```
> restart :
```

LinearAlgebra

```
> # 110 команд содержится в LinearAlgebra
> ?Basis;
> with(LinearAlgebra);
```

[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CARE, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, CompressedSparseForm, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, FromCompressedSparseForm, FromSplitForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct, LA_Main, LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, ProjectionMatrix, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip] **(1.1)**

```
> 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, **(1.2)** *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*]

Matrix

> # ***Задание матриц***

> # ***Матрица из нулей***. *Команду записывать с большой буквы*

> $a1 := Matrix(2);$

$$a1 := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (2.1)$$

> $about(a1);$

```
Matrix(2, 2, [[0,0],[0,0]]):
  nothing known about this object
```

> $a2 := Matrix(2, 1);$

$$a2 := \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.2)$$

> $a3 := Matrix(1, 2);$

$$a3 := \begin{bmatrix} 0 & 0 \end{bmatrix} \quad (2.3)$$

> # ***Рандомная матрица***

> $a4 := RandomMatrix(2, 2); a5 := RandomMatrix(2, 2);$

$$a4 := \begin{bmatrix} -50 & 45 \\ -22 & -81 \end{bmatrix}$$

$$a5 := \begin{bmatrix} 50 & -16 \\ 10 & -9 \end{bmatrix} \quad (2.4)$$

> $a6 := RandomMatrix(1, 7, generator = 1 ..10);$

$$a6 := \begin{bmatrix} 3 & 3 & 3 & 9 & 7 & 4 & 2 \end{bmatrix} \quad (2.5)$$

> $a7 := RandomMatrix(2, density = 0.5, generator = 1 ..5);$

$$a7 := \begin{bmatrix} 0 & 4 \\ 3 & 0 \end{bmatrix} \quad (2.6)$$

> # ***Диагональная матрица (см. Help)***

> $DiagonalMatrix([1, 2, 3, 5]);$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{2.7}$$

> # ***Единичная матрица***

> $a8 := Matrix(3, shape = identity); a81 := IdentityMatrix(2);$

$$a8 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$a81 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.8}$$

> ?Matrix;

> # ***Матрица, заполненная числами***

> $a9 := Matrix(2, 3, 5);$

$$a9 := \begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix} \tag{2.9}$$

> # ***Матрица, заполненная по строкам***

> $a10 := Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);$

$$a10 := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{2.10}$$

> # ***Матрица, заполненная по столбцам***

> a11 := a10^%T;

$$a11 := \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \tag{2.11}$$

> # ***Если матрица больше, то она доопределяется нулями***

> $a12 := Matrix(4, 5, a10);$

$$a12 := \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 \\ 7 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

> # ***Определение матрицы с помощью функций***

> $f := (i, j) \to x\hat{\ }i - j;$

$$f := (i, j) \to x^i - j \tag{2.13}$$

> $g := Matrix(2, 3, f);$

$$g := \begin{bmatrix} x-1 & x-2 & x-3 \\ x^2-1 & x^2-2 & x^2-3 \end{bmatrix} \tag{2.14}$$

> $subs(x = 3, g);$

$$\begin{bmatrix} 2 & 1 & 0 \\ 8 & 7 & 6 \end{bmatrix} \tag{2.15}$$

> ***# Задание симметричной матрицы***

> *# shape=symmetric*

> *# shape=triangular*

> *# readonly=true*

>

 # Константная матрица

> $ConstantMatrix(4, 2, 3);$

$$\begin{bmatrix} 4 & 4 & 4 \\ 4 & 4 & 4 \end{bmatrix} \tag{2.16}$$

> $C := ConstantMatrix(n, 2, shape = triangular[upper]);\ C[1, 1] := x^2;\ C;$

$$C := \begin{bmatrix} n & n \\ 0 & n \end{bmatrix}$$

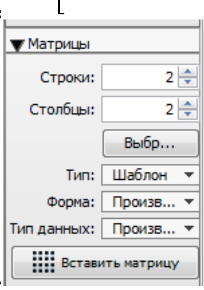
$$C_{1,1} := x^2$$

$$\begin{bmatrix} x^2 & n \\ 0 & n \end{bmatrix} \tag{2.17}$$

> $?Matrix;$

> $Matrix(3, 3, [[1, 0, 0], [0, 1, 0], [0, 0, 1]]);$

> $\begin{bmatrix} 77. & 0. & 0. & 0. \\ 0. & 95. & 0. & 0. \\ 0. & 0. & -89. & 0. \end{bmatrix}$



Большие матрицы

```
> with(LinearAlgebra) :
> RandomMatrix(150);
```

$$\begin{bmatrix} -3 & 94 & -57 & -64 & -92 & -19 & -82 & -43 & -88 & -5 & \dots \\ -41 & 74 & 15 & -29 & 19 & 20 & 51 & 21 & -88 & -74 & \dots \\ 81 & 2 & -88 & 87 & 87 & -75 & 57 & 11 & -85 & 24 & \dots \\ -6 & -48 & 49 & 36 & 37 & 50 & 67 & -74 & -54 & 96 & \dots \\ -90 & -3 & -45 & 96 & 37 & 48 & -82 & -57 & -92 & 43 & \dots \\ 13 & 67 & 19 & 43 & 80 & -27 & -63 & 38 & -50 & 31 & \dots \\ 5 & -94 & -43 & -46 & 5 & -70 & 0 & -28 & -64 & 22 & \dots \\ 42 & 13 & -54 & -92 & -79 & 78 & -32 & -42 & 84 & 57 & \dots \\ -61 & 88 & 70 & 15 & -2 & -68 & 59 & 47 & -22 & -86 & \dots \\ 50 & 63 & 86 & 81 & -86 & -18 & 10 & -1 & -87 & -93 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.18}$$

150 × 150 Matrix

A r r a y

```
> ?Array;
```

```
> # Задание векторов
> b1 := Vector([1, 2, 3]);
```

$$b1 := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{3.1}$$

```
> b2 := Vector[row]([1 , 2, 3]);
```

$$b2 := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \tag{3.2}$$

```
> ?Vector
> with(VectorCalculus) :
```

```
> # Выделение строк-столбцов из матрицы
```

```
> M := Matrix([[1, 1, 1], [2, 2, 2], [3, 7, 4]]);
```

$$M := \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 7 & 4 \end{bmatrix} \tag{4.1}$$

> $Row(M, 2..3);$

$$\begin{bmatrix} 2 & 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 4 \end{bmatrix} \tag{4.2}$$

> $Column(M, 1..2);$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix} \tag{4.3}$$

>

 # Удаление строк-столбцов

> $DeleteRow(M, 1);$
> $DeleteColumn(M, 1);$

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

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 7 & 4 \end{bmatrix} \tag{4.4}$$

>

 # Размерности матриц и векторов

> #?Dimension:

> $restart:$

> $with(LinearAlgebra):$

>

> $V := \langle x, y, z, w \rangle$

$$V := \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{4.5}$$

> $Dimension(V)$

$$4 \tag{4.6}$$

> $A := IdentityMatrix(3, 5)$

$$A := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.7}$$

> $rowdim := RowDimension(A)$

$$rowdim := 3 \tag{4.8}$$

> $coldim := ColumnDimension(A)$

$$coldim := 5 \tag{4.9}$$

> $m, n := Dimension(A)$

$$m, n := 3, 5 \tag{4.10}$$

>

 # *Операции со столбцами* и *строками матрицы*

> # ***RowOperation(A, [ri,rj],expr)***

– изменение строки $ri : ri := ri + rj * expr$,
где expr – число или выражение$\cdot$ (*аналог addrow*), *сложение строк*

> # ***RowOperation(A, r,expr)***

– умножение строки r на выражение expr : $r := r * expr$
(*аналог mulrow*)

> # ***ColumnOperation(A, [ci,cj],expr)***

– изменение столбца $ci : ci := ci + cj * expr$,
где expr – число или выражение$\cdot$ (*аналог addcol*), *сложение столбцов*

> # ***ColumnOperation(A, c,expr)***

– умножение столбца c на выражение expr :
$c := c * expr\cdot$ (*аналог mulcol*)

> $with(LinearAlgebra) :$

> $A := \langle\langle 1, 2, 3\rangle|\langle 4, 5, 6\rangle|\langle 7, 8, 9\rangle\rangle$

$$A := \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \tag{4.11}$$

> *#Умножение*
$RowOperation(A, 3, 3)$

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 9 & 18 & 27 \end{bmatrix} \tag{4.12}$$

> $ColumnOperation(A, [1, 3], inplace = true)$
 # inplace=true - указывает, перезаписывает ли вывод входные данные

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix} \tag{4.13}$$

> *# перестановка строк*
$RowOperation(A, [1, 3]);$

$$\begin{bmatrix} 9 & 6 & 3 \\ 8 & 5 & 2 \\ 7 & 4 & 1 \end{bmatrix} \tag{4.14}$$

> *# **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*

> *?addrow; #depricated*

> *# **Выделение подматрицы·(подвектора)***

> *# SubMatrix(A, r, c, outopts) – выделение подматрицы из матрицы A, r – диапазон (номера) строк, с - 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, out задает тип результате* в *виде output = matrix или /* и *output = determinant·(определитель), meth – метод вычисления определителя* в *виде method = value*

> *with(LinearAlgebra) :*

> *A := Matrix(3, 4, [[1, 2, 3, 4], [5, 6, 7, 8], [9, 0, 1, 2]]);*

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 0 & 1 & 2 \end{bmatrix} \qquad \textbf{(4.15)}$$

> *SubMatrix(A, [1, 2], [2, 3])*

$$\begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \qquad \textbf{(4.16)}$$

> *SubMatrix(A, [1, 3], [2, 3])*

$$\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} \qquad \textbf{(4.17)}$$

> $A := \langle \langle a|b|c \rangle, \langle d|e|f \rangle, \langle g|h|i \rangle \rangle;$

$$A := \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (4.18)$$

> $Minor(A, 3, 3);$

$$a e - b d \quad (4.19)$$

> $Minor(A, 1, 2);$

$$d i - f g \quad (4.20)$$

> **# *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*

> **# *Арифметические операции с матрицами***

> **# *Умножение матриц***

> # *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) :*

> *M1 := RandomMatrix(2, 2, generator = 1 ..9); M2 := RandomMatrix(2, 2, generator = 1 ..9);*

$$M1 := \begin{bmatrix} 4 & 7 \\ 1 & 1 \end{bmatrix}$$

$$M2 := \begin{bmatrix} 9 & 9 \\ 4 & 2 \end{bmatrix} \tag{5.1}$$

> $M1.M2;$

$$\begin{bmatrix} 64 & 50 \\ 13 & 11 \end{bmatrix} \tag{5.2}$$

> $evalm(M1\&{*}M2);$

$$\begin{bmatrix} 64 & 50 \\ 13 & 11 \end{bmatrix} \tag{5.3}$$

> $M1.M2 - M2.M1;$

$$\begin{bmatrix} 19 & -22 \\ -5 & -19 \end{bmatrix} \tag{5.4}$$

> $3 \cdot M1;$

$$\begin{bmatrix} 12 & 21 \\ 3 & 3 \end{bmatrix} \tag{5.5}$$

> **# Линейная комбинация**

> $restart:$
> $with(LinearAlgebra):$
> $M := [[a, d], [b, e], [c, f]];$
> $about(M);$
> $M := convert(M, Matrix);$
> $about(M);$

$$M := [[a, d], [b, e], [c, f]]$$

```
[[a, d], [b, e], [c, f]]:
  nothing known about this object
```

$$M := \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

```
Matrix(3, 2, [[a,d],[b,e],[c,f]]):
  nothing known about this object
```

> $MatrixAdd(M, IdentityMatrix(3, 2));$

$$\begin{bmatrix} a+1 & d \\ b & e+1 \\ c & f \end{bmatrix} \quad (6.1)$$

> $M + 3 \cdot IdentityMatrix(3, 2);$

$$\begin{bmatrix} a+3 & d \\ b & e+3 \\ c & f \end{bmatrix} \quad (6.2)$$

> # ***Обратная матрица***

> $restart : with(LinearAlgebra) :$
$Y := Matrix([[a, b], [c, d]]);$
$Y1 := MatrixInverse(Y);$
$Equal(\ Y1.Y, Y.(Y)^{(-1)});$
$simplify\left(\frac{d\,a}{a\,d - b\,c} - \frac{b\,c}{a\,d - b\,c}\right);$

$$Y := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$Y1 := \begin{bmatrix} \frac{d}{a\,d - b\,c} & -\frac{b}{a\,d - b\,c} \\ -\frac{c}{a\,d - b\,c} & \frac{a}{a\,d - b\,c} \end{bmatrix}$$

$$true$$

$$1 \quad (7.1)$$

> # ***Ортогональная матрица***

> $Q := Matrix([[0.96, -0.28], [0.28, 0.96]]);$

$$Q := \begin{bmatrix} 0.96 & -0.28 \\ 0.28 & 0.96 \end{bmatrix} \quad (8.1)$$

> $IsOrthogonal(Q);$

$$true \quad (8.2)$$

> $evalm(Q^{-1} - Q^{\%T});$

$$\begin{bmatrix} 1.11022302462516 \times 10^{-16} & 5.55111512312578 \times 10^{-17} \\ -5.55111512312578 \times 10^{-17} & 0. \end{bmatrix} \quad (8.3)$$

> $Q \cdot Q^{\%T} = IdentityMatrix(2);$

$$\begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (8.4)$$

> ***# Положительно определенная матрица***

> *# IsDefinite(A, q) – проверяет положительную/отрицательную определенность матрицы A*

> *# Матрица M является положительно определённой, если все собственные значения положительны*

> *# Матрица M является положительно полу−определённой, если все собственные значения больше или равны нулю.*

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

> $IsDefinite(A);$ *# положительно определенная*

$$true \qquad (9.1)$$

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

$$true \qquad (9.2)$$

> $evalf(Eigenvalues(A));$

$$\begin{bmatrix} 5.645751311 \\ 0.354248689 \end{bmatrix} \qquad (9.3)$$

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

$$true \qquad (9.4)$$

> $IsDefinite(A, 'query' = 'negative\_semidefinite');$

$$false \qquad (9.5)$$

> $IsDefinite(A, 'query' = 'negative\_definite');$

$$false \qquad (9.6)$$

>

> ***# Транспонированная матрица***

> $Q.Transpose(Q);$
$Q.Q^{\%T};$
$A;$
$A^{\%T};$

$$Q^2$$

$$Q^2$$

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

$$\begin{bmatrix} 2 & 3 \\ 2 & 4 \end{bmatrix} \quad \textbf{(10.1)}$$

> # Эрми́тово-сопряжённая ма́трица или сопряжённо-транспони́рованная ма́трица — это матрица с комплексными элементами, полученная из исходной матрицы транспонированием и заменой каждого элемента комплексно-сопряжённым ему.

> $M := Matrix([[1 + 2 \cdot I, 3 - I], [0, I]]);$

$$M := \begin{bmatrix} 1+2I & 3-I \\ 0 & I \end{bmatrix} \quad \textbf{(10.2)}$$

> $HermitianTranspose(M);$

$$\begin{bmatrix} 1-2I & 0 \\ 3+I & -I \end{bmatrix} \quad \textbf{(10.3)}$$

> # ***IsUnitary(A)*** *– проверяет, является ли комплексная матрица A унитарной:*

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

$$Q := \begin{bmatrix} \frac{3\sqrt{10}}{10} & \frac{I}{10}\sqrt{10} \\ -\frac{\sqrt{10}}{10} & \frac{3\,I}{10}\sqrt{10} \end{bmatrix} \quad \textbf{(10.4)}$$

> $IsUnitary(Q);$

$$true \quad \textbf{(10.5)}$$

> $HermitianTranspose(Q) \cdot Q = Q.HermitianTranspose(Q);$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \textbf{(10.6)}$$

> $Q.HermitianTranspose(Q) - Q.HermitianTranspose(Q);$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \textbf{(10.7)}$$

> # ***Норма матрицы и вектора***

> $restart:$

```
with(LinearAlgebra) :
v := Vector([a, b, c]);
VectorNorm(v, 1);
M := Matrix([[a, b], [c, d]]) :
MatrixNorm(M, 1);
```

$$v := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$|a| + |b| + |c|$$

$$\max(|a| + |c|, |b| + |d|) \tag{11.1}$$

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

$$\sqrt{|a|^2 + |b|^2 + |c|^2} \tag{11.2}$$

```
> MatrixNorm(M, 2);
```

$$\left(\max\left(\left|RootOf\left(\_Z^2 + \left(-c\,\bar{c} - d\,\bar{d} - a\,\bar{a} - b\,\bar{b}\right)\_Z + \bar{a}\,\bar{d}\,a\,d - \bar{a}\,\bar{d}\,b\,c - \bar{b}\,\bar{c}\,a\,d + \bar{b}\,\bar{c}\,b\,c,\ index = 1\right)\right|, \left|RootOf\left(\_Z^2 + \left(-c\,\bar{c} - d\,\bar{d} - a\,\bar{a} - b\,\bar{b}\right)\_Z + \bar{a}\,\bar{d}\,a\,d - \bar{a}\,\bar{d}\,b\,c - \bar{b}\,\bar{c}\,a\,d + \bar{b}\,\bar{c}\,b\,c,\ index = 2\right)\right|\right)\right)^{1/2} \tag{11.3}$$

```
> # Проверка равенства двух матриц Equal(A,B)
> A := RandomMatrix(3) :
> A1 := MatrixInverse(A) :
> A · A1 = IdentityMatrix(3);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11.4}$$

```
> Equal(A · A1, IdentityMatrix(3)) ;
```

$$true \tag{11.5}$$

```
> # MatrixNorm(A-B,1) или MatrixNorm(A-B,infinity) – проверка равенства
  матриц по норме, для приближенных значений
```

```
> A := RandomMatrix(3, density = 0.75, generator = 0..0.5);
```

$$A := \begin{bmatrix} 0.00826029311923809351 & 0.0700719200506078055 & 0.196794925297690770 \\ 0. & 0.458558956698768283 & 0.377894946583917257 \\ 0.474462543989446373 & 0. & 0. \end{bmatrix} \tag{11.6}$$

```
> AA := A · A^(-1) :
```

```
> MatrixNorm(AA - IdentityMatrix(3), 1);
```
$$6.66133814775093924 \times 10^{-16} \quad (11.7)$$

```
> MatrixNorm(AA - IdentityMatrix(3), infinity);
```
$$4.44089209850062616 \times 10^{-16} \quad (11.8)$$

```
> # Ранг матрицы. Размерность. Количество строк/столбцов
> A := ScalarMatrix(n, 3);
  k1, k2 := Dimension(A);
  ColumnDimension(A);
  RowDimension(A);
```
$$A := \begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & n \end{bmatrix}$$
$$k1, k2 := 3, 3$$
$$3$$
$$3 \quad (12.1)$$

```
> Rank(A);
```
$$3 \quad (12.2)$$

```
> # Определитель матрицы
> Determinant(A);
```
$$n^3 \quad (13.1)$$

```
> # Минор матрицы
> Minor(A, 1, 1);
```
$$n^2 \quad (14.1)$$

```
> # Операция map
> M3 := RandomMatrix(2, 2, generator = 1..9);
```
$$M3 := \begin{bmatrix} 9 & 2 \\ 5 & 4 \end{bmatrix} \quad (15.1)$$

> $map(x \to x^2, M3);$

$$\begin{bmatrix} 81 & 4 \\ 25 & 16 \end{bmatrix} \tag{15.2}$$

> # ***Операция zip***

> $p1 := Matrix([[1, 2], [3, 4]]); p2 := Matrix([[10, 20], [30, 40]]);$

$$p1 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$p2 := \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \tag{16.1}$$

> $zip((x, y) \to x + y, p1, p2);$

$$\begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix} \tag{16.2}$$

> # ***Решение систем линейных уравнений***

> # ***LinearSolve(A,B)*** *– решение уравнения $AX = B$,*
> *где B – матрица или вектор правой части,*
> *X – матрица или вектор неизвестных.*

> #
> ***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 '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*

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

> *# LinearAlgebra[GenerateMatrix] - generate the coefficient Matrix from equations*

> $restart:$
$with(LinearAlgebra):$
$s := [2 \cdot x + 3 \cdot y = 8, 10 \cdot x - 3 \cdot y = 4];$
$v := [x, y]:$
$A, b := GenerateMatrix(s, [x, y]);$
$Determinant(A);$

$$s := [2x + 3y = 8, 10x - 3y = 4]$$

$$A, b := \begin{bmatrix} 2 & 3 \\ 10 & -3 \end{bmatrix}, \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

$$-36 \qquad \textbf{(17.1)}$$

> *# Система, которую нужно решить*

> $A.Vector(v) = b;$

$$\begin{bmatrix} 2x + 3y \\ 10x - 3y \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix} \qquad \textbf{(17.2)}$$

> ***# Решение с помощью LinearSolve***

> $r := LinearSolve(A, b);$

$$r := \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \textbf{(17.3)}$$

> $restart : with(LinearAlgebra):$

> $A := Matrix(3, 4, [[1, 2, 1, -1], [0, 1, 0, -1], [0, 0, 0, -3]]);$

$$A := \begin{bmatrix} 1 & 2 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & -3 \end{bmatrix} \qquad \textbf{(17.4)}$$

> $b := Vector([2, -1, -9]);$

$$b := \begin{bmatrix} 2 \\ -1 \\ -9 \end{bmatrix} \qquad \textbf{(17.5)}$$

> $LinearSolve(A, b, method = 'subs', free = 's');$

(17.6)

$$\begin{bmatrix} 1 - s_1 \\ 2 \\ s_1 \\ 3 \end{bmatrix} \quad \textbf{(17.6)}$$

```
> # Метод обратной матрицы
  Решение с помощью умножения на обратную матрицу
```

```
> x := A^(-1).b;
```

$$x := \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \textbf{(17.7)}$$

```
> # Решение методом Крамера. Определитель отличен от нуля.
```

```
> # Пример решения для матрицы 2 порядка
  Delta := Determinant(A);
  with(linalg) :
  A1 := convert(concat(b, DeleteColumn(A, 1)), Matrix); Delta1 := Determinant(A1);
  A2 := convert(concat(DeleteColumn(A, 2), b), Matrix); Δ2 := Determinant(A2);
  x1 := Delta1/Delta; x2 := Δ2/Delta;
```

$$\Delta := -36$$

$$A1 := \begin{bmatrix} 8 & 3 \\ 4 & -3 \end{bmatrix}$$

$$\Delta 1 := -36$$

$$A2 := \begin{bmatrix} 2 & 8 \\ 10 & 4 \end{bmatrix}$$

$$\Delta 2 := -72$$

$$x1 := 1$$

$$x2 := 2 \quad \textbf{(17.8)}$$

```
> # Нахождение решения, когда матрица треугольная
```

*LinearAlgebra[**BackwardSubstitute**] - solve A . X = B where A is in upper row echelon form*

```
> A1 := Matrix([[1, 2], [0, 3]]);
  b := Vector([3, 9]);
  x := BackwardSubstitute(A1, b);
  evalm(A1.x - b);
```

$$A1 := \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$$

$$b := \begin{bmatrix} 3 \\ 9 \end{bmatrix}$$

$$x := \begin{bmatrix} -3 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \end{bmatrix} \tag{17.9}$$

*LinearAlgebra[**ForwardSubstitute**] - solve A . X = B where A is in lower row echelon form*

```
> A2 := Matrix([[3, 0], [1, 2]]);
  b := Vector([3, 9]);
  x := ForwardSubstitute(A2, b);
```

$$A2 := \begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}$$

$$b := \begin{bmatrix} 3 \\ 9 \end{bmatrix}$$

$$x := \begin{bmatrix} 1 \\ 4 \end{bmatrix} \tag{17.10}$$

> ***# Решение методом Гаусса***

> ***# (метод последовательного исключения неизвестных)***

> *# Если столбец свободных членов равен нулю, то система линейных алгебраических уравнений называется однородной, в противном случае – неоднородной.*

> *# Если существует хотя бы одно решение системы линейных алгебраических уравнений, то она называется совместной, в противном случае – несовместной.*

> *# Если СЛАУ имеет единственное решение, то она называется определенной. Если решений больше одного, то система называется неопределенной.*

> *# Если к матрице A добавить в качестве $(n+1)$-ого столбца матрицу-столбец свободных членов, то получим так называемую расширенную матрицу системы линейных уравнений.*

> *# Квадратная матрица A называется вырожденной, если ее определитель равен нулю.*

> *# Если с системой линейных алгебраических уравнений произвести **следующие действия***

> *# **1)** поменять местами два уравнения,*

> *# **2)** умножить обе части какого-либо уравнения на произвольное и отличное от нуля действительное (или комплексное) число k,*

> *# **3)** к обеим частям какого-либо уравнения прибавить соответствующие части другого уравнения, умноженные на произвольное число k,*

> *# то получится **эквивалентная система**, которая имеет такие же решения (или также как и исходная не имеет решений).*

> *# Процесс последовательного исключения неизвестных называется прямым ходом метода Гаусса.*

> *# Процесс последовательного нахождения неизвестных переменных при движении от последнего уравнения к первому называется обратным ходом метода Гаусса.*

l i n a l g

> *# addcol — добавляет к одному из столбцов другой столбец, умноженный на некоторое число;*

> *# addrow — добавляет к одной из строк другую строку, умноженную на некоторое число;*

> *# angle — вычисляет угол между векторами;*

> *# augment — объединяет две или больше матриц по горизонтали;*

> *# backsub — реализует метод обратной подстановки при решении системы линейных уравнений (см. также forwardsub);*

> *# band — создает ленточную матрицу;*

> *# basis — находит базис векторного пространства;*

> *# bezout — создает Bezout-матрицу двух полиномов;*

> *# BlockDiagonal — создает блок-диагональную матрицу;*

> *# blockmatrix — создает блок-матрицу;*

> *# cholesky — декомпозиция Холесского для квадратной положительно определенной матрицы;*

> *# charmat — создает характеристическую матрицу (charmat(M,v) — матрица, вычисляемая как v E-M);*

> *# charpoly — возвращает характеристический полином матрицы;*

> *# colspace — вычисляет базис пространства столбцов;*

> *# colspan — находит базис линейной оболочки столбцов матрицы;*

> *# companion — вычисляет сопровождающую матрицу, ассоциированную с полиномом;*

> *# cond — вычисляет число обусловленности матрицы (cond(M) есть величина norm(M) norm(M-1);*

> *# curl — вычисляет ротор вектора;*

> *# definite — тест на положительную (отрицательную) определенность матрицы*

> *# diag — создает блок-диагональную матрицу;*

> *# diverge — вычисляет дивергенцию векторной функции;*

> # eigenvals — вычисляет собственные значения матрицы;

> # eigenvects — вычисляет собственные векторы матрицы;

> # equal — определяет, являются ли две матрицы равными;

> # exponential — создает экспоненциальную матрицу;

> # ffgausselim — свободное от дробей Гауссово исключение в матрице;

> # fibonacci — матрица Фибоначчи;

> # forwardsub — реализует метод прямой подстановки при решении системы линейных уравнений (например, для матрицы L и вектора b

> # forwardsub(L, b) возвращает вектор решения x системы линейных уравнений L-x=b);

> # frobenius — вычисляет форму Фробениуса (Frobenius) матрицы;

> # gausselim — Гауссово исключение в матрице;

> # gaussjord — синоним для rref (метод исключения Гаусса—Жордана);

> # geneqns — генерирует элементы матрицы из уравнений;

> # genmatrix — генерирует матрицу из коэффициентов уравнений;

> # grad — градиент векторного выражения;

> # GramSchmidt — вычисляет ортогональные векторы;

> # hadamard — вычисляет ограничение на коэффициенты детерминанта;

> # hessian — вычисляет гессиан-матрицу выражения;

> # hilbert — создает матрицу Гильберта;

> # htranspose — находит эрмитову транспонированную матрицу;

> # ihermite — целочисленная эрмитова нормальная форма;

> # indexfunc — определяет функцию индексации массива;

> # Innerprod — вычисляет векторное произведение;

> # Intbasis — определяет базис пересечения пространств;

> # ismith — целочисленная нормальная форма Шмитта;

> # iszero — проверяет, является ли матрица ноль-матрицей;

> # jacobian —' вычисляет якобиан векторной функции;

> # JordanBlock — возвращает блок-матрицу Жордана;

> # kernel — находит базис ядра преобразования, соответствующего данной матрице;

> # laplacian — вычисляет лапласиан;

> # leastsqrs — решение уравнений по методу наименьших квадратов;

> # linsolve — решение линейных уравнений;

> # LudeComp — осуществляет LU-разложение;

> # minpoly — вычисляет минимальный полином матрицы;

> # mulcol — умножает столбец матрицы на заданное выражение;

> # mulrow — умножает строку матрицы на заданное выражение;

> # multiply — перемножение ‘матриц или матрицы и вектора;

> # normalize — нормализация вектора;

> # orthog — тест на ортогональность матрицы;

> # permanent — вычисляет перманент матрицы — определитель, вычисляемый без перестановок;

> # pivot — вращение относительно элементов матрицы;

> # potential — вычисляет потенциал векторного поля;

> # Qrdecomp — осуществляет QR-разложение;
> # randmatrix — генерирует случайные матрицы;
> # randvector — генерирует случайные векторы;
> # ratform — вычисляет рациональную каноническую форму;
> # references — выводит список основополагающих работ по линейной алгебре;
> # rowspace — вычисляет базис пространства строки;
> # rowspan — вычисляет векторы охвата для места столбца;
> # rref — реализует преобразование Гаусса-Жордана матрицы;
> # scalarmul — умножение матрицы или вектора на заданное выражение;
> # singval — вычисляет сингулярное значение квадратной матрицы;
> # singularvals — возвращает список сингулярных значений квадратной матрицы;
> # smith — вычисляет Шмиттову нормальную форму матрицы;
> # submatrix — извлекает указанную подматрицу из матрицы;
> # subvector — извлекает указанный вектор из матрицы;
> # sumbasis — определяет базис объединения системы векторов;
> # swapcol — меняет местами два столбца в матрице;
> # swaprow — меняет местами две строки в матрице;
> # sylvester — создает матрицу Сильвестра из двух полиномов;
> # toeplitz — создает матрицу Теплица;
> # trace — возвращает след матрицы;
> # vandermonde — создает вандермондову матрицу;
> # vecpotent — вычисляет векторный потенциал;
> # vectdim — определяет размерность вектора;
> # wronskian — вронскиан векторных функций.
> # http://wiselab.ru/category/maple-15/page/5/

> # **Собственные векторы и собственные значения матриц**
> # **Определение:**
> # ненулевой вектор, который при умножении на некоторую квадратную матрицу превращается в самого же себя с числовым коэффициентом, называется собственным вектором матрицы.
> # Число L называют собственным значением или собственным числом данной матрицы. Au=Lu

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

> # **Eigenvalues(A)** – возвращает собственные значения матрицы A в виде вектор-столбца.

```
> A := RandomMatrix(2, generator = 0 ..5);
```

(19.1)

$$A := \begin{bmatrix} 4 & 0 \\ 3 & 0 \end{bmatrix} \quad \textbf{(19.1)}$$

> $Eigenvalues(A);$

$$\begin{bmatrix} 0 \\ 4 \end{bmatrix} \quad \textbf{(19.2)}$$

> $Eigenvalues(A, output = 'Vector[row]');$

$$\begin{bmatrix} 0 & 4 \end{bmatrix} \quad \textbf{(19.3)}$$

> $Eigenvalues(A, output = 'list');$

$$[0, 4] \quad \textbf{(19.4)}$$

> $A := RandomMatrix(5, generator = 0..5);$

$$A := \begin{bmatrix} 2 & 3 & 0 & 4 & 4 \\ 3 & 4 & 5 & 1 & 2 \\ 2 & 2 & 2 & 3 & 3 \\ 2 & 5 & 3 & 2 & 0 \\ 1 & 3 & 0 & 2 & 3 \end{bmatrix} \quad \textbf{(19.5)}$$

> $evalf(Eigenvalues(A, output = 'list'));$

$$[0.3372679497, 1.539840685, 12.43447496, -0.6557917979 + 3.079054427\,I, -0.6557917979 - 3.079054427\,I] \quad \textbf{(19.6)}$$

> $Eigenvalues(A, implicit, output = 'list');$

$$[RootOf(\_Z^5 - 13\_Z^4 + 15\_Z^3 - 117\_Z^2 + 228\_Z - 64, index = 1), RootOf(\_Z^5 - 13\_Z^4 + 15\_Z^3 - 117\_Z^2 + 228\_Z - 64, index = 2), RootOf(\_Z^5 - 13\_Z^4 + 15\_Z^3 - 117\_Z^2 + 228\_Z - 64, index = 3), RootOf(\_Z^5 - 13\_Z^4 + 15\_Z^3 - 117\_Z^2 + 228\_Z - 64, index = 4), RootOf(\_Z^5 - 13\_Z^4 + 15\_Z^3 - 117\_Z^2 + 228\_Z - 64, index = 5)] \quad \textbf{(19.7)}$$

>

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

> $A := Matrix([[2, 2], [3, 1]]);$

$$A := \begin{bmatrix} 2 & 2 \\ 3 & 1 \end{bmatrix} \quad \textbf{(19.8)}$$

> $e, E := Eigenvectors(A);$

$$e, E := \begin{bmatrix} -1 \\ 4 \end{bmatrix}, \begin{bmatrix} -\frac{2}{3} & 1 \\ 1 & 1 \end{bmatrix} \quad \textbf{(19.9)}$$

> *E[1]; E[2]; e[1]; e[2];*

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

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

$$-1$$

$$4 \qquad \textbf{(19.10)}$$

> *Eigenvectors(A, output = vectors);*

$$\begin{bmatrix} 1 & -\frac{2}{3} \\ 1 & 1 \end{bmatrix} \qquad \textbf{(19.11)}$$

> *# [число, кратность числа]*
Eigenvectors(A, output = vectors, output = list);

$$\left[\left[-1, 1, \left\{\begin{bmatrix} -\frac{2}{3} \\ 1 \end{bmatrix}\right\}\right], \left[4, 1, \left\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}\right]\right] \qquad \textbf{(19.12)}$$

> *MatrixVectorMultiply(A, Transpose(E[1])) = e[1]·Transpose(E[1]);*

$$\begin{bmatrix} \frac{2}{3} \\ -1 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ -1 \end{bmatrix} \qquad \textbf{(19.13)}$$

> *MatrixVectorMultiply(A, Transpose(E[2])) = e[2]·Transpose(E[2]);*

$$\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \qquad \textbf{(19.14)}$$

> ***# Характеристический многочлен и характеристическая матрица***

> *fl := CharacteristicPolynomial(A, lambda);*

$$fl := \lambda^2 - 3\lambda - 4 \qquad \textbf{(19.15)}$$

> *MinimalPolynomial(A, lambda);*

$$\lambda^2 - 3\lambda - 4 \qquad \textbf{(19.16)}$$

> *L := solve(fl, lambda);*

$$L := 4, -1 \qquad \textbf{(19.17)}$$

> *Eigenvalues(A, output = list);*

$$[4, -1] \qquad \textbf{(19.18)}$$

> *ch := CharacteristicMatrix(A, lambda);*

$$ch := \begin{bmatrix} \lambda - 2 & -2 \\ -3 & \lambda - 1 \end{bmatrix} \qquad \textbf{(19.19)}$$

> *fd := Determinant(ch);*

$$fd := \lambda^2 - 3\lambda - 4 \qquad \textbf{(19.20)}$$

> $A := Matrix([[-13, -10], [21, 16]]);$

$$A := \begin{bmatrix} -13 & -10 \\ 21 & 16 \end{bmatrix} \qquad \textbf{(19.21)}$$

> **# Факторизация матриц: QR-разложение**

> **# QRDecomposition(A) – QR-разложение матрицы :**

> # A=QR,
> # где Q – ортогональная матрица, R – верхнетреугольная матрица.

> $A := Matrix\left(3, 3, \left[[2, 4, -1], \left[3, 6, -\frac{3}{2}\right], [1, -1, -2]\right]\right);$

$$A := \begin{bmatrix} 2 & 4 & -1 \\ 3 & 6 & -\frac{3}{2} \\ 1 & -1 & -2 \end{bmatrix} \qquad \textbf{(19.22)}$$

> $Q, R := QRDecomposition(A);$

$$Q, R := \begin{bmatrix} \frac{\sqrt{14}}{7} & \frac{\sqrt{182}}{91} \\ \frac{3\sqrt{14}}{14} & \frac{3\sqrt{182}}{182} \\ \frac{\sqrt{14}}{14} & -\frac{\sqrt{182}}{14} \end{bmatrix}, \begin{bmatrix} \sqrt{14} & \frac{25\sqrt{14}}{14} & -\frac{17\sqrt{14}}{28} \\ 0 & \frac{3\sqrt{182}}{14} & \frac{3\sqrt{182}}{28} \end{bmatrix} \qquad \textbf{(19.23)}$$

> $mQ, nQ := Dimension(Q); mR, nR := Dimension(R);$

$$mQ, nQ := 3, 2$$
$$mR, nR := 2, 3 \qquad \textbf{(19.24)}$$

> $Equal(Q \cdot R, A);$

$$true \qquad \textbf{(19.25)}$$

> $Qf, Rf := QRDecomposition(A, fullspan);$# полное разложение

$$Qf, Rf := \begin{bmatrix} \frac{\sqrt{14}}{7} & \frac{\sqrt{182}}{91} & \frac{3\sqrt{13}}{13} \\ \frac{3\sqrt{14}}{14} & \frac{3\sqrt{182}}{182} & -\frac{2\sqrt{13}}{13} \\ \frac{\sqrt{14}}{14} & -\frac{\sqrt{182}}{14} & 0 \end{bmatrix}, \begin{bmatrix} \sqrt{14} & \frac{25\sqrt{14}}{14} & -\frac{17\sqrt{14}}{28} \\ 0 & \frac{3\sqrt{182}}{14} & \frac{3\sqrt{182}}{28} \\ 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(19.26)}$$

> $Equal(Qf \cdot Rf, A);$

$$true \qquad \textbf{(19.27)}$$

> **# Факторизация матриц: LU-разложение**

> # ***LUDecomposition(A) – LU-разложение матрицы***, *такое что:*

> # *A=PLU, где L – нижнетреугольная матрица,*
> # *U – верхнетреугольная матрица,*
> # *P – матрица перестановки*
> # *(единичная матрица с переставленными строками).*

> $A := Matrix(4, 4, [[0, 1, 1, -3], [-2, 3, 1, 4], [0, 0, 0, 1], [3, 1, 0, 0]]);$

$$A := \begin{bmatrix} 0 & 1 & 1 & -3 \\ -2 & 3 & 1 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{bmatrix} \qquad (19.28)$$

> $p, l, u := LUDecomposition(A);$

$$p, l, u := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{2} & \frac{11}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19.29)$$

> $p \cdot l \cdot u;$

$$\begin{bmatrix} 0 & 1 & 1 & -3 \\ -2 & 3 & 1 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{bmatrix} \qquad (19.30)$$

> $LUDecomposition(A, output = 'U')\ ;$

$$\begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19.31)$$

> **# *Приведение матриц к специальному виду***

> # ***GaussianElimination(A)*** *– приведение матрицы к верхнетреугольной форме методом исключения Гаусса.*

> # *Эквивалент команды LUDecomposition(A, output = ['U'])*

> $A := Matrix(4, 4, [[8, 4, -5, -5], [3, -5, 8, 5], [-1, 0, 3, -4], [-5, -2, -1, -9]])$

$$A := \begin{bmatrix} 8 & 4 & -5 & -5 \\ 3 & -5 & 8 & 5 \\ -1 & 0 & 3 & -4 \\ -5 & -2 & -1 & -9 \end{bmatrix} \qquad \textbf{(19.32)}$$

> $GaussianElimination(A);$

$$\begin{bmatrix} 8 & 4 & -5 & -5 \\ 0 & -\frac{13}{2} & \frac{79}{8} & \frac{55}{8} \\ 0 & 0 & \frac{163}{52} & -\frac{213}{52} \\ 0 & 0 & 0 & -\frac{2607}{163} \end{bmatrix} \qquad \textbf{(19.33)}$$

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

$$\begin{bmatrix} 8 & 4 & -5 & -5 \\ 0 & -\frac{13}{2} & \frac{79}{8} & \frac{55}{8} \\ 0 & 0 & \frac{163}{52} & -\frac{213}{52} \\ 0 & 0 & 0 & -\frac{2607}{163} \end{bmatrix} \qquad \textbf{(19.34)}$$

> $GaussianElimination(A, 'method' = 'FractionFree');$

$$\begin{bmatrix} 8 & 4 & -5 & -5 \\ 0 & -52 & 79 & 55 \\ 0 & 0 & -163 & 213 \\ 0 & 0 & 0 & 2607 \end{bmatrix} \qquad \textbf{(19.35)}$$

> ***# Приведение матриц к специальному виду: жорданова форма***

> *# 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

> # Жорданова матрица — квадратная блочно-диагональная матрица , с блоками вида

> # $$J_\lambda = \begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda & 1 & \cdots & 0 & 0 \\ 0 & 0 & \lambda & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & \lambda & 1 \\ 0 & 0 & 0 & \cdots & 0 & \lambda \end{pmatrix}$$

> # Каждый блок называется жордановой клеткой с собственным значением lambda · (*собственные значения* в *различных блоках, вообще говоря, могут совпадать*).

> # Теорема о приведении матрицы к нормальной жордановой форме. Любая квадратная матрица подобна жордановой матрице. Две жордановы матрицы подобны тогда и только тогда, когда они составлены из одинаковых жордановых клеток и отличаются друг от друга лишь расположением клеток на главной диагонали, другими словами, любую квадратную матрицу при помощи преобразования подобия можно привести к нормальной жордановой форме и притом единственной (с точностью до перестановок жордановых клеток).

> *with*(*LinearAlgebra*) :

> *A* := *Matrix*(4, 4, [[0, −3, 1, 2], [−2, 1, −1, 2], [−2, 1, −1, 2], [−2, −3, 1, 4]]);

$$A := \begin{bmatrix} 0 & -3 & 1 & 2 \\ -2 & 1 & -1 & 2 \\ -2 & 1 & -1 & 2 \\ -2 & -3 & 1 & 4 \end{bmatrix} \qquad \textbf{(19.36)}$$

> *J* := *JordanForm*(*A*);

$$J := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad \textbf{(19.37)}$$

> *Q* := *JordanForm*(*A*, *output* = '*Q*');

(19.38)

$$Q := \begin{bmatrix} -1 & -\frac{3}{2} & 2 & 1 \\ -1 & -\frac{1}{2} & \frac{1}{2} & 0 \\ -1 & \frac{1}{2} & \frac{1}{2} & 0 \\ -1 & -\frac{3}{2} & \frac{5}{2} & 1 \end{bmatrix} \qquad (19.38)$$

> $Q^{-1} \cdot A \cdot Q$; # *проверка*

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad (19.39)$$

> *JordanBlockMatrix*([[x, 3], [5, 1], [y, 2]]);

$$\begin{bmatrix} x & 1 & 0 & 0 & 0 & 0 \\ 0 & x & 1 & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & 1 \\ 0 & 0 & 0 & 0 & 0 & y \end{bmatrix} \qquad (19.40)$$

> # ***Векторный анализ в пакетах***
LinearAlgebra* и *VectorCalculus

> *with*(*VectorCalculus*);

[&x, \`*\`, \`+\`, \`-\`, \`.\`, <,>, <|>, About, AddCoordinates, ArcLength, BasisFormat, Binormal, ConvertVector, CrossProduct, Curl, Curvature, D, Del, DirectionalDiff, Divergence, DotProduct, Flux, GetCoordinateParameters, GetCoordinates, GetNames, GetPVDescription, GetRootPoint, GetSpace, Gradient, Hessian, IsPositionVector, IsRootedVector, IsVectorField, Jacobian, Laplacian, LineInt, MapToBasis,∇, Norm, Normalize, PathInt, PlotPositionVector, PlotVector, PositionVector, PrincipalNormal, RadiusOfCurvature, RootedVector, ScalarPotential, SetCoordinateParameters, SetCoordinates, SpaceCurve, SurfaceInt, TNBFrame, TangentLine, TangentPlane, TangentVector, Torsion, Vector, VectorField, VectorPotential, VectorSpace, Wronskian, diff, eval, evalVF, int, limit, series] **(19.41)**

> # ***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

> with(VectorCalculus) :

> b1 := Vector[row]([1, 2, 3]);

$$b1 := (1)e_x + (2)e_y + (3)e_z \tag{19.42}$$

> about(b1);

```
Vector[row](3, [1,2,3], attributes = [coords = cartesian]):
  nothing known about this object
```

> s := {1 = 0, 2 = 1} :

> Vector(2, s)

$$(0)e_x + (1)e_y \tag{19.43}$$

> Vector(3, symbol = v);

$$(v_1)e_x + (v_2)e_y + (v_3)e_z \tag{19.44}$$

> $f := j \mapsto x^{j-1}$:

> Vector(3, f)

$$(1)e_x + (x)e_y + (x^2)e_z \tag{19.45}$$

> S := Vector(3, fill = 2) + Vector(3, [1, 2, 3]); whattype(S);

$$S := (3)e_x + (4)e_y + (5)e_z$$

$$Vector_{column} \tag{19.46}$$

> A := Vector([1, 2, 3], readonly = true);

$$A := (1)e_x + (2)e_y + (3)e_z \tag{19.47}$$

> VectorCalculus[CrossProduct](⟨a, b, c⟩, ⟨d, e, f⟩);

$$(bf - ce)e_x + (-af + cd)e_y + (ae - bd)e_z \tag{19.48}$$

> with(VectorCalculus) :

> $CrossProduct(\langle a, b, c\rangle, \langle d, e, f\rangle\ );$

$$(bf - c\,e)e_x + (-af + c\,d)e_y + (a\,e - b\,d)e_z \quad \textbf{(19.49)}$$

> # ***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*

> $v := VectorField(\langle x, y, z\rangle, \text{'cartesian'}[x, y, z]);$

$$v := (x)\bar{e}_x + (y)\bar{e}_y + (z)\bar{e}_z \quad \textbf{(19.50)}$$

> $attributes(v)$

$$vectorfield,\ coords = cartesian_{x, y, z} \quad \textbf{(19.51)}$$

> $About(v)$

$$\begin{bmatrix} Type: & Vector\ Field \\ Components: & [x, y, z] \\ Coordinates: & cartesian_{x, y, z} \end{bmatrix} \quad \textbf{(19.52)}$$

> $SetCoordinates(\ \text{'spherical'}[r, \phi, \theta])$

$$spherical_{r, \phi, \theta} \quad \textbf{(19.53)}$$

> $v := VectorField\left(\left\langle \frac{1}{r^2}, \sin(\phi), \cos(\theta)\right\rangle\right)$

$$v := \left(\frac{1}{r^2}\right)\bar{e}_r + (\sin(\phi))\bar{e}_\phi + (\cos(\theta))\bar{e}_\theta \quad \textbf{(19.54)}$$

> $About(v)$

$$\begin{bmatrix} Type: & Vector\ Field \\ Components: & \left[\frac{1}{r^2}, \sin(\phi), \cos(\theta)\right] \\ Coordinates: & spherical_{r, \phi, \theta} \end{bmatrix} \quad \textbf{(19.55)}$$

> $SetCoordinates(\ \text{'cylindrical'}[r, \theta, z])$

$$cylindrical_{r, \theta, z} \quad \textbf{(19.56)}$$

> $v := VectorField(\langle r\cos(\theta), \sin(\theta), z^2 \rangle)$

$$v := (r\cos(\theta))\bar{e}_r + (\sin(\theta))\bar{e}_\theta + (z^2)\bar{e}_z \quad \textbf{(19.57)}$$

> $About(v)$

$$\begin{bmatrix} \textit{Type:} & \textit{Vector Field} \\ \textit{Components:} & [r\cos(\theta), \sin(\theta), z^2] \\ \textit{Coordinates:} & \textit{cylindrical}_{r,\,\theta,\,z} \end{bmatrix} \quad \textbf{(19.58)}$$

> with(Student[VectorCalculus]);

[&x, `*`, `+`, `-`, `.`, <,>, <|>, About, ArcLength, BasisFormat, Binormal, ConvertVector, CrossProduct, Curl, Curvature, D, Del, DirectionalDiff, Divergence, DotProduct, FlowLine, Flux, GetCoordinates, GetPVDescription, GetRootPoint, GetSpace, Gradient, Hessian, IsPositionVector, IsRootedVector, IsVectorField, Jacobian, Laplacian, LineInt, MapToBasis, ∇, Norm, Normalize, PathInt, PlotPositionVector, PlotVector, PositionVector, PrincipalNormal, RadiusOfCurvature, RootedVector, ScalarPotential, SetCoordinates, SpaceCurve, SpaceCurveTutor, SurfaceInt, TNBFrame, TangentLine, TangentPlane, TangentVector, Torsion, Vector, VectorField, VectorFieldTutor, VectorPotential, VectorSpace, diff, evalVF, int, limit, series] **(19.59)**

> ?VectorFieldTutor;

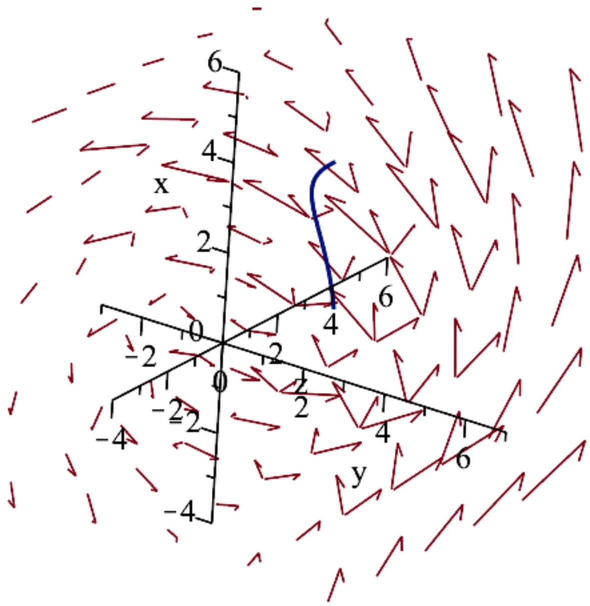
> VectorFieldTutor( );

> SetCoordinates(cartesian[x, y]) :

> $VectorFieldTutor(VectorField(\langle y, -x \rangle), \langle 1, 2 \rangle)$

> SetCoordinates(cartesian[x, y, z]) :

> $VectorFieldTutor(VectorField(\langle y, -x, z \rangle), \langle 1, 2, 1 \rangle)$



> # ***Скалярное и векторное произведение векторов***

> *# v1.v2 – скалярное умножение векторов*
> $DotProduct(v1, v2)$

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

> *restart :*

> $with(VectorCalculus) :$

> $V1 := \langle x, y, z \rangle$

$$V1 := (x)e_x + (y)e_y + (z)e_z \tag{19.60}$$

> $V2 := \langle 3, 4, 5 \rangle$

$$V2 := (3)e_x + (4)e_y + (5)e_z \tag{19.61}$$

> $DotProduct(V1, V2)$

$$3x + 4y + 5z \tag{19.62}$$

> $CrossProduct(V1, V2)$

(19.63)

$$(5y-4z)e_x+(-5x+3z)e_y+(4x-3y)e_z \quad (19.63)$$

> **# Угол между векторами, норма и нормализация вектора**

> *restart :*

> *with(LinearAlgebra) :*

> $V1 := \langle 1, 0, 1\rangle;$

$$V1 := \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad (19.64)$$

> $V2 := \langle 1, 1, 0\rangle;$

$$V2 := \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (19.65)$$

> *VectorAngle(V1, V1)*

$$0 \quad (19.66)$$

> *VectorAngle(V1, V2)*

$$\frac{\pi}{3} \quad (19.67)$$

> # ***Norm(v, p, c)*** *– p-норма вектора*

> # *VectorNorm(v, p, c) – p-норма вектора, c – (необязательные) опции для результирующего объекта*

> # *Normalize(v) – нормализация вектора*

> *?VectorNorm*

> $v := \langle a, b, c\rangle$

$$v := (a)e_x+(b)e_y+(c)e_z \quad (19.68)$$

> *VectorNorm(v, 2, conjugate = false)*

$$\sqrt{a^2+b^2+c^2} \quad (19.69)$$

> *VectorNorm(v, 2, conjugate = true)*

$$\sqrt{|a|^2+|b|^2+|c|^2} \quad (19.70)$$

> *Normalize(⟨a, b, c⟩, Euclidean, conjugate = false);*

(19.71)

$$\left[\begin{array}{c}\frac{a}{\sqrt{a^2+b^2+c^2}}\\ \frac{b}{\sqrt{a^2+b^2+c^2}}\\ \frac{c}{\sqrt{a^2+b^2+c^2}}\end{array}\right] \quad (19.71)$$

> $Normalize(\langle a, b, c\rangle, Euclidean);$

$$\left[\begin{array}{c}\frac{a}{\sqrt{|a|^2+|b|^2+|c|^2}}\\ \frac{b}{\sqrt{|a|^2+|b|^2+|c|^2}}\\ \frac{c}{\sqrt{|a|^2+|b|^2+|c|^2}}\end{array}\right] \quad (19.72)$$

> $Normalize(\langle 3, 0, 4\rangle, Euclidean)$

$$\left[\begin{array}{c}\frac{3}{5}\\ 0\\ \frac{4}{5}\end{array}\right] \quad (19.73)$$

> ***# Нахождение базиса системы векторов. Ортогонализация Грамма-Шмидта***

> *# Базисом n-мерного пространства называется любая система из n независимых векторов этого пространства.*

> # ***Basis*** –

> # ***GramSchmidt*** –

> *restart :*

> *with(LinearAlgebra) :*

> $v1 := \langle 1|0|0\rangle :$

> $v2 := \langle 0|1|0\rangle :$

> $v3 := \langle 0|0|1\rangle :$

> $v4 := \langle 0|1|1\rangle :$

> $v5 := \langle 1|1|1\rangle$:

> $v6 := \langle 4|2|0\rangle$:

> $v7 := \langle 3|0|-1\rangle$:

> $Basis([v1, v2, v3, v4, v5, v6]);$

$$\left[\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\right] \tag{19.74}$$

> $bas := Basis(\{v4, v5, v6\});$

$$bas := \left\{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 4 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}\right\} \tag{19.75}$$

> $DotProduct(bas[1], bas[2])$

$$6 \tag{19.76}$$

> $DotProduct(bas[3], bas[2])$

$$2 \tag{19.77}$$

> $DotProduct(bas[1], bas[3])$

$$2 \tag{19.78}$$

> $ord := GramSchmidt([v4, v5, v6]);$

$$ord := \left[\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & -1 \end{bmatrix}\right] \tag{19.79}$$

> $DotProduct(ord[2], ord[3])$

$$0 \tag{19.80}$$

> $DotProduct(ord[1], ord[3])$

$$0 \tag{19.81}$$

> $DotProduct(ord[1], ord[2])$

$$0 \tag{19.82}$$

>

Gradient(f, c), Del(f,c), Nabla(f,c) – *f ,*
c – (необязательный) список переменных или координат

$$\operatorname{grad}\varphi = \nabla\varphi = \frac{\partial\varphi}{\partial x}\vec{e}_x + \frac{\partial\varphi}{\partial y}\vec{e}_y + \frac{\partial\varphi}{\partial z}\vec{e}_z.$$

Divergence(F) *– дивергенция векторного поля F*

$$\operatorname{div}\ \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

$$\operatorname{div}\ \mathbf{F} = \nabla\cdot\mathbf{F}$$

Curl(F) *– ротор векторного поля*

$$\operatorname{rot}\mathbf{F} \equiv \nabla\times\mathbf{F}$$

$$\operatorname{rot}\,(F_x\mathbf{e}_x + F_y\,\mathbf{e}_y + F_z\mathbf{e}_z) =$$
$$= (\partial_y F_z - \partial_z F_y)\,\mathbf{e}_x + (\partial_z F_x - \partial_x F_z)\,\mathbf{e}_y + (\partial_x F_y - \partial_y F_x)\,\mathbf{e}_z \equiv$$
$$\equiv \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\mathbf{e}_x + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\mathbf{e}_y + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\mathbf{e}_z.$$

> *Curl(v);*

$$(0)\bar{e}_r + (0)\bar{e}_\theta + \left(\frac{\sin(\theta) + r\sin(\theta)}{r}\right)\bar{e}_z \qquad \textbf{(19.83)}$$

> *restart :*

> *with(VectorCalculus) :*

> *v := VectorField(⟨x^3 + y^3, 2·exp(y), sin(z)⟩, 'cartesian'[x, y, z]);*

$$v := \left(x^3 + y^3\right)\bar{e}_x + \left(2\,e^y\right)\bar{e}_y + \left(\sin(z)\right)\bar{e}_z \qquad \textbf{(19.84)}$$

> *Divergence(v);*

$$3\,x^2 + 2\,e^y + \cos(z) \qquad \textbf{(19.85)}$$

> *Curl(v);*

$$(0)\bar{e}_x + (0)\bar{e}_y + \left(-3\,y^2\right)\bar{e}_z \qquad \textbf{(19.86)}$$

> *with(VectorCalculus) :*

> *Gradient(x^2 + y^2, [x, y]);*

$$(2\,x)\bar{e}_x + (2\,y)\bar{e}_y \qquad \textbf{(19.87)}$$

$$\Delta F = \frac{\partial^2 F}{\partial x_1^2} + \frac{\partial^2 F}{\partial x_2^2} + \ldots + \frac{\partial^2 F}{\partial x_n^2}$$

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

> *with(VectorCalculus) :*

> *Laplacian(x^2 + y^2 + z^2, [x, y, z])*

$$6 \qquad \textbf{(19.88)}$$

> *Laplacian(f(r, θ), 'polar'[r, θ])*

$$\frac{\frac{\partial}{\partial r} f(r,\theta) + r\left(\frac{\partial^2}{\partial r^2} f(r,\theta)\right) + \frac{\frac{\partial^2}{\partial \theta^2} f(r,\theta)}{r}}{r} \qquad \textbf{(19.89)}$$

> *SetCoordinates('cylindrical'[r, θ, z])*

$$cylindrical_{r,\,\theta,\,z} \qquad \textbf{(19.90)}$$

> $Laplacian(f(r,\theta,z))$

$$\frac{\frac{\partial}{\partial r} f(r,\theta,z) + r\left(\frac{\partial^2}{\partial r^2} f(r,\theta,z)\right) + \frac{\frac{\partial^2}{\partial \theta^2} f(r,\theta,z)}{r} + r\left(\frac{\partial^2}{\partial z^2} f(r,\theta,z)\right)}{r} \qquad (19.91)$$

> $v := VectorField(\langle exp(x), sin(y), x\cdot y\cdot z^2\rangle, 'cartesian'[x, y, z]);$

$$v := (e^x)\bar{e}_x + (sin(y))\bar{e}_y + (x y z^2)\bar{e}_z \qquad (19.92)$$

> $Laplacian(v);$

$$(e^x)\bar{e}_x + (-sin(y))\bar{e}_y + (2xy)\bar{e}_z \qquad (19.93)$$

$$J(x) = \begin{pmatrix} \frac{\partial u_1}{\partial x_1}(x) & \frac{\partial u_1}{\partial x_2}(x) & \cdots & \frac{\partial u_1}{\partial x_n}(x) \\ \frac{\partial u_2}{\partial x_1}(x) & \frac{\partial u_2}{\partial x_2}(x) & \cdots & \frac{\partial u_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial u_m}{\partial x_1}(x) & \frac{\partial u_m}{\partial x_2}(x) & \cdots & \frac{\partial u_m}{\partial x_n}(x) \end{pmatrix}$$

Jacobian*(f, v, det);*

Jacobian(f, v=p, det) – матрица Якоби и якобиан f, где
f - - ,
v – (необязательный) список переменных дифференцирования или
. v = p ,

det – (необязательный) параметр в виде determinant=true/false
( ) . d e t e r m i n a n t
d e t e r m i n a n t = t r u e

> $with(VectorCalculus):$

> $Jacobian([r\,cos(t), r\,sin(t), r^2 t], [r, t]);$

$$\begin{bmatrix} cos(t) & -r\,sin(t) \\ sin(t) & r\,cos(t) \\ 2rt & r^2 \end{bmatrix} \qquad (19.94)$$

>

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

$$\begin{bmatrix} 2r & 0 \\ t & r \end{bmatrix} \qquad (19.95)$$

> $M, d := Jacobian([r\,sin(\phi)\,cos(\theta), r\,sin(\phi)\,sin(\theta), r\,cos(\phi)], [r, \phi, \theta], 'determinant')$

$$M, d := \begin{bmatrix} \sin(\phi)\cos(\theta) & r\cos(\phi)\cos(\theta) & -r\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi)\cos(\theta) \\ \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix}, \sin(\phi)^3\cos(\theta)^2 r^2 + \sin(\phi)^3\sin(\theta)^2 r^2 + \sin(\phi)\cos(\theta)^2\cos(\phi)^2 r^2 + \sin(\phi)\sin(\theta)^2\cos(\phi)^2 r^2 \tag{19.96}$$

> $simplify(d, trig)$

$$r^2\sin(\phi) \tag{19.97}$$

> $Jacobian(\langle x^2 + y, 2y\rangle, [x, y] = [-1, 1])$

$$\begin{bmatrix} -2 & 1 \\ 0 & 2 \end{bmatrix} \tag{19.98}$$

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

$$\begin{bmatrix} 2r & 0 \\ t & r \end{bmatrix} \tag{19.99}$$

> $SetCoordinates('cylindrical'[r, \theta, z])$

$$cylindrical_{r,\,\theta,\,z} \tag{19.100}$$

> $M, d := Jacobian([r\theta, \theta^2 z, z], 'determinant')$

$$M, d := \begin{bmatrix} \theta & r & 0 \\ 0 & 2\theta z & \theta^2 \\ 0 & 0 & 1 \end{bmatrix}, 2\theta^2 z \tag{19.101}$$

> $M, d := Jacobian([a r t, t w, w^2 a], 'determinant', [w, t, a] = [-1, 1, 2])$

$$M, d := \begin{bmatrix} 0 & 2r & r \\ 1 & -1 & 0 \\ -4 & 0 & 1 \end{bmatrix}, -6r \tag{19.102}$$

>