetCoordinates, GetPVDescription, GetRootPoint,
    GetSpace, Gradient, Hessian, 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]
```

Структура матрицы и вектора

- Команды для задания матриц и векторов специального вида
- Доступ к элементам векторов и матриц
- Выяснение размерностей векторов и матриц
- Операции со столбцами и строками матрицы

Команды-конструкторы для задания матриц и векторов специального вида

```
> 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 матрица с целыми числами

```
> RandomMatrix( 3 )
```

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

Диапазон целых чисел: −99..99
Изменение диапазона – опция generator=a..b
(необязательно целый диапазон, см. Help)

Доступ к элементам векторов и матриц

> $v := \langle 23.5, -1.7, 8.2 \rangle$

$$v := \begin{bmatrix} 23.5 \\ -1.7 \\ 8.2 \end{bmatrix}$$

> $v[1]$

$$23.5$$

Выделение подвектора

> $v[1..2]$

$$\begin{bmatrix} 23.5 \\ -1.7 \end{bmatrix}$$

> $v[-1]$

$$8.2$$

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

> $M[2, 1]$

$$-7$$

Выделение подматрицы:
1-й аргумент - номера строк,
2-й аргумент - номера столбцов

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

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

Выяснение размерностей векторов и матриц

LinearAlgebra:

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

linalg:

- **vectdim(v)** – размерность вектора
- **rowdim(A)** – число строк матрицы
- **coldim(A)** – число столбцов матрицы

```
> with(LinearAlgebra):
> M := RandomMatrix(2, 3)
```

$$M := \begin{bmatrix} 33 & -77 & 27 \\ -98 & 57 & -93 \end{bmatrix}$$

```
> Dimension(M)
```

$$2, 3$$

```
> v := Vector([5, 6, 7]) : Dimension(v)
```

$$3$$

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

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

```
> RowDimension(A)
```

$$2$$

```
> ColumnDimension(A)
```

$$3$$

Операции со столбцами и строками матрицы: удаление строк и столбцов

LinearAlgebra:

- **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

  outopts - (необязательный параметр) опции **outputoptions** для результирующего объекта, например outputoptions=[datatype=float,shape=….]

linalg:

- **delrows(A,i..j)** – удаление строк матрицы **A**
- **delcols(A,i..j)** – удаление столбцов матрицы **A**

```
> with(LinearAlgebra):
> A := ⟨⟨1, 2, 3⟩|⟨4, 5, 6⟩|⟨7, 8, 9⟩⟩
```

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

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

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

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

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

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

$$\begin{bmatrix} 2 & 5 & 8 \end{bmatrix}$$

Операции со столбцами и строками матрицы: извлечение строк и столбцов

LinearAlgebra:

- **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

  outopts - (необязательный параметр) опции **outputoptions** для результирующего объекта, например outputoptions=[datatype=float,shape=….]

linalg:

- **row(A,i)** – извлечение строки **i** матрицы **A**
- **col(A,j)** – извлечение столбца **j** матрицы **A**

```
> with(LinearAlgebra):
> A := RandomMatrix(4)
```

$$A := \begin{bmatrix} 12 & -16 & 45 & 87 \\ -2 & -9 & -81 & 33 \\ 50 & -50 & -38 & -98 \\ 10 & -22 & -18 & -77 \end{bmatrix}$$

```
> r2 := Row(A, 2); whattype(r2)
```

$$r2 := \begin{bmatrix} -2 & -9 & -81 & 33 \end{bmatrix}$$

$$Vector_{row}$$

```
> c3 := Column(A, 3); whattype(c3)
```

$$c3 := \begin{bmatrix} 45 \\ -81 \\ -38 \\ -18 \end{bmatrix}$$

$$Vector_{column}$$

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

$$B := \begin{bmatrix} 12 & -16 & 45 & 87 \end{bmatrix}, \begin{bmatrix} -2 & -9 & -81 & 33 \end{bmatrix}, \begin{bmatrix} 50 & -50 & -38 & -98 \end{bmatrix}$$

$$Vector_{row}$$

Операции со столбцами и строками матрицы: элементарная перестановка строк и столбцов

LinearAlgebra:

RowOperation(A, K, s, ip, outopts) – операции со строками матрицы **A**

ColumnOperation(A, K, s, ip, outopts) – операции со столбцами матрицы **A**

A – матрица; K – целое число или список двух целых чисел; s – алгебраическое выражение; ip – выражение вида inplace=true/false (optional), определяет, изменять ли матрицу A; outopts – выражение вида outputoptions=list; inplace и outputopts являются взаимоисключающими

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

linalg:

- **swaprow(A,ri,rj)** – перестановка строк матрицы **A**
- **swapcol(A,ci,cj)** – перестановка столбцов матрицы **A**

```
> with(LinearAlgebra):
> A := RandomMatrix(3)
```

$$A := \begin{bmatrix} 86 & -48 & 31 \\ 20 & 77 & -50 \\ -61 & 9 & -80 \end{bmatrix}$$

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

$$\begin{bmatrix} -61 & 9 & -80 \\ 20 & 77 & -50 \\ 86 & -48 & 31 \end{bmatrix}$$

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

$$\begin{bmatrix} 86 & 31 & -48 \\ 20 & -50 & 77 \\ -61 & -80 & 9 \end{bmatrix}$$

Операции со столбцами и строками матрицы: сложение и умножение строк и столбцов

LinearAlgebra:

С помощью команд RowOperation/ColumnOperation

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

linalg:

- **addrow(A,ri,rj,expr)** – изменение строки **ri**: **ri:=ri*expr+rj**, где **expr** – число или выражение (сложение строк)
- **addcol(A,ci,cj,expr)** – изменение столбца **ci**: **ci:=ci*expr+cj**, где **expr** – число или выражение (сложение строк)
- **mulrow(A,r,expr)** – матрица, получаемая из матрицы **A** с помощью умножения строки **r** на выражение **expr**
- **mulcol(A,c,expr)** – аналогично для столбца

Операции со столбцами и строками матрицы: сложение и умножение строк и столбцов

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

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

```
> RowOperation(A, [1, 3], 100)#сложение строк
```

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

```
> ColumnOperation(A, 2, 10)#умножение столбца
```

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

Выделение подматрицы (подвектора), минора

LinearAlgebra:

- **SubMatrix(A, r, c, outopts)** – выделение подматрицы из матрицы **A**, **r** – диапазон (номера) строк, **c** - диапазон (номера) стобцов
- **SubVector(V, i, outopts)** – выделение подвектора из матрицы **V**, **i** – диапазон (номера) элементов
- **Minor**(**A, r, c, out, meth, outopts)** – вычисление минора M(i,j) к элементу A[i,j] матрицы **A** (по умолчанию выдается определитель), **out** задает тип результата в виде **output=matrix** или/и **output=determinant** (определитель), **meth** – метод вычисления определителя в виде **method=value**, возможные значения **value** см. в Help

linalg:

- **submatrix(A,ri..rj,ci..cj)** – выделение подматрицы
- **subvector(v,i..j)** – выделение подвектора
- **minor(A,i,j)** – возвращает матрицу, полученную вычеркиванием строки **i** и столбца **j** матрицы **A**
- **det(minor(A,i,j)**)– вычисление минора

Примеры выделения подматрицы (подвектора) и вычисления минора

```
> with(LinearAlgebra) : A := Matrix(3, [[1, 2, 3], [4, 5, 6], [7, 0, 1]])
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 0 & 1 \end{bmatrix}$$

```
> SubMatrix(A, [1, 2], [2 ..3])
```

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

```
> V := Vector[row]([1, 2, 3, 4, 5, 6])
```

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

```
> SubVector(V, [2, 4 .. -1, 1])
```

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

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

$$-14$$

```
> Minor(A, 1, 3, output = ['matrix','determinant'])
```

$$\begin{bmatrix} 4 & 5 \\ 7 & 0 \end{bmatrix}, -35$$

```
> Minor(A, 2, 2, output = ['matrix'], outputoptions = [datatype = float])
```

$$\begin{bmatrix} 1. & 3. \\ 7. & 1. \end{bmatrix}$$

Основные матричные и векторные операции

- Сложение и умножение на число
- Матричное умножение
- Возведение в степень
- Обратная матрица, транспонированная или эрмитово-сопряженная матрица
- Определитель, ранг, след матрицы

Сложение и умножение на число

LinearAlgebra:

- **A+B –** сложение матриц или векторов **A** и **B**
- **A*c –** умножение элементов матрицы (вектора) **A** на скаляр **c**
- Команды: **Add(A,B)** или **MatrixAdd(A,B)** и **Multiply(A,c)**

linalg:

- **evalm(A+B)** или **matadd(A,B) –** сложение матриц или векторов **A** и **B**
- **matadd(A,B,c,d) –** линейная комбинация c**A**+d**B**: **evalm(A*c+B*d)**
- **evalm(A*c) –** умножение на скаляр

```
> 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; A · 100
```

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

$$\begin{bmatrix} 100 & 200 \\ 300 & 400 \end{bmatrix}$$

```
> v := ⟨1, 2, 3⟩ : 10 · v
```

$$\begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

Матричное и матрично-векторное умножение

LinearAlgebra:

- **A.B** – матричное (некоммутативное) умножение матриц и векторов
- **MatrixVectorMultiply(A, u)** – умножение матрицы **A** на вектор **u**
- **MatrixMatrixMultiply(A, B)** – умножение матрицы **A** на матрицу **B**
- Общая команда: **Multiply(A,B)**

linalg:

- **evalm(A&*B)**– произведение **AB** матриц **A** и **B**
- **multiply(A,B)** – произведение **AB** матриц **A** и **B**

```
> 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;
```

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

```
> 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)
```

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

Возведение в степень

LinearAlgebra:

- **A^n –** возведение квадратной матрицы **A** в степень **n**
- **A^(-1) –** вычисление обратной матрицы (если существует)

linalg:

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

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

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

```
> A^3
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

```
> with(linalg) : B := matrix([[1, 2], [3, 4]]);
```

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

```
> evalm(B^3)
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

Вычисление обратной матрицы

LinearAlgebra:

- **A^(-1)** – вычисление обратной матрицы
- **MatrixInverse(A)**

Полный синтаксис: **MatrixInverse(A, m, mopts, c, out, outopts)** –см. Help

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

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

```
> MatrixInverse(A)
```

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

Матрица с линейно-зависимыми столбцами

```
> C := Matrix([[3, 5], [6, 10]], scan = columns)
```

$$C := \begin{bmatrix} 3 & 6 \\ 5 & 10 \end{bmatrix}$$

Такая матрица вырождена (не существует обратной)

```
> MatrixInverse(C)
Error, (in LinearAlgebra:-LA_Main:-MatrixInverse) singular matrix
```

linalg:
evalm(1/A)
inverse(A)

```
> with(linalg): B := matrix([[1, 2], [3, 4]]);
```

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

```
> evalm(1/B);
```

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

```
> inverse(B)
```

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

Вычисление транспонированной и эрмитово-сопряженной матрицы

LinearAlgebra:

- **A^(%T)** – транспонированная матрица A^T, команда:

  Transpose(A)

- **A^(%H)** – эрмитово-сопряженная матрица $A^H = \overline{A^T}$ (операция транспонирования и комплексного сопряжения элементов), команда:

  HermitianTranspose(A)

linalg:

- **transpose(A) и htranspose(A)**

```
> 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)
```

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

Определитель, ранг, след матрицы

LinearAlgebra:

- **Determinant(A)** – вычисление определителя матрицы **A**
- **Rank(A)** – ранг матрицы
- **Trace(A)** – след матрицы (сумма диагональных элементов)

linalg:

- **det(A)** – определитель
- **rank(A)** – ранг
- **trace(A)** – след

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

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

```
> Determinant(A)
```

$$-2$$

```
> Rank(A)
```

$$2$$

```
> Trace(A)
```

$$5$$

Матрица с линейно-зависимыми столбцами

```
> C := Matrix([[3, 5], [6, 10]], scan = columns)
```

$$C := \begin{bmatrix} 3 & 6 \\ 5 & 10 \end{bmatrix}$$

У такой матрицы ранг меньше размерности

```
> Rank(C)
```

$$1$$

Такая матрица вырождена (определитель равен нулю)

```
> Determinant(C)
```

$$0$$

Решение задач линейной алгебры

- Нормы матриц и векторов
- Проверка равенства двух матриц
- Выяснение типа матрицы (положительная/отрицательная определенность; ортогональность и унитарность)
- Спектральный анализ: собственные числа и собственные векторы, характеристический многочлен
- Решение систем линейных уравнений

Нормы векторов

LinearAlgebra:

- **Norm(A, p, c)** – p-норма матрицы или вектора
- **VectorNorm(A, p, c)** – p-норма вектора, c – (необязательные) опции для результирующего объекта

Значения параметра p (по умолчанию = infinity)

- для векторов
  - **2, Euclidean** или **Frobenius** – Евклидова норма $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
  - **p**(>0) – p-норма

```
VectorNorm(A,p)=add( abs(V[i])^p, i = 1 ..
Dimension(V)) ^ (1/p)
```

  - **infinity** – максимальный по модулю элемент

linalg:

- **norm(A, normname)** – норма матрицы или вектора

  norm(A) – infinity-норма матрицы или вектора (по умолчанию)
- Для векторов **normname** может быть **:** целое число>=1, **'infinity', 'frobenius'**

Нормы матриц

LinearAlgebra:

- **Norm(A, p, c)** – p-норма матрицы или вектора
- **MatrixNorm(A, p, c)** – p-норма матрицы, c – (необязательные) опции для результирующего объекта

Значения параметра p (по умолчанию = infinity)

- для матриц
- **1** – максимальная столбцовая норма

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

```
MatrixNorm(A, 1) = max(seq(VectorNorm(A[1..-1, j],
  1), j = 1 .. ColumnDimension(A)))
```

- **infinity** – максимальная строчная норма

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

```
MatrixNorm(A, infinity) = max(seq(VectorNorm(A[i,
  1..-1], 1), i = 1 .. RowDimension(A)))
```

Нормы матриц (продолжение)

Значения параметра p (по умолчанию = infinity)

- **2** или **Euclidean** – спектральная норма (корень из максимального по модулю собственного значения матрицы $A^H A$)

$$\|A\|_2 = \sqrt{\max_i |\lambda_i(A^H A)|} \quad \forall A \in C^{n\times m}, \qquad A^H = \overline{A^T}$$

```
MatrixNorm(A, 2) = sqrt(max(seq(Eigenvalues(A.
A^(%H))[i], i = 1 .. RowDimension(A))))
```

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

  (ненастоящая матричная норма, т.е. не связана с векторными нормами)

```
MatrixNorm(A, Frobenius) =
sqrt(add(add(abs(A[i,j])^2, j = 1 ..
ColumnDimension(A)), i = 1 .. RowDimension(A)))
```

linalg:

- **norm(A, normname)** – норма матрицы или вектора

  norm(A) – infinity-норма матрицы или вектора (по умолчанию)
- Для матриц **normname** может быть **1, 2, 'infinity', 'frobenius'.**

Примеры вычисления норм векторов и матриц

```
> with(LinearAlgebra):
> v := Vector([1,-2, 3])
```

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

По умолчанию - infinity-норма (макс. по модулю элемент)

```
> VectorNorm(v)
```

$$3$$

1-норма (сумма модулей элементов)

```
> VectorNorm(v, 1)
```

$$6$$

Евклидова норма

```
> VectorNorm(v, 2)
```

$$\sqrt{14}$$

```
> VectorNorm(v, Euclidean)
```

$$\sqrt{14}$$

```
> VectorNorm(v, Frobenius)
```

$$\sqrt{14}$$

3-норма (кубический корень из суммы модулей кубов элементов)

```
> VectorNorm(v, 3)
```

$$36^{1/3}$$

infinity-норма (макс. по модулю элемент)

```
> VectorNorm(v, infinity)
```

$$3$$

```
> A := Matrix([[10, 0, 2], [0, 9, 1], [2, 4, 1]])
```

$$A := \begin{bmatrix} 10 & 0 & 2 \\ 0 & 9 & 1 \\ 2 & 4 & 1 \end{bmatrix}$$

По умолчанию - максимальная строчная норма (infinity)

```
> MatrixNorm(A)
```

$$12$$

Максимальная столбцовая норма (1)

```
> MatrixNorm(A, 1)
```

$$13$$

Спектральная норма

```
> MatrixNorm(A, 2)
```

$$\sqrt{RootOf\left(-196 + \_Z^3 - 207\,\_Z^2 + 10577\,\_Z, index = 3\right)}$$

Часто вычисляется только в приближенном виде

```
> evalf(%)
```

$$10.73324841$$

Евклидова норма матрицы = спектральная норма

```
> MatrixNorm(A, Euclidean)
```

$$\sqrt{RootOf\left(-196 + \_Z^3 - 207\,\_Z^2 + 10577\,\_Z, index = 3\right)}$$

Норма Фробениуса (ненастоящая матричная норма)

```
> MatrixNorm(A, Frobenius)
```

$$3\sqrt{23}$$

Максимальная строчная норма (infinity)

```
> MatrixNorm(A, infinity)
```

$$12$$

Проверка равенства двух матриц

LinearAlgebra:

- **Equal(A,B)** – проверка логического равенства матриц A и B, результат – true/false
- **MatrixNorm(A-B,1)** или **MatrixNorm(A-B,infinity)** – проверка равенства матриц по норме, для приближенных значений коэффициентов

  Если A≃B, то A–B ≃нулевая матрица, норма которой близка к нулю

  linalg: equal(A,B) для проверки логического равенства

```
> with(LinearAlgebra):
> A := RandomMatrix(3)
```

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

Обратная матрица

```
> invA := MatrixInverse(A)
```

$$invA := \begin{bmatrix} -\frac{3307}{327244} & \frac{2585}{327244} & \frac{5737}{327244} \\ \frac{2675}{327244} & \frac{4539}{327244} & -\frac{1573}{327244} \\ \frac{1649}{327244} & -\frac{5643}{327244} & \frac{9}{327244} \end{bmatrix}$$

Проверка: A*A^(-1)=E

```
> A.invA
```

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

```
> E := IdentityMatrix(3)
```

$$E := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Проверка равенства A*A^(-1) и E

```
> Equal(A.invA, E)
```

$$true$$

Проверка равенства двух матриц: пример для приближенных значений

```
> with(LinearAlgebra):
> A := RandomMatrix(3)
```

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

```
> E := IdentityMatrix(3)
```

$$E := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Обратная матрица в приближенном виде

```
> invA1 := evalf(invA)
```

$$invA1 := \begin{bmatrix} -0.01010560927 & 0.007899304494 & 0.01753126108 \\ 0.008174328636 & 0.01387038418 & -0.004806810820 \\ 0.005039053428 & -0.01724401364 & 0.00002750241410 \end{bmatrix}$$

Проверка: A*A^(-1)=E

```
> A.invA1
```

$$\begin{bmatrix} 1.00000000005000000 & 2.78000057002403268\ 10^{-10} & 7.71999649536864175\ 10^{-11} \\ 1.60000259569192949\ 10^{-11} & 1.00000000001199996 & 2.28999896844696294\ 10^{-11} \\ 3.00000597280858372\ 10^{-11} & 1.26000026384742726\ 10^{-10} & 1.00000000018470004 \end{bmatrix}$$

Проверка равенства A*A^(-1) и E

```
> Equal(A.invA1, E)
```

$$false$$

Проверка по норме

```
> MatrixNorm(A.invA1 - E, 1)
```

$$4.16000039971109458\ 10^{-10}$$

```
> MatrixNorm(A.invA1 - E, infinity)
```

$$4.05200026093108234\ 10^{-10}$$

Выяснение типа матрицы: положительная (отрицательная) определенность

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, т.е.:
    $x^H Ax > 0 \ \forall x \in C^n, \ x \neq 0, \ A \in C^{n \times n}$ для комплексной матрицы, где
    x^H – сопряженный транспонированный вектор, A – эрмитова матрица: $A = A^H$
    Для вещественной матрицы такой запрос возвратит true, если
    $x^T Ax > 0 \ \forall x \in R^n, \ x \neq 0, \ A \in R^{n \times n}$, где x^T – транспонированный вектор,
    При этом матрица A может быть несимметричной
    Эквивалентное определения: все собственные значения положительны: СЗ>0
  - **'positive_semidefinite'** – положительно полуопределенная: $x^H Ax \geq 0$ (СЗ>=0)
  - **'negative_definite'** – отрицательно определенная: $x^H Ax < 0$ (СЗ<0)
  - **'negative_semidefinite'** – отрицательно полуопределенная: $x^H Ax \leq 0$ (СЗ<=0)
  - **'indefinite'** – неопределенная
- **IsDefinite(A)** – проверяет положительную определенность матрицы **A** т. е. по умолчанию **query= 'positive_definite'**

Положительная/отрицательная определенность матрицы: продолжение и примеры

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)
```

Матрица положительно определена

true

```
> A = A^%T
```

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

```
> Equal(A, A^%T)
```

Матрица симметричная

true

```
> Eigenvalues(A); evalf(%)
```

Все собственные значения положительны

$$\begin{bmatrix} \frac{5}{2} + \frac{1}{2}\sqrt{5} \\ \frac{5}{2} - \frac{1}{2}\sqrt{5} \end{bmatrix}$$

$$\begin{bmatrix} 3.618033988 \\ 1.381966012 \end{bmatrix}$$

```
> with(linalg):
> AA := matrix(2, 2, [2, 1, 1, 3])
```

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

```
> definite(AA,'positive_def')
```

true

Положительная/отрицательная определенность матрицы: примеры

> $B := DiagonalMatrix([-5, 0, -1])$

$$B := \begin{bmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Матрица не является положительно определенной

> $IsDefinite(B)$

$$false$$

Матрица не является положительно полуопределенной

> $IsDefinite(B, 'query'='positive\_semidefinite')$

$$false$$

Одинарные кавычки в запросе можно опустить

Матрица не является отрицательно определенной

> $IsDefinite(B, query = negative\_definite)$

$$false$$

Матрица является отрицательно полуопределенной

> $IsDefinite(B, query = negative\_semidefinite)$

$$true$$

> $Eigenvalues(B);$

Все собственные значения неотрицательны

$$\begin{bmatrix} 0 \\ -1 \\ -5 \end{bmatrix}$$

Матрица симметричная

> $Equal(B, B^{\%T})$

$$true$$

> $A1 := Matrix([[3, 2], [1, 4]])$

$$A1 := \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix} \quad (1)$$

> $A2 := DiagonalMatrix([2, 2])$

$$A2 := \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad (2)$$

> $A := DiagonalMatrix([A1, A2])$

$$A := \begin{bmatrix} 3 & 2 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad (3)$$

Матрица является положительно определенной

> $IsDefinite(A, query = positive\_definite)$

$$true \quad (4)$$

Но при этом матрица не является симметричной

> $Equal(A, A^{\%T})$

$$false \quad (5)$$

Собственные значения положительны

> $Eigenvalues(A)$

$$\begin{bmatrix} 5 \\ 2 \\ 2 \\ 2 \end{bmatrix} \quad (6)$$

Выяснение типа матрицы: ортогональность и унитарность

LinearAlgebra:

- **IsOrthogonal(A)** – проверяет, является ли матрица **A** ортогональной, т.е. такой, что $AA^T = A^T A = E$, где E – единичная матрица. Эквивалентно: $A^T = A^{-1}$
- **IsUnitary(A)** – проверяет, является ли комплексная матрица **A** унитарной: т. е. такой, что $AA^{\mathrm{H}} = A^{\mathrm{H}} A = E$, где H – эрмитово сопряжение. Эквивалентно: $A^H = A^{-1}$

linalg:

- **orthog(A)** – проверяет, является ли матрица **A** ортогональной

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

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

```
> IsOrthogonal(A)
```

$$false$$

```
> Q := <<sqrt(10)·3/10, -sqrt(10)/10> | <sqrt(10) I/10, 3 sqrt(10) I/10>>
```

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

```
> IsOrthogonal(Q)
```

$$false$$

```
> IsUnitary(Q)
```

$$true$$

```
> with(linalg):
> B := matrix([[-1/2, sqrt(3)/2], [sqrt(3)/2, 1/2]])
```

$$B := \begin{bmatrix} -\frac{1}{2} & \frac{1}{2}\sqrt{3} \\ \frac{1}{2}\sqrt{3} & \frac{1}{2} \end{bmatrix}$$

```
> orthog(B)
```

$$true$$

Спектральный анализ: общие сведения о собственных числах и собственных векторах

Определения из курса линейной алгебры

Пусть $\boldsymbol{A}$ – квадратная матрица размера n×n, в общем случае комплексная.

- Если $\boldsymbol{Au} = \lambda \boldsymbol{u}$, то вектор **u** называется собственным вектором матрицы $\boldsymbol{A}$, а число λ – собственным числом (значением), соответствующим данному собственному вектору.
- Совокупность всех собственных чисел матрицы называется спектром матрицы.
- Для матрицы размера n×n количество собственных чисел равно n.
- Если в спектре матрицы одно и тоже собственное число встречается k раз, то говорят, что это собственное число кратное и его (алгебраическая) кратность равна k.
- Собственные числа матрицы $\boldsymbol{A}$ являются корнями характеристического многочлена $P_A(\lambda) = \det(\boldsymbol{A} - \lambda \boldsymbol{E})$, где $\boldsymbol{E}$ – единичная матрица.
- Алгебраическая кратность собственного числа λ – это его кратность как корня характеристического многочлена.
- Матрица $\boldsymbol{A} - \lambda \boldsymbol{E}$ называется характеристической матрицей для матрицы $\boldsymbol{A}$

Команды пакеты LinearAlgebra:

- **Eigenvalues(A)** – возвращает собственные значения матрицы **A**
- **Eigenvectors(A)** – возвращает собственные значения матрицы **A** в виде вектор-столбца и матрицу из собственных векторов (по столбцам)
- **CharacteristicPolynomial(A, lambda)** – характеристический многочлен
- **CharacteristicMatrix(A, lambda, outopts)** – характеристическая матрица

Спектральный анализ: собственные числа

LinearAlgebra:

Полный синтаксис:

- **Eigenvalues(A, C, imp, o, outopts)** – остальные аргументы необязательны

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

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

  o – задает формат вывода результата в виде: **output =obj**, где **obj** может принимать значения **'Vector[row]'**, **'Vector[column]'**, **'list'** или список этих значений

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

linalg:

- **eigenvalues(A)** или **eigenvals(A)**
- Варианты: **eigenvalues(A, C), eigenvalues(A, 'implicit'), eigenvalues(A, 'radical')**

В стандартной библиотеке:

- **Eigenvals(A, vecs)** – команда отложенного исполнения для числовой матрицы **A,** возвращает массив из собственных чисел и матрицу из собственных векторов по столбцам(в параметре **vecs**)

Примеры вычисления собственных чисел: Eigenvalues

```
> with(LinearAlgebra) :
> A := Matrix(3, 3, [x, 0, y, x, y, 0, y, 0, x])
```

$$A := \begin{bmatrix} x & 0 & y \\ x & y & 0 \\ y & 0 & x \end{bmatrix}$$

Собственные числа в виде вектор-столбца

```
> Eigenvalues(A)
```

$$\begin{bmatrix} y \\ y+x \\ x-y \end{bmatrix}$$

Собственные числа в виде списка

```
> Eigenvalues(A, output = list)
```

$$[y, y+x, x-y]$$

```
> B := Matrix([[3, -I, 0], [I, 3, 0], [0, 0, 4]])
```

$$B := \begin{bmatrix} 3 & -I & 0 \\ I & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

```
> Eigenvalues(B, output = list)
```

$$[2, 4, 4]$$

```
> M := <<1, 4, -2>|<-1, 0, 1>|<-1, 2, 1>>
```

$$M := \begin{bmatrix} 1 & -1 & -1 \\ 4 & 0 & 2 \\ -2 & 1 & 1 \end{bmatrix}$$

```
> Eigenvalues(M, output = list)
```

$$[2, I, -I]$$

Собственные числа в неявном виде

```
> Eigenvalues(M, implicit, output = list)
```

$$[2, \text{RootOf}(\_Z^2 + 1, index = 1), \text{RootOf}(\_Z^2 + 1, index = 2)]$$

Задание опций outputoptions конструктора

```
> R := RandomMatrix(3, outputoptions = [shape = symmetric, storage
      = rectangular, datatype = float])
```

$$R := \begin{bmatrix} 30. & -23. & 91. \\ -23. & 63. & -38. \\ 91. & -38. & -38. \end{bmatrix}$$

Собственные числа в виде вектор-строки

```
> Eigenvalues(R, output = Vector[row])
```

$$[-103.221603816942689, 36.3950990097986775,$$
$$121.826504807143976]$$

Примеры вычисления собственных чисел: Eigenvals и команды пакета linalg

Использование команды Eigenvals

```
> C := array([[1, 2], [3, 4]])
```

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

```
> Eigenvals(C, vecs); vecs
```

$$Eigenvals\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, vecs\right)$$

$$vecs$$

```
> evalf(Eigenvals(C, vecs)); print(vecs)
```

$$\begin{bmatrix} -4.527868129 & 0.4417089775 \end{bmatrix}$$

$$\begin{bmatrix} -.8245648401 & -.4222291504 \\ 0.5657674650 & -.9230523142 \end{bmatrix}$$

Собственные векторы по столбцам

Использование команд пакета linalg

```
> with(linalg) : A := matrix(3, 3, [x, 0, y, x, y, 0, y, 0, x]) :
> eigenvals(A)
```

$$y, y + x, x - y$$

Спектральный анализ: собственные векторы

LinearAlgebra:

- **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 уравнения 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'**, **'vectors'**, **'list'** или список этих значений

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

linalg:

- **eigenvectors(A)** или **eigenvects(A)**
- Варианты: **eigenvectors(A, 'implicit'), eigenvectors(A, 'radical')**

В стандартной библиотеке:

- **Eigenvals(A, vecs)** – команда отложенного исполнения для числовой матрицы **A**, возвращает массив из собственных чисел и матрицу из собственных векторов по столбцам(в параметре **vecs**)

Примеры вычисления собственных векторов

```
> with(LinearAlgebra) :
> A := Matrix(3, 3, [x, 0, y, x, y, 0, y, 0, x])
```

$$A := \begin{bmatrix} x & 0 & y \\ x & y & 0 \\ y & 0 & x \end{bmatrix}$$

Собственные числа в виде вектор-столбца и матрица из собственных векторов по столбцам

```
> Eigenvectors(A)
```

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

Только матрица из собственных векторов по столбцам

```
> Eigenvectors(A, output = vectors)
```

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

Список из собственных значений, их кратностей и соответствующих им собственных векторов

```
> Eigenvectors(A, output = list)
```

$$\left[\left[y, 1, \left\{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right\}\right], \left[y+x, 1, \left\{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right\}\right], \left[x-y, 1, \left\{\begin{bmatrix} -1 \\ -\frac{x}{-2y+x} \\ 1 \end{bmatrix}\right\}\right]\right]$$

```
> B := Matrix([[3, -I, 0], [I, 3, 0], [0, 0, 4]])
```

$$B := \begin{bmatrix} 3 & -I & 0 \\ I & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Пример кратного собственного значения

```
> Eigenvectors(B, output = list)
```

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

Использование команд пакета linalg

```
> with(linalg) :
> B := matrix([[-1/2, sqrt(3)/2], [sqrt(3)/2, 1/2]])
```

$$B := \begin{bmatrix} -\frac{1}{2} & \frac{1}{2}\sqrt{3} \\ \frac{1}{2}\sqrt{3} & \frac{1}{2} \end{bmatrix}$$

Собственное значение, его крантность и соотв. собст. вектор в виде вектор-строки

```
> eigenvectors(B)
```

$$\left[-1, 1, \left\{\begin{bmatrix} 1 & -\frac{1}{3}\sqrt{3} \end{bmatrix}\right\}\right], \left[1, 1, \left\{\begin{bmatrix} \frac{1}{3}\sqrt{3} & 1 \end{bmatrix}\right\}\right]$$

Примеры вычисления собственных чисел и собственных векторов

```
> A := <<2, 3, 1>|<4, 6, -1>|<-1, -3/2, -2>>;
```

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

```
> Rank(A)
```

$$2$$

```
> Eigenvalues(A)
```

$$\begin{bmatrix} 0 \\ 3 + \frac{1}{2}\sqrt{102} \\ 3 - \frac{1}{2}\sqrt{102} \end{bmatrix}$$

Собственный вектор, соответствующий нулевому собственному значению, является также вектором ядра матрицы (Ax=0)

```
> x := NullSpace(A)
```

$$x := \left\{ \begin{bmatrix} \frac{3}{2} \\ -\frac{1}{2} \\ 1 \end{bmatrix} \right\}$$

```
> A.x[1]
```

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

В матрице неполного ранга будут нулевые собственные значения

```
> Eigenvectors(A, output = list)
```

$$\left[0, 1, \left\{ \begin{bmatrix} \frac{3}{2} \\ -\frac{1}{2} \\ 1 \end{bmatrix} \right\} \right], \left[3 + \frac{1}{2}\sqrt{102}, 1, \left\{ \begin{bmatrix} -\frac{49\left(3 + \frac{1}{2}\sqrt{102}\right)}{\left(-123 + \frac{25}{2}\sqrt{102}\right)\left(1 + \frac{1}{2}\sqrt{102}\right)} \\ -\frac{3}{2}\,\frac{45 + 2\sqrt{102}}{-123 + \frac{25}{2}\sqrt{102}} \\ 1 \end{bmatrix} \right\} \right], \left[3 - \frac{1}{2}\sqrt{102}, 1, \left\{ \begin{bmatrix} -\frac{49\left(3 - \frac{1}{2}\sqrt{102}\right)}{\left(-123 - \frac{25}{2}\sqrt{102}\right)\left(1 - \frac{1}{2}\sqrt{102}\right)} \\ -\frac{3}{2}\,\frac{45 - 2\sqrt{102}}{-123 - \frac{25}{2}\sqrt{102}} \\ 1 \end{bmatrix} \right\} \right]$$

Точные числа с радикалами и дробями, удобно вывести в приближенном виде

```
> evalf(%)
```

$$\left[\left[0., 1., \left\{ \begin{bmatrix} 1.500000000 \\ -0.5000000000 \\ 1. \end{bmatrix} \right\} \right], \left[8.049752470, 1., \left\{ \begin{bmatrix} -20.09950451 \\ -30.14925675 \\ 1. \end{bmatrix} \right\} \right], \left[-2.049752470, 1., \left\{ \begin{bmatrix} 0.09950493837 \\ 0.1492574075 \\ 1. \end{bmatrix} \right\} \right] \right]$$

Матрица с нулевым собственным значением будет вырожденной

```
> MatrixInverse(A)
Error, (in LinearAlgebra:-LA_Main:-MatrixInverse) singular matrix
```

Спектральный анализ: характеристический многочлен и характеристическая матрица

LinearAlgebra:

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

linalg:

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

```
> with(LinearAlgebra):
> M := Matrix([[1, 1, 0], [0, 1, 3], [0, 0, 2]])
```

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

```
> CharacteristicPolynomial(M, x)
```

$$-2 + x^3 - 4x^2 + 5x$$

```
> solve(%, x)
```

$$2, 1, 1$$

```
> Eigenvalues(M, output='list')
```

$$[2, 1, 1]$$

```
> CharacteristicMatrix(M, lambda)
```

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

Решение систем линейных уравнений, ядро матрицы

LinearAlgebra:

- **LinearSolve(A,B)** – решение уравнения AX=B, где B – матрица или вектор правой части, X – матрица или вектор неизвестных. Полный синтаксис:

  LinearSolve(A, B, m, t, c, ip, outopts, methopts) – остальные аргументы необязательны (подробности – см. Help). Значения некоторых параметров:

  m – параметр используемого метода в виде **method = name,** где **name** может принимать значения **'none', 'solve', 'subs', 'Cholesky', 'LU', 'QR', 'hybrid', 'modular', 'SparseLU', 'SparseDirect'** или **'SparseIterative'**

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

  GenerateEquations(A, [x, y, z], b)
- **GenerateMatrix(eqns, vars)** – генерирование матрицы коэффициентов из списка (множества) уравнений **eqns** и списка (множества) неизвестных **vars**

  GenerateMatrix([eq1, e2, eq3], [x, y, z])

linalg:

- **linsolve(A,b)** – решение системы Ax=b, **kernel(A)** – ядро матрицы A
- **geneqns (A,[x,y,z],v), genmatrix(eqns,[x,y,z])**

Примеры решения системы линейных уравнений и поиска ядра матрицы

```
> with(LinearAlgebra):
  sys := [ 3*x[1] + 2*x[2] + 3*x[3] = 1,
           x[1] +  x[2] +  x[3]     = 3,
           x[1] + 2*x[2] - x[3]     = 2 ]:
  var := [ x[1], x[2], x[3] ]:
  (A, b) := GenerateMatrix( sys, var );
```

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

```
> LinearSolve(A, b)
```

$$\begin{bmatrix} -\frac{19}{2} \\ 8 \\ \frac{9}{2} \end{bmatrix}$$

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

$$s := [3x + 2y + 3z = 1, x + y + z = 3, x + 2y - z = 2]$$

```
> solve(s)
```

$$\left\{z = \frac{9}{2}, y = 8, x = -\frac{19}{2}\right\}$$

```
> with(LinearAlgebra):
  A := <<6,3,0>|<4,2,0>|<2,1,0>>;
```

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

Ядро

```
> kern := NullSpace(A);
```

$$kern := \left\{\begin{bmatrix} -\frac{1}{3} \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -\frac{2}{3} \\ 1 \\ 0 \end{bmatrix}\right\}$$

Проверка

```
> A.kern[1];A.kern[2];
```

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

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

Основные случаи при решении СЛАУ: случай 1

```
> restart; with(LinearAlgebra) : with(plots) :
```

1) Существует единственное решение

```
> A1 := Matrix([[1,-1], [1, 2]]); b1 := Vector([1, 4])
```

$$A1 := \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}$$

$$b1 := \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

```
> LinearSolve(A1, b1)
```

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

```
> Determinant(A1)
```

$$3$$

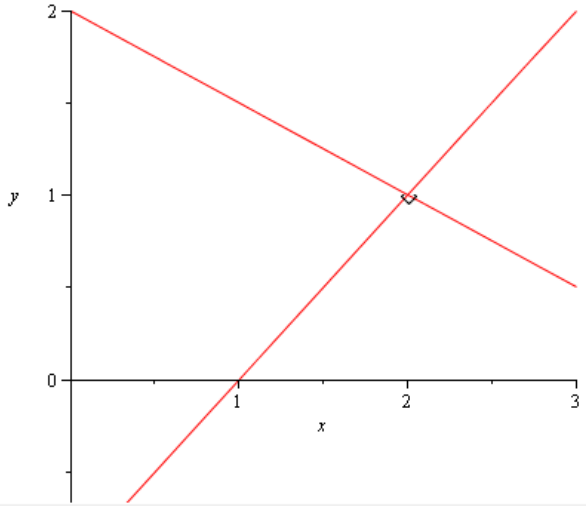
```
> sys := GenerateEquations(A1, [x, y], b1)
```

$$sys := [x - y = 1, x + 2y = 4]$$

```
> sol := solve(sys); assign(sol); x1 := x : y1 := y : unassign('x','y')
```

$$sol := \{x = 2, y = 1\}$$

```
> display([implicitplot([sys[1], sys[2]], x =-3 ..3, y =-1 ..2), pointplot([x1, y1], symbolsize = 20)])
```



Основные случаи при решении СЛАУ: случаи 2 и 3

2) Бесконечно много решений

```
> A2 := Matrix([[1,-1], [-2, 2]]); b2 := Vector([1,-2])
```

$$A2 := \begin{bmatrix} 1 & -1 \\ -2 & 2 \end{bmatrix}$$

$$b2 := \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

```
> LinearSolve(A2, b2)
```

$$\begin{bmatrix} 1 + \_t0_2 \\ \_t0_2 \end{bmatrix}$$

```
> Determinant(A2); MatrixInverse(A2)
```

$$0$$

```
Error, (in LinearAlgebra:-LA_Main:-MatrixInverse) singular matrix
```

$b \in \{u \mid u=Ax\}$

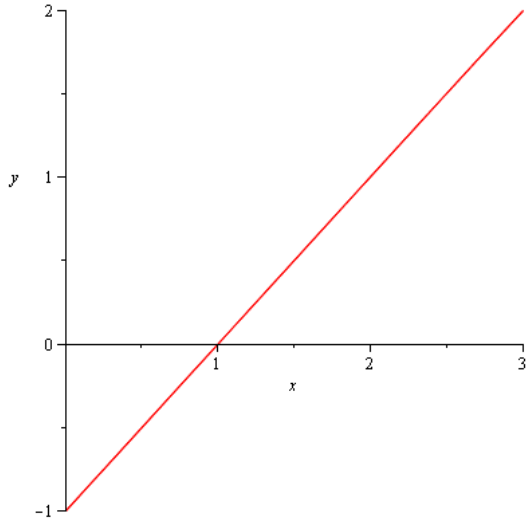
```
> sys := GenerateEquations(A2, [x, y], b2)
```

$$sys := [x - y = 1, -2x + 2y = -2]$$

```
> sol := solve(sys);
```

$$sol := \{x = 1 + y, y = y\}$$

```
> implicitplot([sys[1], sys[2]], x = -3..3, y = -1..2)
```



3) Нет решений

```
> A3 := Matrix([[1,-1], [1,-1]]); b3 := Vector([1,-1])
```

$$A3 := \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$b3 := \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

```
> LinearSolve(A3, b3);
Error, (in LinearAlgebra:-LA_Main:-LinearSolve) inconsistent
system
> Determinant(A3); MatrixInverse(A3)
```

$$0$$

```
Error, (in LinearAlgebra:-LA_Main:-MatrixInverse) singular
matrix
```

$b \notin \{u \mid u=Ax\}$

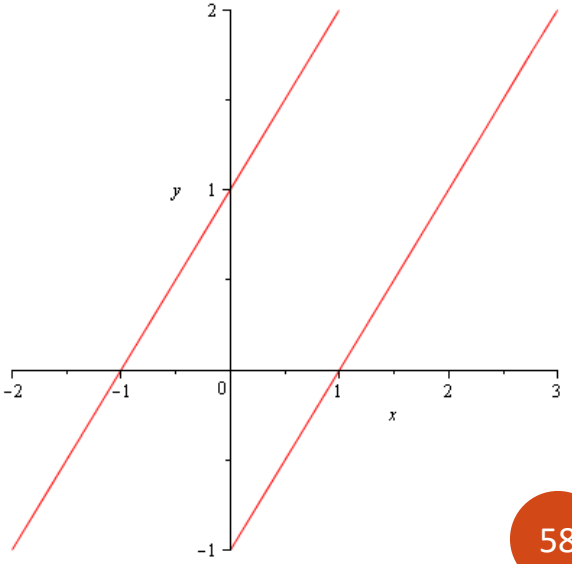
```
> sys := GenerateEquations(A3, [x, y], b3)
```

$$sys := [x - y = 1, x - y = -1]$$

```
> sol := solve(sys);
```

$$sol :=$$

```
> implicitplot([sys[1], sys[2]], x = -3..3, y = -1..2)
```



Для решения СЛАУ

- Факторизация матриц: QR- и LU-разложение
- Приведение матриц к специальному виду: верхнетреугольная форма, жорданова форма

Факторизация матриц: QR-разложение

LinearAlgebra:

- **QRDecomposition(A)** – QR-разложение матрицы: A=QR, где Q – ортогональная матрица, R – верхнетреугольная матрица.
- 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.

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

linalg:

- **QRdecomp(A)**

Пример вычисления QR-разложения матрицы

```
> with(LinearAlgebra):
> A := <<2, 3, 1> | <4, 6, -1> | <-1, -3/2, -2>>
```

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

```
> Rank(A)
```

Т.о. A - матрица неполного ранга

$$2$$

Обычное QR-разложение (усеченное)

```
> Q, R := QRDecomposition(A)
```

$$Q, R := \begin{bmatrix} \frac{1}{7}\sqrt{14} & \frac{1}{91}\sqrt{182} \\ \frac{3}{14}\sqrt{14} & \frac{3}{182}\sqrt{182} \\ \frac{1}{14}\sqrt{14} & -\frac{1}{14}\sqrt{182} \end{bmatrix}, \begin{bmatrix} \sqrt{14} & \frac{25}{14}\sqrt{14} & -\frac{17}{28}\sqrt{14} \\ 0 & \frac{3}{14}\sqrt{182} & \frac{3}{28}\sqrt{182} \end{bmatrix}$$

Размерности матриц Q, R:

```
> mQ, nQ := Dimensions(Q); mR, nR := Dimensions(R);
```

$$mQ, nQ := 3, 2$$

$$mR, nR := 2, 3$$

Проверка: A=QR

```
> Q.R
```

$$\begin{bmatrix} 2 & 4 & -1 \\ 3 & 6 & -\frac{3}{2} \\ 1 & -1 & -2 \end{bmatrix}$$

```
> Equal(A, Q.R)
```

$$true$$

Полное QR-разложение (матрица R - квадратная верхне треугольная)

```
> Qf, Rf := QRDecomposition(A, fullspan)
```

$$Qf, Rf := \begin{bmatrix} \frac{1}{7}\sqrt{14} & \frac{1}{91}\sqrt{182} & \frac{3}{13}\sqrt{13} \\ \frac{3}{14}\sqrt{14} & \frac{3}{182}\sqrt{182} & -\frac{2}{13}\sqrt{13} \\ \frac{1}{14}\sqrt{14} & -\frac{1}{14}\sqrt{182} & 0 \end{bmatrix}, \begin{bmatrix} \sqrt{14} & \frac{25}{14}\sqrt{14} & -\frac{17}{28}\sqrt{14} \\ 0 & \frac{3}{14}\sqrt{182} & \frac{3}{28}\sqrt{182} \\ 0 & 0 & 0 \end{bmatrix}$$

Размерности матриц Qf, Rf:

```
> mQf, nQf := Dimensions(Qf); mR, nR := Dimensions(Rf);
```

$$mQf, nQf := 3, 3$$

$$mR, nR := 3, 3$$

Проверка: A=Qf*Rf

```
> Qf.Rf
```

$$\begin{bmatrix} 2 & 4 & -1 \\ 3 & 6 & -\frac{3}{2} \\ 1 & -1 & -2 \end{bmatrix}$$

```
> Equal(A, Qf.Rf)
```

$$true$$

Факторизация матриц: LU-разложение

LinearAlgebra:

- **LUDecomposition(A)** – LU-разложение матрицы, такое что: A=PLU, где L – нижнетреугольная матрица, U – верхнетреугольная матрица, P – матрица перестановки (единичная матрица с переставленными строками).
- 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'** (для A=PLU1R); **'NAG', 'determinant', 'rank'** или список этих значений.

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

linalg:

- **Ludecomp(A)**

Пример вычисления LU-разложения матрицы

```
> with(LinearAlgebra):
> A := <<0, -2, 0, 3>|<1, 3, 0, 1>|<1, 1, 0, 0>|<-3, 4, 1, 0>>
```

$$A := \begin{bmatrix} 0 & 1 & 1 & -3 \\ -2 & 3 & 1 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{bmatrix}$$

Матрицы: перестановки P, нижнетреугольная L, верхнетреугольная U

```
> P, L, U := LUDecomposition(A)
```

$$P, L, U := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{2} & \frac{11}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Проверка: A=PLU

```
> P.L.U
```

$$\begin{bmatrix} 0 & 1 & 1 & -3 \\ -2 & 3 & 1 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{bmatrix}$$

```
> Equal(A, P.L.U)
```

$$true$$

Верхнетреугольная форма

```
> LUDecomposition(A, output = 'U')
```

$$\begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> GaussianElimination(A)
```

$$\begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Пример вычисления LU-разложения для симметричной положительно определенной матрицы

```
> with(LinearAlgebra):
> B := RandomMatrix(3)
```

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

Создадим симметричную матрицу

```
> A := B.B^%T
```

$$A := \begin{bmatrix} 18994 & 235 & 12383 \\ 235 & 1866 & -249 \\ 12383 & -249 & 11186 \end{bmatrix}$$

Проверка матрицы на симметричность

```
> Equal(A, A^%T)
```

$$true$$

Проверка матрицы на положительную определенность

```
> IsDefinite(A)
```

$$true$$

```
> P, L, U := LUDecomposition(A)
```

$$P, L, U := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ \frac{235}{18994} & 1 & 0 \\ \frac{12383}{18994} & -\frac{7639511}{35387579} & 1 \end{bmatrix}, \begin{bmatrix} 18994 & 235 & 12383 \\ 0 & \frac{35387579}{18994} & -\frac{7639511}{18994} \\ 0 & 0 & \frac{107088635536}{35387579} \end{bmatrix}$$

Проверка: A=LU

```
> Equal(A, L.U)
```

$$true$$

Пример вычисления LU-разложения для симметричной положительно определенной матрицы (продолжение)

Использование метода Холецкого дает нижнетреугольную матрицу с неединичной диагональю

> $Lh := LUDecomposition(A, method = 'Cholesky')$

$$Lh := \begin{bmatrix} \sqrt{18994} & 0 & 0 \\ \frac{235}{18994}\sqrt{18994} & \frac{1}{18994}\sqrt{672151675526} & 0 \\ \frac{12383}{18994}\sqrt{18994} & -\frac{7639511}{672151675526}\sqrt{672151675526} & \frac{92}{35387579}\sqrt{447732461015171} \end{bmatrix}$$

Проверка: A=Lh*Lh'

> $Lh.Transpose(Lh)$

$$\begin{bmatrix} 18994 & 235 & 12383 \\ 235 & 1866 & -249 \\ 12383 & -249 & 11186 \end{bmatrix}$$

> $Equal(A, Lh.Transpose(Lh))$

$$true$$

Приведение матриц к специальному виду: верхнетреугольная форма

LinearAlgebra:

- **GaussianElimination(A)** – приведение матрицы к верхнетреугольной форме методом исключения Гаусса. Эквивалент команды **LUDecomposition(A, output=['U'])**
- 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. Эквивалент команды **LUDecomposition(A, output=[‘R'])**

linalg:

- **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

Приведение к верхнетреугольной форме методом Гаусса: примеры

```
> with(LinearAlgebra):
  AA := Matrix(3, 3, [x, 1, 0, 0, 0, 1, 1, y, 1]);
```

$$AA := \begin{bmatrix} x & 1 & 0 \\ 0 & 0 & 1 \\ 1 & y & 1 \end{bmatrix}$$

```
> GaussianElimination(AA)
```

$$\begin{bmatrix} x & 1 & 0 \\ 0 & \frac{yx-1}{x} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> GaussianElimination(AA,'method'='FractionFree')
```

$$\begin{bmatrix} x & 1 & 0 \\ 0 & yx-1 & x \\ 0 & 0 & yx-1 \end{bmatrix}$$

```
> ReducedRowEchelonForm(AA)
```

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

```
> with(linalg): A := matrix(3, 3, [x, 1, 0, 0, 0, 1, 1, y, 1])
```

$$A := \begin{bmatrix} x & 1 & 0 \\ 0 & 0 & 1 \\ 1 & y & 1 \end{bmatrix}$$

```
> gausselim(A)
```

$$\begin{bmatrix} x & 1 & 0 \\ 0 & \frac{yx-1}{x} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> ffgausselim(A)
```

$$\begin{bmatrix} x & 1 & 0 \\ 0 & yx-1 & x \\ 0 & 0 & yx-1 \end{bmatrix}$$

```
> gaussjord(A)
```

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

Приведение матриц к специальному виду: жорданова форма

LinearAlgebra:

- **JordanForm(A)** – нормальная жорданова форма. Полный синтаксис:
  JordanForm(A, out, outopts, …) – остальные аргументы необязательны
  out – параметр в виде output = ‘J’ (жорданова форма) или output = ‘Q’ (матрица перехода)
  outopts – задает опции **outputoptions** для результирующего объекта

linalg:

- **jordan(A)**

$$\begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & 0 & \lambda \end{pmatrix}$$

```
> with(LinearAlgebra):
> A := <<2, 4, 0, 0>|<4, 6, 0, 4>|<-6, -3, 4, 6>|<0, -4, 0, 2>>
```

$$A := \begin{bmatrix} 2 & 4 & -6 & 0 \\ 4 & 6 & -3 & -4 \\ 0 & 0 & 4 & 0 \\ 0 & 4 & 6 & 2 \end{bmatrix}$$

```
> J := JordanForm(A)
```

$$J := \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

```
> Eigenvalues(A, output = list)
```

$$[6, 4, 2, 2]$$

Матрица перехода

```
> Q := JordanForm(A, output = 'Q')
```

$$Q := \begin{bmatrix} 24 & -\frac{15}{2} & -30 & -15 \\ \frac{27}{2} & -\frac{15}{2} & 0 & -\frac{15}{2} \\ 1 & 0 & 0 & 0 \\ 30 & -\frac{15}{2} & -30 & -\frac{45}{2} \end{bmatrix}$$

Проверка: J=Q^(-1)AQ

```
> Q^(-1).A.Q;
```

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Векторный анализ в пакетах LinearAlgebra и VectorCalculus

- Особенности задания векторов и векторных полей в пакете VectorCalculus
- Скалярное и векторное произведение, угол между векторами
- Базис системы векторов, ортогональный базис
- Градиент, дивергенция, ротор
- Лапласиан
- Матрица Якоби

Особенности задания векторов и векторных полей в пакете VectorCalculus

- **Vector[o](n, init, f, c)** – задает вектор в заданной системе координат (по умолчанию декартова); все параметры являются необязательными; по умолчанию выводит разложение вектора по базису (**BasisFormat(true)**)
  o – ориентация вектора (**row** или **column**)
  init – значения элементов вектора, могут задаваться функцией, процедурой, списком, массивом и др.
  f – заполняет незаданные элементы вектора в виде **fill=value**
  c – задает систему координат в виде **coords=name** или **coordinates=name**
- **<x1,x2,…,xn>** – также задает вектор в заданной системе координат
- Многие команды пакета VectorCalculus требуют векторное поле, а не вектор, в качестве входного аргумента
- **VectorField(v, c)** – задает векторное поле в заданной системе координат
  v – список list или вектор Vector компонент вектора в заданной системе координат
  c – задает координатную систему и имена для координат в виде **symbol[name, name, …]**
- **SetCoordinates(v,c)** – задает глобальную систему координат для векторов и векторных полей; **v, c** определены как выше

Примеры задания векторов и векторных полей в пакете VectorCalculus

```
> with(VectorCalculus):
```

здесь команда Vector берется из подключенного пакета

```
> v := Vector([3, 4, 5])
```

$$v := 3e_x + 4e_y + 5e_z$$

```
> v1 := ⟨x, 2 y, 3 xz⟩
```

$$v1 := (x)e_x + 2\,ye_y + 3\,xze_z$$

```
> v2 := ⟨x|2 y|3 xz⟩
```

$$v2 := (x)e_x + 2\,ye_y + 3\,xze_z$$

Векторное поле в декартовой системе координат

```
> V := VectorField(⟨3, 4, 5⟩, 'cartesian'[x, y, z])
```

$$V := 3\bar{e}_x + 4\bar{e}_y + 5\bar{e}_z$$

Векторное поле в цилиндрической системе координат

```
> V1 := VectorField(⟨r cos(θ), sin(θ), z^2⟩, 'cylindrical'[r, θ, z])
```

$$V1 := (r\cos(\theta))\bar{e}_r + (\sin(\theta))\bar{e}_\theta + (z^2)\bar{e}_z$$

Векторное поле в сферической системе координат

```
> SetCoordinates('spherical'[r, ϕ, θ])
```

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

```
> V2 := VectorField(⟨1/r^2, sin(ϕ), cos(θ)⟩)
```

$$V2 := \left(\frac{1}{r^2}\right)\bar{e}_r + (\sin(\phi))\bar{e}_\phi + (\cos(\theta))\bar{e}_\theta$$

```
> v3 := Vector([1, 2, 3])
```

$$v3 := e_r + 2e_\phi + 3e_\theta$$

Скалярное и векторное произведение векторов

- **v1.v2** – скалярное умножение векторов

LinearAlgebra, VectorCalculus:

- **DotProduct(v1,v2)**

linalg: dotprod(v1,v2)

$$(x, y) = \sum_{i=1}^{n} x_i \bar{y}_i = y^H x$$

```
> v1 := ⟨x, y, 1, 1⟩
```

$$v1 := \begin{bmatrix} x \\ y \\ 1 \\ 1 \end{bmatrix}$$

```
> v2 := ⟨3, 4, 5, 6⟩
```

$$v2 := \begin{bmatrix} 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

```
> v1. v2
```

$$11 + 3\,\overline{x} + 4\,\overline{y}$$

```
> v2. v1
```

$$11 + 3\,x + 4\,y$$

```
> with(LinearAlgebra) :
> DotProduct(v1, v2)
```

$$11 + 3\,\overline{x} + 4\,\overline{y}$$

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

$$11 + 3\,x + 4\,y$$

LinearAlgebra, VectorCalculus:

- **v1 &x v2** – векторное произведение векторов в трехмерном пространстве
- **CrossProduct(v1,v2)**

linalg: crossprod(v1,v2)

$$\left|\left[\bar{a} \times \bar{b}\right]\right| = |\bar{a}| \cdot |\bar{b}| \cdot \sin \angle(\bar{a}; \bar{b})$$

```
> with(LinearAlgebra) :
> a := ⟨2, -2, 1⟩; b := ⟨2, 3, 6⟩; c := a &x b
```

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

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

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

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

$$\begin{bmatrix} -15 \\ -10 \\ 10 \end{bmatrix}$$

Угол между векторами, норма и нормализация вектора

LinearAlgebra:

- **VectorAngle(v1,v2)**

linalg: angle(v1,v2)

```
> with(LinearAlgebra):
> V1 := ⟨1, 0, 1⟩; V2 := ⟨1, 1, 0⟩
```

$$V1 := \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$V2 := \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

```
> VectorAngle(V1, V1)
```

$$0$$

```
> VectorAngle(V1, V2)
```

$$\frac{1}{3}\pi$$

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

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

LinearAlgebra:

- **Norm(v, p, c)** – p-норма вектора
- **VectorNorm(v, p, c)** – p-норма вектора, c – (необязательные) опции для результирующего объекта
- **Normalize(v)** – нормализация вектора

Значения параметра p (по умолчанию = infinity)

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

infinity – максимальный по модулю элемент

linalg: norm(v,p), normalize(v)

```
> Norm(v, 2)
```

$$3\sqrt{2}$$

```
> Norm(v, infinity)
```

$$3$$

```
> Normalize(v)
```

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

Нахождение базиса системы векторов. Ортогонализация Грамма-Шмидта

LinearAlgebra:

- **Basis([v1, v2, ..., vn], outopts)**

linalg: basis([v1, v2, ..., vn])

- **GramSchmidt([v1, v2, ..., vn])** – ортогональная система векторов, ортонормированная, если задать опцию **normalized=true**

```
> 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([a1, a2, a3, a4, a5])
```

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

```
> GramSchmidt([a1, a2, a3, a4, a5], normalized = true)
```

$$\left[\begin{bmatrix} \frac{1}{10}\sqrt{10} \\ \frac{1}{5}\sqrt{10} \\ \frac{1}{5}\sqrt{10} \\ -\frac{1}{10}\sqrt{10} \end{bmatrix}, \begin{bmatrix} \frac{1}{13}\sqrt{26} \\ \frac{3}{26}\sqrt{26} \\ -\frac{3}{26}\sqrt{26} \\ \frac{1}{13}\sqrt{26} \end{bmatrix}, \begin{bmatrix} \frac{27}{23530}\sqrt{11765} \\ -\frac{31}{23530}\sqrt{11765} \\ \frac{109}{23530}\sqrt{11765} \\ \frac{183}{23530}\sqrt{11765} \end{bmatrix}, \begin{bmatrix} \frac{23}{362}\sqrt{181} \\ -\frac{13}{362}\sqrt{181} \\ -\frac{1}{362}\sqrt{181} \\ -\frac{5}{362}\sqrt{181} \end{bmatrix}\right]$$

Градиент

Градиент скалярной функции многих переменных f(x,y,z)

$$\operatorname{grad} f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

∇

VectorCalculus:

- **Gradient(f, c), Del(f,c), Nabla(f,c)** – градиент функции многих переменных **f**, **c** – (необязательный) список переменных или координат
- **linalg: grad(f,[x,y,z], c), c:** coords=cylindrical, coords=spherical

```
> with(VectorCalculus):
> u := arctan(y/x): g := Gradient(u, [x, y])
```

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

```
> simplify(g)
```

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

```
> Gradient(r^2, 'polar'[r, θ])
```

$$2r\bar{e}_r$$

```
> SetCoordinates('spherical'[r, ϕ, θ])
```

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

```
> g3 := ∇(r^2 ϕ)
```

$$g3 := 2r\phi\bar{e}_r + (r)\bar{e}_\phi$$

Дивергенция и ротор

Дивергенция вектор-функции **F**(x,y,z)

$$\mathrm{div}\boldsymbol{F}(x,y,z)=\frac{\partial F_x}{\partial x}+\frac{\partial F_y}{\partial y}+\frac{\partial F_z}{\partial z}$$

VectorCalculus:

- **Divergence(F)** – дивергенция векторного поля F
- **linalg: diverge(F,[x,y,z],c), c:** coords=cylindrical, coords=spherical

Ротор вектор-функции **F**(x,y,z)

$$\mathrm{rot}\boldsymbol{F}=\left[\left(\frac{\partial F_z}{\partial y}-\frac{\partial F_y}{\partial z}\right),\left(\frac{\partial F_x}{\partial z}-\frac{\partial F_z}{\partial x}\right),\left(\frac{\partial F_y}{\partial x}-\frac{\partial F_x}{\partial y}\right)\right]$$

VectorCalculus:

- **Curl(F)** – ротор векторного поля F в 3D
- **linalg: curl(F,[x,y,z],c), c:** coords=cylindrical, coords=spherical

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

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

```
> divF := Divergence(F)
```

$$divF := 6\,x\,y\,z$$

```
> curlF := Curl(F)
```

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

Лапласиан

Лапласиан скалярной функции f(x,y,z) $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$

VectorCalculus:

- **Laplacian(f, c)** – лапласиан функции многих переменных
- **Laplacian(F)** – лапласиан векторного поля

  c – (необязательный) список переменных или координат
- **linalg: laplacian(f,[x,y,z],c), c:** coords=cylindrical, coords=spherical

```
> with(VectorCalculus):
> SetCoordinates(cartesian[x,y]):
> v := VectorField(<x, y/x>);
```

$$v := (x)\bar{e}_x + \left(\frac{y}{x}\right)\bar{e}_y$$

```
> Laplacian(v)
```

$$\frac{2y}{x^3}\bar{e}_y$$

```
> restart, with(VectorCalculus):
> u := x^3 + a·x·y^2:
> DeltaU := Laplacian(u, [x, y]);
```

$$DeltaU := 6x + 2ax$$

Матрица Якоби и якобиан

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

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

VectorCalculus:

- **Jacobian(f, v, det); Jacobian(f, v=p, det)** – матрица Якоби и якобиан **f**, где
- **f** - вектор-функция, вектор или векторное поле
- **v** – (необязательный) список переменных дифференцирования или координат. Запись v=p задает точку, в которой будет вычисляться матрица Якоби
- **det** – (необязательный) параметр в виде **determinant=true/false** позволяет вычислить якобиан (определитель). Запись **determinant** эквивалентна determinant=true
- **linalg: jacobian(f,[x,y,z])**

Примеры вычисления матрицы Якоби и якобиана

```
> with(VectorCalculus):
```

v1 - вектор-функция многих переменных (задана как список выражений)

```
> v1 := [x, y/x]
```

$$v1 := \left[x, \frac{y}{x}\right]$$

```
> Jacobian(v1, [x, y])
```

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

Если не указать переменные дифференцирования, будет ошибка

```
> Jacobian(v1)
Error, (in VectorCalculus:-Jacobian) unable
to determine differentiation variables
```

v2 - вектор

```
> v2 := Vector([x, y/x])
```

$$v2 := (x)\vec{e}_x + \left(\frac{y}{x}\right)\vec{e}_y$$

```
> Jacobian(v2, [x, y], determinant)
```

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

v3 - векторное поле

```
> SetCoordinates(cartesian[x, y]):
> v3 := VectorField(<x, y/x>)
```

$$v3 := (x)\vec{e}_x + \left(\frac{y}{x}\right)\vec{e}_y$$

Можно не указывать переменные дифференцирования, так как задана система координат

```
> Jacobian(v3)
```

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

Матрица Якоби в полярных координатах

```
> Jacobian(Vector([r^2, r t], 'coordinates' = 'polar'[r, t]))
```

$$\begin{bmatrix} 2r & 0 \\ t & r \end{bmatrix}$$

Матрица Якоби и якобиан в точке

```
> Jacobian(<x^2 + y, 2 y>, [x, y] = [-1, 1], determinant)
```

$$\begin{bmatrix} -2 & 1 \\ 0 & 2 \end{bmatrix}, -4$$