# Векторы и матрицы


| | |
|---|---|
| Author | Eugene Strakhov |
| Date | 2017-02-28T21:13:58 |
| Project | bf655f90-eca0-470e-9c17-2df7d93ad139 |
| Location | Handouts/Lecture 3 (Mar 08 - day off)/Lecture_3_(live).sagews |
| Original file | Lecture_3_(live).sagews |


# Векторы и матрицы

В данной лекции будут рассмотрены основы работы с векторами и матрицами в SageMath. Напомним, ранее в языке Python мы уже работали с векторами и матрицами на основе стандартных типов данных (а именно, списков). Встроенные типы `vector` и `matrix` сделают использование этих структур более удобным и интуитивно понятным.

Лекция построена на примерах.

## Создание векторов

```
#1
a = vector([1, -1.5, 1/3]) # вектор на основе списка
a # как видим, все элементы вектора приводятся к одному полю (BaseRing)
type(a)
```

```
(1.00000000000000, -1.50000000000000, 0.333333333333333)
```

```
#2
x = var('x')
a = vector([x, x^2, x^3]) # вектор может содержать символьные компоненты
a
type(a)
```

```
(x, x^2, x^3)
```

```
#3
try :
    a = vector([1, 2, 3, 'x']) # нельзя задавать в качестве элементов строки
except :
    print 'TypeError: unable to find a common ring for all elements'
```

```
TypeError: unable to find a common ring for all elements
```

```
#4
a = vector([sin(x) for x in xsrange(0, 2*pi, pi/4, include_endpoint=True)]) # на основе генератора
a
type(a)
```

```
(0, 1/2*sqrt(2), 1, 1/2*sqrt(2), 0, -1/2*sqrt(2), -1, -1/2*sqrt(2), 0)
```

```
#5
a = vector(QQ, [1, -1/3, 4.5]) # вектор над полем рациональных чисел
a
type(a)
a = vector(RDF, [1, -1/3, 4.5]) # тот же вектор над полем действительных чисел с двойной точностью
a
```

```
type(a)
# другие варианты - ZZ, RR, CC, CDF, SR
```

```
(1, -1/3, 9/2)

(1.0, -0.3333333333333333, 4.5)
```

```
#6
a = vector(ZZ, {0:1, 120:1, 156:1}) # вектор длины 157 над кольцом целых чисел; разреженный (sparse)
a
type(a)
```

```
(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1)
```

```
#7: случайные векторы
a1 = random_vector(ZZ, 4) # над кольцом целых чисел длины 4
a1
a2 = random_vector(ZZ, 5, x=-100, y=100) # задаем границы распределения случайных чисел
a2
a3 = random_vector(ZZ, 5, x=50) # только ВЕРХНЯЯ граница
a3
b1 = random_vector(QQ, 3) # над QQ
b1
b2 = random_vector(QQ, 10, num_bound=15, den_bound=5) # ограничения для числителя и знаменателя
b2
c1 = random_vector(RR, 6) # поле RR
c1
c2 = random_vector(RR, 6, min=-1.25, max=2.25) # задание границ распределения случайных чисел
c2
```

```
(-9, -1, 0, 7)
(7, 48, -75, 17, -9)
(19, 47, 30, 14, 7)
(1, 1, 2/377)
(0, -7, -13/5, 2/5, -7/3, 2, 14/5, -1/3, -15/4, 11)
(0.360310081520300, 0.108943716961480, 0.466564352781779, 0.760469407421565, -0.180118718501494,
0.250852242581010)
(0.902595401519875, 0.667463966165874, 1.67662607267187, 0.790861565739694, 1.82154748567383,
0.953257168264769)
```

```
#8: нулевой вектор
a = zero_vector(5) # по умолчанию - над ZZ
a
b = zero_vector(QQ, 5) # можно указать поле (кольцо)
b
```

```
(0, 0, 0, 0, 0)
(0, 0, 0, 0, 0)
```

## Операции над векторами

```
u = vector(QQ, [3, -5/4, 1/8])
v = vector(ZZ, [1, 8, -2])
u[1] # доступ по индексу; нумерация с 0
u[-1] # доступ по отрицательному индексу также возможен
try :
    u[1] = 0
    print 'Новый вектор:', u
except :
    print 'Элементы вектора изменять нельзя'
```

```
u.degree() # размерность вектора
sum(u) # сумма элементов вектора
w = -2*u+5*v # линейная комбинация; умножение на число, суммирование векторов
w
type(w) # результат - вектор над полем QQ
uv = u*v # скалярное произведение; результат - число
uv
uv = u.dot_product(v) # тоже скалярное произведение
uv
uv = u.cross_product(v) # векторное произведение; результат - вектор
uv
uv = u.outer_product(v) # внешнее произведение; результат - матрица
uv
```

```
-5/4
1/8
Новый вектор: (3, 0, 1/8)
3
25/8
(-1, 40, -41/4)

11/4
11/4
(-1, 49/8, 24)
[   3   24   -6]
[   0    0    0]
[ 1/8    1 -1/4]
```

## Нормы векторов

```
x, y, z = var('x y z')
p = var('p', domain='integer')
sample = vector([x, y, z])
show(sample.norm()) # по умолчанию - евклидова норма
show(sample.norm(1)) # манхэттенская норма
show(sample.norm(p)) # p-норма
show(vector([1, 3, -5]).norm(Infinity)) # норма бесконечность (максимальное по модулю значение)
```

$$\sqrt{|x|^2 + |y|^2 + |z|^2}$$

$$|x| + |y| + |z|$$

$$(|x|^p + |y|^p + |z|^p)^{\left(\frac{1}{p}\right)}$$

$$5$$

## Создание матриц

```
#1
M = matrix([[1, 2], [3, 4]]) # матрица из списка списков
M
show(M) # красивый вывод
```

```
[1 2]
[3 4]
```

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

```
#2
M = matrix(ZZ, [1, 2, 3]) # матрица из одной строки над ZZ
show(M)
type(M)
```

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

```
#3
M = matrix(QQ, 3, [1/x for x in xrange(1, 10)]) # матрица из 3 строк над QQ на основе списка
show(M)
```

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

```
#4
M = matrix(ZZ, [[2*i+j for i in range(5)] for j in range(5)]) # матрица на основе списка списков
show(M)
```

$$\begin{pmatrix} 0 & 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 & 9 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 5 & 7 & 9 & 11 \\ 4 & 6 & 8 & 10 & 12 \end{pmatrix}$$

```
#5
x, y = var('x y')
M = matrix(SR, [[x, x^2, x^3], [y, y^2, y^3]]) # символьная матрица
show(M)
```

$$\begin{pmatrix} x & x^2 & x^3 \\ y & y^2 & y^3 \end{pmatrix}$$

```
#6: нулевая матрица
M = zero_matrix(ZZ, 5, 6) # матрица из нулей над ZZ размерности 5x6
show(M)
```

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

```
#7: единичная матрица
E = identity_matrix(5) # 5x5
show(E)
```

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

```
#8: блочная матрица
A = matrix([[1, 2], [3, 4]])
B = matrix(ZZ, 3, 3, [x for x in range(1, 10)])
C = block_matrix([[A, 0], [0, B]]) # можно указывать имена матриц, 0 или 1 (нулевая, единичная соответственн
show(C)
```

$$\left(\begin{array}{cc|ccc} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 4 & 5 & 6 \\ 0 & 0 & 7 & 8 & 9 \end{array}\right)$$

```
#9: разреженная (sparse) матрица
M = matrix(ZZ, 10, 20, {(3,3):5, (5,6):-1, (9,5):10})
show(M)
```

$$\left(\begin{array}{rrrrrrrrrrrrrrrrrrrr}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

```
#10: случайная матрица
M1 = random_matrix(ZZ, 5, 5) # ZZ - по умолчанию
M2 = random_matrix(ZZ, 5, 5, x=0, y=10) # указание верхней и нижней границ
M3 = random_matrix(QQ, 3, 4, num_bound=5, den_bound=10) # QQ
M4 = random_matrix(RR, 5, 1, min=-5, max=5) # вектор-столбец
show(M1, M2, M3, M4)
```

$$\left(\begin{array}{rrrrr}
1 & -6 & 1 & 0 & -10 \\
-32 & 1 & -1 & -1 & 1 \\
-1 & 2 & 10 & 0 & 2 \\
4 & 1 & 0 & -1 & 2 \\
-2 & 4 & -22 & -109 & 11
\end{array}\right) \left(\begin{array}{rrrrr}
4 & 9 & 9 & 8 & 4 \\
1 & 5 & 5 & 3 & 6 \\
3 & 4 & 6 & 4 & 3 \\
7 & 8 & 1 & 3 & 6 \\
8 & 5 & 4 & 4 & 5
\end{array}\right) \left(\begin{array}{rrrr}
\frac{3}{8} & \frac{1}{4} & \frac{1}{7} & \frac{5}{7} \\
-1 & \frac{1}{2} & -\frac{2}{3} & 0 \\
\frac{1}{2} & -2 & \frac{1}{2} & -2
\end{array}\right) \left(\begin{array}{r}
1.18103487502450 \\
1.97115068639619 \\
0.772758495174996 \\
-0.244367843676320 \\
-2.64496190019996
\end{array}\right)$$

# Доступ к элементам матриц

```
# Доступ по индексу
A = random_matrix(ZZ, 5, 5)
show(A)
A[0,4] = 100 # ВНИМАНИЕ! "Традиционное" A[0][4]=100 приводит к ошибке
show(A)
A[2,3] = 1/2 # ОШИБКА, так как матрица определена над ZZ
```

$$\left(\begin{array}{rrrrr}
-2 & 6 & -23 & -2 & 1 \\
2 & -1 & 6 & 0 & -2 \\
2 & 3 & -1 & -2 & 3 \\
-1 & 0 & 2 & 2 & 2 \\
-27 & 1 & 6 & 2 & 0
\end{array}\right)$$

$$\left(\begin{array}{rrrrr}
-2 & 6 & -23 & -2 & 100 \\
2 & -1 & 6 & 0 & -2 \\
2 & 3 & -1 & -2 & 3 \\
-1 & 0 & 2 & 2 & 2 \\
-27 & 1 & 6 & 2 & 0
\end{array}\right)$$

```
Error in lines 5-5
Traceback (most recent call last):
  File "/projects/sage/sage-7.5/local/lib/python2.7/site-packages/smc_sagews/sage_server.py", line 982, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in
  File "sage/matrix/matrix0.pyx", line 1377, in sage.matrix.matrix0.Matrix.__setitem__ (/projects/sage/sage-7.5/src/build/cythonized/sage/matrix/matrix0.c:8308)
    self.set_unsafe(row, col, self._coerce_element(value))
  File "sage/matrix/matrix0.pyx", line 1482, in sage.matrix.matrix0.Matrix._coerce_element (/projects/sage/sage-7.5/src/build/cythonized/sage/matrix/matrix0.c:9787)
    return self._base_ring(x)
  File "sage/structure/parent.pyx", line 955, in sage.structure.parent.Parent.__call__ (/projects/sage/sage-7.5/src/build/cythonized/sage/structure/parent.c:9852)
    return mor._call_(x)
  File "sage/rings/rational.pyx", line 3913, in sage.rings.rational.Q_to_Z._call_ (/projects/sage/sage-7.5/src/build/cythonized/sage/rings/rational.c:33031)
    raise TypeError("no conversion of this rational to integer")
TypeError: no conversion of this rational to integer

*** WARNING: Code contains non-ascii characters ***
```

```
# Доступ к строкам и столбцам
A = random_matrix(ZZ, 5, 5)
show(A)
show(A[2]) # A[i] - i-я строка матрицы (нумерация с 0)
type(A[2]) # это вектор
show(A.row(2)) # A.row(i) - i-я строка
type(A.row(2)) # это также вектор
show(A.rows()) # список строк
show(A.columns()) # список столбцов
```

$$\left(\begin{array}{rrrrr} -1 & -15 & -1 & 4 & -13 \\ 1 & 0 & -16 & 3 & 1 \\ -2 & 1 & 6 & 0 & -3 \\ -2 & -1 & 1 & -2 & -10 \\ 19 & 3 & -1 & -3 & -1 \end{array}\right)$$

$$\left(-2,\,1,\,6,\,0,\,-3\right)$$

$$\left(-2,\,1,\,6,\,0,\,-3\right)$$

$$\left[\left(-1,\,-15,\,-1,\,4,\,-13\right),\,\left(1,\,0,\,-16,\,3,\,1\right),\,\left(-2,\,1,\,6,\,0,\,-3\right),\,\left(-2,\,-1,\,1,\,-2,\,-10\right),\,\left(19,\,3,\,-1,\,-3,\,-1\right)\right]$$

$$\left[\left(-1,\,1,\,-2,\,-2,\,19\right),\,\left(-15,\,0,\,1,\,-1,\,3\right),\,\left(-1,\,-16,\,6,\,1,\,-1\right),\,\left(4,\,3,\,0,\,-2,\,-3\right),\,\left(-13,\,1,\,-3,\,-10,\,-1\right)\right]$$

## Срезы матриц

```
# Все операции возвращают новую матрицу
A = random_matrix(ZZ, 5, 10)
show('$A='+latex(A)+'$')
show(A.matrix_from_columns([1, 5, 1])) # матрица из столбцов исходной матрицы; столбцы могут повторяться
show(A.matrix_from_rows([2, 3, 1])) # матрица из строк исходной матрицы
show(A.matrix_from_rows_and_columns([2, 4, 2], [3, 1])) # матрица из строк и столбцов
show(A.submatrix(1, 1, 2, 3)) # подматрица с позиции (1,1) размером 2x3
# Традиционные Python-срезы также работают!
show(A[2:])
show(A[:, 1]) # столбец
show(A[2:4, 1:7:2])
show(A[:, ::-1]) # что это будет?
```

$$A=\left(\begin{array}{rrrrrrrrrr} 0 & 5 & 2 & 0 & 2 & 0 & -7 & 1 & 1 & 0 \\ -1 & 0 & -12 & 1 & -1 & 0 & 4 & 1 & -1 & 1 \\ -32 & -1 & 2 & 1 & 3 & 0 & -2 & 0 & -1 & 2 \\ 2 & 0 & -10 & -2 & -3 & 0 & 1 & -1 & 1 & 1 \\ 0 & 0 & 1 & -2 & 1 & -3 & -1 & -1 & 1 & 23 \end{array}\right)$$

$$\left(\begin{array}{rrr} 5 & 0 & 5 \\ 0 & 0 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & -3 & 0 \end{array}\right)$$

$$\left(\begin{array}{rrrrrrrrrr} -32 & -1 & 2 & 1 & 3 & 0 & -2 & 0 & -1 & 2 \\ 2 & 0 & -10 & -2 & -3 & 0 & 1 & -1 & 1 & 1 \\ -1 & 0 & -12 & 1 & -1 & 0 & 4 & 1 & -1 & 1 \end{array}\right)$$

$$\left(\begin{array}{rr} 1 & -1 \\ -2 & 0 \\ 1 & -1 \end{array}\right)$$

$$\left(\begin{array}{rrr} 0 & -12 & 1 \\ -1 & 2 & 1 \end{array}\right)$$

$$\left(\begin{array}{rrrrrrrrrr} -32 & -1 & 2 & 1 & 3 & 0 & -2 & 0 & -1 & 2 \\ 2 & 0 & -10 & -2 & -3 & 0 & 1 & -1 & 1 & 1 \\ 0 & 0 & 1 & -2 & 1 & -3 & -1 & -1 & 1 & 23 \end{array}\right)$$

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

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

$$\begin{pmatrix} 0 & 1 & 1 & -7 & 0 & 2 & 0 & 2 & 5 & 0 \\ 1 & -1 & 1 & 4 & 0 & -1 & 1 & -12 & 0 & -1 \\ 2 & -1 & 0 & -2 & 0 & 3 & 1 & 2 & -1 & -32 \\ 1 & 1 & -1 & 1 & 0 & -3 & -2 & -10 & 0 & 2 \\ 23 & 1 & -1 & -1 & -3 & 1 & -2 & 1 & 0 & 0 \end{pmatrix}$$

```
File:
Docstring :
```

## Комбинирование матриц

```
A = random_matrix(QQ, 3, 2)
show('A:', A)
B = random_matrix(QQ, 3, 3)
show('B:', B)
v = random_vector(QQ, 2)
show('v:', v)
show(A.augment(B)) # добавление столбцов справа; аргумент B должен быть матрицей
show(A.stack(v)) # матрица A в верхних строках, далее - строки аргумента v (v может быть и матрицей)
show(A.block_sum(B)) # блочная матрица, вверху слева A, внизу справа B
```

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

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

v: $\left(1, -\frac{1}{10}\right)$

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

$$\begin{pmatrix} -\frac{1}{2} & 0 \\ -\frac{1}{2} & 2 \\ -1 & 0 \\ 1 & -\frac{1}{10} \end{pmatrix}$$

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

## Операции над матрицами

### Алгебраические операции

```
A = random_matrix(ZZ, 3, 4)
B = random_matrix(QQ, 3, 3)
C = random_matrix(QQ, 3, 3)
v = random_vector(QQ, 4)
w = random_vector(ZZ, 3)
x = var('x')
```

```
show(2*A, 0.5*B, B^2, 2*B-C, A*v, w*B, x^2*A)
if not B.is_singular() :
    show(B^(-1))
if not (B^3).is_singular() :
    show(B^(-3))
```

$$\begin{pmatrix} -2 & 2 & -2 & -8 \\ -12 & 2 & 2 & -2 \\ 0 & 0 & -30 & 6 \end{pmatrix} \begin{pmatrix} 0.500000000000000 & 0.250000000000000 & 1.00000000000000 \\ 1.00000000000000 & 0.500000000000000 & -1.00000000000000 \\ 0.000000000000000 & -1.00000000000000 & -1.00000000000000 \end{pmatrix} \begin{pmatrix} 2 & -3 & -3 \\ 4 & 6 & 6 \\ -4 & 2 & 8 \end{pmatrix} \begin{pmatrix} \frac{3}{2} & 1 & 3 \\ 4 & 4 & -\frac{7}{2} \\ 0 & -5 & -3 \end{pmatrix} (16,\ 1,\ -27$$

$$(4,\ 0,\ 0) \begin{pmatrix} -x^2 & x^2 & -x^2 & -4\,x^2 \\ -6\,x^2 & x^2 & x^2 & -x^2 \\ 0 & 0 & -15\,x^2 & 3\,x^2 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{3} & \frac{1}{6} & -\frac{1}{2} \\ \frac{1}{3} & -\frac{1}{6} & 0 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & 0 \\ -\frac{7}{27} & -\frac{7}{108} & -\frac{1}{9} \\ \frac{4}{27} & \frac{1}{27} & \frac{1}{36} \end{pmatrix}$$

### Нормы и другие числовые характеристики

```
A = random_matrix(QQ, 3, 3)
show(A)
print 'Ранг:', A.rank()
print 'Определитель:', A.determinant()
print 'Определитель:', A.det()
print 'След:', A.trace() # след - сумма элементов главной диагонали
print 'Спектральная норма:', A.norm() # A.norm(2) - то же самое
print 'Максимальная сумма элементов столбца:', A.norm(1)
print 'Максимальная сумма элементов строки:', A.norm(Infinity)
print 'Норма Фробениуса (евклидова):', A.norm('frob')
```

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

```
Ранг: 3
Определитель: -5
Определитель: -5
След: 1
Спектральная норма: 3.48539903308
Максимальная сумма элементов столбца: 5.0
Максимальная сумма элементов строки: 4.0
Норма Фробениуса (евклидова): 4.12310562562
```

### Матричные операции

```
A = random_matrix(ZZ, 3, 3)
show('A:', A)
show(A.transpose()) # транспонирование; результат - новая матрица
show(A.antitranspose()) # антитранспонирование; результат - новая матрица
show(A.inverse()) # обратная матрица; результат - новая матрица
show(A^(-1)) # также обратная матрица
```

A: $\begin{pmatrix} -13 & -3 & 0 \\ -4 & 2 & 8 \\ 39 & 1 & -8 \end{pmatrix}$

$$\begin{pmatrix} -13 & -4 & 39 \\ -3 & 2 & 1 \\ 0 & 8 & -8 \end{pmatrix}$$

$$\begin{pmatrix} -8 & 8 & 0 \\ 1 & 2 & -3 \\ 39 & -4 & -13 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{22} & \frac{1}{22} & \frac{1}{22} \\ -\frac{35}{66} & -\frac{13}{66} & -\frac{13}{66} \\ \frac{41}{264} & \frac{13}{66} & \frac{19}{264} \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{22} & \frac{1}{22} & \frac{1}{22} \\ -\frac{35}{66} & -\frac{13}{66} & -\frac{13}{66} \\ \frac{41}{264} & \frac{13}{66} & \frac{19}{264} \end{pmatrix}$$

```
A = random_matrix(ZZ, 3, 3)
show('A:', A)
A.rescale_row(0, 2) # умножить первую строку на 2; эта операция ИЗМЕНЯЕТ саму матрицу
show(A)
show(A.with_rescaled_row(0, 2)) # аналог, но при этом A не меняется, а возвращается новая матрица
A.rescale_col(1, -1)
show(A.with_rescaled_col(1, -1)) # то же самое, но для столбцов
A.add_multiple_of_row(0, 1, 2) # A[0] + A[1]*2; ИЗМЕНЯЕТ матрицу; with_added_multiple_of_row
A.add_multiple_of_column(0, 1, 2) # A[:,0] + A[:,1]*2; ИЗМЕНЯЕТ матрицу; with_added_multiple_of_column
show('A:', A)
# A.swap_rows(1, 2) # перестановка строк; ИЗМЕНЯЕТ матрицу
show('A.with_swapped_rows(1, 2):', A.with_swapped_rows(1, 2))
# A.swap_columns(0, 2) # перестановка столбцов; ИЗМЕНЯЕТ матрицу
show('A.with_swapped_columns(0, 2):', A.with_swapped_columns(0, 2))
show('A.delete_rows([1, 2]):', A.delete_rows([1, 2])) # удаление строк; результат - новая матрица
show('A.delete_columns([0]):', A.delete_columns([0])) # удаление столбцов; результат - новая матрица
show('A:', A)
```

A: $\begin{pmatrix} -3 & -1 & 0 \\ 1 & 1 & 2 \\ 2 & 14 & 2 \end{pmatrix}$

$$\begin{pmatrix} -6 & -2 & 0 \\ 1 & 1 & 2 \\ 2 & 14 & 2 \end{pmatrix}$$

$$\begin{pmatrix} -12 & -4 & 0 \\ 1 & 1 & 2 \\ 2 & 14 & 2 \end{pmatrix}$$

$$\begin{pmatrix} -6 & -2 & 0 \\ 1 & 1 & 2 \\ 2 & 14 & 2 \end{pmatrix}$$

A: $\begin{pmatrix} -4 & 0 & 4 \\ -1 & -1 & 2 \\ -26 & -14 & 2 \end{pmatrix}$

A.with_swapped_rows(1, 2): $\begin{pmatrix} -4 & 0 & 4 \\ -26 & -14 & 2 \\ -1 & -1 & 2 \end{pmatrix}$

A.with_swapped_columns(0, 2): $\begin{pmatrix} 4 & 0 & -4 \\ 2 & -1 & -1 \\ 2 & -14 & -26 \end{pmatrix}$

A.delete_rows([1, 2]): $\begin{pmatrix} -4 & 0 & 4 \end{pmatrix}$

A.delete_columns([0]): $\begin{pmatrix} 0 & 4 \\ -1 & 2 \\ -14 & 2 \end{pmatrix}$

A: $\begin{pmatrix} -4 & 0 & 4 \\ -1 & -1 & 2 \\ -26 & -14 & 2 \end{pmatrix}$

## Логические операции

```
A = random_matrix(ZZ, 3, 3, x=0, y=2)
show(A)
A.is_zero()
A.is_one()
A.is_scalar()
A.is_symmetric()
A.is_square()
```

```
A.is_singular()
A.is_invertible()
A.is_positive_definite()
```

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

```
False
False
False
True
True
True
False
False
```

## Собственные значения и собственные векторы

```
A = random_matrix(RR, 3, 3)
A.eigenvalues() # собственные значения; ПРЕДУПРЕЖДЕНИЕ о том, что возможны ошибки при округлении
B = matrix([[5, 6, 3], [-1, 0, 1], [1, 2, -1]])
B.charpoly('lambda') # характеристический полином, т. е. det(A-lambda*E)
B.charpoly() # по умолчанию - полином от x
B.eigenvalues()
show(B.eigenvectors_left()) # x*A = lambda*x; ответ в виде списка кортежей вида (СЗ, СВ (список), кратность (
show(B.eigenvectors_right()) # A*x = lambda*x; векторы-столбцы
show(B.eigenmatrix_left()) # диагональная матрица СЗ + матрица СВ (векторы-строки)
show(B.eigenmatrix_right()) # диагональная матрица СЗ + матрица СВ (векторы-столбцы)
```

```
[0.0961851049269839, 1.02815436359211 - 0.450899757860235*I, 1.02815436359211 + 0.450899757860235*I]
lambda^3 - 4*lambda^2 - 4*lambda + 16
x^3 - 4*x^2 - 4*x + 16
[4, 2, -2]
```

$$[(4, [(1, 2, 1)], 1), (2, [(1, 6, 3)], 1), (-2, [(1, 2, -5)], 1)]$$

$$[(4, [\left(1, -\frac{2}{9}, \frac{1}{9}\right)], 1), (2, [\left(1, -\frac{1}{2}, 0\right)], 1), (-2, [(0, 1, -2)], 1)]$$

$$(\begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -2 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 1 & 6 & 3 \\ 1 & 2 & -5 \end{pmatrix})$$

$$(\begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -2 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ -\frac{2}{9} & -\frac{1}{2} & 1 \\ \frac{1}{9} & 0 & -2 \end{pmatrix})$$

## Жорданова нормальная форма

```
# Работает только для QQ и ZZ
A = matrix(QQ, 3, [11, 2, -8, 2, 2, 10, -8, 10, 5])
jordan = A.jordan_form(transformation=True, base_ring=QQ)
show('$A=P^{-1}\cdot J\cdot P$')
show('$J='+latex(jordan[0])+'$')
show('$P='+latex(jordan[1])+'$')
```

$$A = P^{-1} \cdot J \cdot P$$

$$J = \left(\begin{array}{r|r|r} 18 & 0 & 0 \\ \hline 0 & 9 & 0 \\ \hline 0 & 0 & -9 \end{array}\right)$$

$$P = \begin{pmatrix} 1 & 1 & 1 \\ -\frac{1}{2} & 1 & -2 \\ -1 & \frac{1}{2} & 2 \end{pmatrix}$$

# Решение систем линейных алгебраических уравнений (СЛАУ)

```
A = matrix(QQ, [[1, 2], [3, 4]])
b = vector(QQ, [3, 4])
A.solve_right(b) # решение системы A*x == b; ответ - вектор
A\b # то же самое (т. н. "левое деление")
A.solve_left(b) # x*A == b
C = matrix(QQ, [[5, 6], [7, 8]])
show(A.solve_right(C)) # решение матричного уравнения A*X == C
show(A.solve_left(C))
D = matrix([[1, 2, 3], [4, 5, 6]]) # не квадратная матрица!
D.solve_right(b) # частное решение (при x3=0)
```

```
(-2, 5/2)
(-2, 5/2)
(0, 1)
```

$$\begin{pmatrix} -3 & -4 \\ 4 & 5 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 2 \\ -2 & 3 \end{pmatrix}$$

```
(-7/3, 8/3, 0)
```