Sage Quick Reference: Linear Algebra

Robert A. Beezer (w/mod. by nu)

Sage Version 3.4

http://wiki.sagemath.org/quickref

Based on work by Peter Jipsen, William Stein

ベクトルの作成 Vector Constructions

Caution: ベクトルの添字は 0 始まり

`u = vector(QQ, [1, 3/2, -1])` 有理数体上, 長さ 3

`v = vector(QQ, {2:4, 95:4, 210:0})`
211 成分, 非零なのは第 2 成分と第 95 成分の 4 だけ, sparse

ORGINAL TEXT

Caution: First entry of a vector is numbered 0
`u = vector(QQ, [1, 3/2, -1])` length 3 over rationals
`v = vector(QQ, {2:4, 95:4, 210:0})`
211 entries, nonzero in entry 2 and entry 95, sparse

ベクトルへの操作 Vector Operations

`u=vector(QQ,[1, 3/2, -1])`, `v=vector(ZZ,[1, 8, -2])`

`2*u - 3*v` 線型結合

`u.dot_product(v)`

`u.cross_product(v)` 順序: u×v

`u.inner_product(v)` u と v の内積

`u.pairwise_product(v)` 結果のベクトル

`u.norm() == u.norm(2)` ユークリッドノルム

`u.norm(1)` 要素の絶対値の和

`u.norm(Infinity)` 絶対値が最大の要素

`A.gram_schmidt()` 行列 A の行を変換

ORGINAL TEXT

`u=vector(QQ,[1, 3/2, -1])`, `v=vector(ZZ,[1, 8, -2])`
`2*u - 3*v` linear combination
`u.dot_product(v)`
`u.cross_product(v)` order: u×v
`u.inner_product(v)` inner product matrix from parent
`u.pairwise_product(v)` vector as a result
`u.norm() == u.norm(2)` Euclidean norm
`u.norm(1)` sum of entries
`u.norm(Infinity)` maximum entry
`A.gram_schmidt()` converts the rows of matrix A

行列の作成 Matrix Constructions

Caution: 行も列も添字は 0 始まり

`A = matrix(ZZ, [[1,2],[3,4],[5,6]])` 3 × 2 整数行列

`B = matrix(QQ, 2, [1,2,3,4,5,6])`
リストから 2 行の行列を作る. 従って 2 × 3 有理数行列.

`C = matrix(CDF, 2, 2, [[5*I, 4*I], [I, 6]])`
複素数, 53-bit 精度の行列

`Z = matrix(QQ, 2, 2, 0)` 零行列

`D = matrix(QQ, 2, 2, 8)` 対角成分は 8, それ以外は 0

`I = identity_matrix(5)` 5 × 5 単位行列

`J = jordan_block(-2,3)`
3 × 3 行列, 対角は −2, その一つ上は 1

`var('x y z'); K = matrix(SR,[[x,y+z],[0,x^2*z]])`
シンボリックな数式のなす環 SR の元を成分とする行列.

`L=matrix(ZZ, 20, 80, {(5,9):30, (15,77):-6})`
20 × 80, 2 つの要素だけ非零な行列, sparse

ORGINAL TEXT

Caution: Row, column numbering begins at 0
`A = matrix(ZZ, [[1,2],[3,4],[5,6]])`
3 × 2 over the integers
`B = matrix(QQ, 2, [1,2,3,4,5,6])`
2 rows from a list, so 2 × 3 over rationals
`C = matrix(CDF, 2, 2, [[5*I, 4*I], [I, 6]])`
complex entries, 53-bit precision
`Z = matrix(QQ, 2, 2, 0)` zero matrix
`D = matrix(QQ, 2, 2, 8)`
diagonal entries all 8, other entries zero
`I = identity_matrix(5)` 5 × 5 identity matrix
`J = jordan_block(-2,3)`
3 × 3 matrix, −2 on diagonal, 1's on super-diagonal
`var('x y z'); K = matrix(SR,[[x,y+z],[0,x^2*z]])`
symbolic expressions live in the ring SR
`L=matrix(ZZ, 20, 80, {(5,9):30, (15,77):-6})`
20 × 80, two non-zero entries, sparse representation

行列の積 Matrix Multiplication

`u=vector(QQ,[1,2,3])`, `v=vector(QQ,[1,2])`,
`A = matrix(QQ, [[1,2,3],[4,5,6]])`,
`B = matrix(QQ, [[1,2],[3,4]])`,
`u*A`, `A*v`, `B*A`, `B^6`, `B^(-3)` などと出来る

`B.iterates(v, 6)` で $vB^0, vB^1, \ldots, vB^5$ が出来る
`rows = False` なら v は行列の巾の右にくる

`f(x)=x^2+5*x+3` なら `f(B)` と出来る

`B.exp()` 行列の指数関数, つまり $\sum_{k=0}^{\infty} \frac{B^k}{k!}$

ORGINAL TEXT

`u=vector(QQ,[1,2,3])`, `v=vector(QQ,[1,2])`,
`A = matrix(QQ, [[1,2,3],[4,5,6]])`,
`B = matrix(QQ, [[1,2],[3,4]])`,
`u*A`, `A*v`, `B*A`, `B^6`, `B^(-3)` all possible
`B.iterates(v, 6)` produces $vB^0, vB^1, \ldots, vB^5$
`rows = False` moves v to right of matrix powers
`f(x)=x^2+5*x+3` then `f(B)` is possible
`B.exp()` matrix exponential, i.e. $\sum_{k=0}^{\infty} \frac{B^k}{k!}$

行列の空間 Matrix Spaces

`M = MatrixSpace(QQ, 3, 4)` 3 × 4 行列の 12 次元空間

`A = M([1,2,3,4,5,6,7,8,9,10,11,12])` 3 × 4 行列, M の元

`M.basis()`

`M.dimension()`

`M.zero_matrix()`

ORGINAL TEXT

`M = MatrixSpace(QQ, 3, 4)`
dimension 12 space of 3 × 4 matrices
`A = M([1,2,3,4,5,6,7,8,9,10,11,12])`
is a 3 × 4 matrix, an element of M
`M.basis()`
`M.dimension()`
`M.zero_matrix()`

行列の操作 Matrix Operations

`5*A+2*B` 線型結合

`A.inverse()` (`A^(-1)`, `~A` でも OK)
もし正則でないなら ZeroDivisionError

`A.transpose()`

`A.antitranspose()` 転置 + 順序の反転

`A.adjoint()` 余因子行列

`A.conjugate()` 成分ごとの複素共役

`A.restrict(V)` 不変部分空間 V への制限

ORGINAL TEXT

`5*A+2*B` linear combination
`A.inverse()`, also `A^(-1)`, `~A`
ZeroDivisionError if singular
`A.transpose()`
`A.antitranspose()` transpose + reverse orderings
`A.adjoint()` matrix of cofactors
`A.conjugate()` entry-by-entry complex conjugates
`A.restrict(V)` restriction on invariant subspace V

行基本変形 Row Operations

行変形: (直接行列を書き換える)

Caution: 最初の行は 0 行目

`A.rescale_row(i,a)` a*(i 行目)

`A.add_multiple_of_row(i,j,a)` a*(j 行目) + i 行目

`A.swap_rows(i,j)`

列基本変形は, row→col

書き換えたくない時は `B = A.with_rescaled_row(i,a)` 等.

ORGINAL TEXT

Row Operations: (change matrix in place)
Caution: first row is numbered 0
`A.rescale_row(i,a)` a*(row i)
`A.add_multiple_of_row(i,j,a)` a*(row j) + row i
`A.swap_rows(i,j)`
Each has a column variant, row→col
For a new matrix, use e.g. `B = A.with_rescaled_row(i,a)`

階段行列 Echelon Form

`A.echelon_form()`, `A.echelonize()`, `A.hermite_form()`

Caution: どの環の元かによって結果が代わってくる

`A = matrix(ZZ,[[4,2,1],[6,3,2]])`

`B = matrix(QQ,[[4,2,1],[6,3,2]])`

`A.echelon_form()` `B.echelon_form()`

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

`A.pivots()` 列空間を生成している列の添字

`A.pivot_rows()` 行空間を生成している行の添字

ORGINAL TEXT

`A.echelon_form()`, `A.echelonize()`, `A.hermite_form()`

Caution: Base ring affects results

```
A = matrix(ZZ,[[4,2,1],[6,3,2]])
B = matrix(QQ,[[4,2,1],[6,3,2]])
```

`A.echelon_form()` `B.echelon_form()`

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

`A.pivots()` indices of columns spanning column space

`A.pivot_rows()` indices of rows spanning row space

小行列など Pieces of Matrices

Caution: 行も列も添字は 0 から

`A.nrows()`, `A.ncols()`

`A[i,j]` i 行 j 列の成分

Caution: OK: `A[2,3] = 8`, Error: `A[2][3] = 8`

`A[i]` i 行目, (immutable Python tuple)

`A.row(i)` Sage vector として i 行目を返す

`A.column(j)` Sage vector として j 行目を返す

`A.list()` single Python list を返す. (row-major order)

`A.matrix_from_columns([8,2,8])`
リストにある列で新しい行列を作る. 列が重複してもよい.

`A.matrix_from_rows([2,5,1])`
リストにある行で新しい行列を作る. 未ソートでも可.

`A.matrix_from_rows_and_columns([2,4,2],[3,1])`
行と列から新しい行列

`A.rows()` 全ての行 (tuples のリスト)

`A.columns()` 全ての列 (tuples のリスト)

`A.submatrix(i,j,nr,nc)`
`(i,j)` から始め `nr` 行 `nc` 列を使った行列

`A[2:4,1:7]`, `A[0:8:2,3::-1]` Python-style list slicing

ORGINAL TEXT

Caution: row, column numbering begins at 0

`A.nrows()`, `A.ncols()`

`A[i,j]` entry in row `i` and column `j`

Caution: OK: `A[2,3] = 8`, Error: `A[2][3] = 8`

`A[i]` row `i` as immutable Python tuple

`A.row(i)` returns row `i` as Sage vector

`A.column(j)` returns column `j` as Sage vector

`A.list()` returns single Python list, row-major order

`A.matrix_from_columns([8,2,8])`
new matrix from columns in list, repeats OK

`A.matrix_from_rows([2,5,1])`
new matrix from rows in list, out-of-order OK

`A.matrix_from_rows_and_columns([2,4,2],[3,1])`
common to the rows and the columns

`A.rows()` all rows as a list of tuples

`A.columns()` all columns as a list of tuples

`A.submatrix(i,j,nr,nc)`
start at entry `(i,j)`, use `nr` rows, `nc` cols

`A[2:4,1:7]`, `A[0:8:2,3::-1]` Python-style list slicing

行列の組合せ Combining Matrices

`A.augment(B)` A を左に, B を右に置いてできる行列

`A.stack(B)` A を上に, B を下に置いてできる行列

`A.block_sum(B)` A を左上に, B を右下, ブロック対角行列

`A.tensor_product(B)` A に従って B の定数倍を配置した行列

ORGINAL TEXT

`A.augment(B)` A in first columns, B to the right

`A.stack(B)` A in top rows, B below

`A.block_sum(B)` Diagonal, A upper left, B lower right

`A.tensor_product(B)` Multiples of B, arranged as in A

行列のスカラー関数 Scalar Functions on Matrices

`A.rank()`

`A.nullity() == A.left_nullity()`

`A.right_nullity()`

`A.determinant() == A.det()`

`A.permanent()`

`A.trace()`

`A.norm() == A.norm(2)` ユークリッドノルム

`A.norm(1)` 列和の最大

`A.norm(Infinity)` 行和の最大

`A.norm('frob')` フロベニウスノルム

ORGINAL TEXT

`A.rank()`

`A.nullity() == A.left_nullity()`

`A.right_nullity()`

`A.determinant() == A.det()`

`A.permanent()`

`A.trace()`

`A.norm() == A.norm(2)` Euclidean norm

`A.norm(1)` largest column sum

`A.norm(Infinity)` largest row sum

`A.norm('frob')` Frobenius norm

行列の情報 Matrix Properties

`.is_zero()` (零行列?), `.is_one()` (単位行列?),
`.is_scalar()` (単位行列の定数倍?), `.is_square()`,
`.is_symmetric()`, `.is_invertible()`, `.is_nilpotent()`

ORGINAL TEXT

`.is_zero()` (totally?), `.is_one()` (identity matrix?),
`.is_scalar()` (multiple of identity?), `.is_square()`,
`.is_symmetric()`, `.is_invertible()`, `.is_nilpotent()`

固有値 Eigenvalues

`A.charpoly('t')` 変数を指定しない時のデフォルトは `x`

`A.characteristic_polynomial() == A.charpoly()`

`A.fcp('t')` 因数分解した特性多項式

`A.minpoly()` 最小多項式

`A.minimal_polynomial() == A.minpoly()`

`A.eigenvalues()` 重複のある未ソートのリスト.

`A.eigenvectors_left()` ベクトルは左, `_right` も有
三つ組 `(e,V,n)` のリストを返す:
`e`: 固有値
`V`: ベクトルのリスト, 固有空間の基底
`n`: 代数的重複度

`A.eigenmatrix_right()` ベクトルは右, `_left` も有
行列 2 つ `(D,P)` を返す:
`D`: 固有値が対角にある対角行列
`P`: 各列が固有ベクトルの行列 (left なら行)
もし対角化可能でなければ, 0 ベクトルが列に現れる

ORGINAL TEXT

`A.charpoly('t')` no variable specified defaults to x
`A.characteristic_polynomial() == A.charpoly()`

`A.fcp('t')` factored characteristic polynomial

`A.minpoly()` the minimum polynomial
`A.minimal_polynomial() == A.minpoly()`

`A.eigenvalues()` unsorted list, with mutiplicities

`A.eigenvectors_left()` vectors on left, `_right` too
Returns a list of triples, one per eigenvalue:
`e`: the eigenvalue
`V`: list of vectors, basis for eigenspace
`n`: algebraic multiplicity

`A.eigenmatrix_right()` vectors on right, `_left` too
Returns two matrices:
`D`: diagonal matrix with eigenvalues
`P`: eigenvectors as columns (rows for left version)
has zero columns if matrix not diagonalizable

分解 Decompositions

Note: どの環の元かによって使えないものあり

`A.jordan_form(transformation=True)` 行列のペアを返す:
`J`: 固有値に対するジョルダンブロックの行列
`P`: 正則行列
⇝ `A == P^(-1)*J*P`

`A.smith_form()` 行列の 3 つ組を返す:
`D`: 単因子の対角行列
`U`, `V`: 固有値 1 の行列
⇝ `D == U*A*V`

`A.LU()` 行列の 3 つ組を返す:
`P`: 置換行列
`L`: 下三角行列
`U`: 上三角行列
⇝ `P*A == L*U`

`A.QR()` 行列の組を返す:
`Q`: 直交行列

R: 上三角行列
⇝ A == Q*R

A.SVD() 行列の 3 つ組を返す:
U: 直交行列
S: zero off the diagonal, A と同じ次元
V: 直交行列
⇝ A == U*S*(V-conjugate-transpose)

A.symplectic_form()
A.hessenberg_form()
A.cholesky()

ORGINAL TEXT

Note: availability depends on base ring of matrix
A.jordan_form(transformation=True)
returns a pair of matrices:
J: matrix of Jordan blocks for eigenvalues
P: nonsingular matrix
so A == P^(-1)*J*P
A.smith_form() returns a triple of matrices:
D: elementary divisors on diagonal
U, V: with unit determinant
so D == U*A*V
A.LU() returns a triple of matrices:
P: a permutation matrix
L: lower triangular matrix
U: upper triangular matrix
so P*A == L*U
A.QR() returns a pair of matrices:
Q: an orthogonal matrix
R: upper triangular matrix
so A == Q*R
A.SVD() returns a triple of matrices:
U: an orthogonal matrix
S: zero off the diagonal, same dimensions as A
V: an orthogonal matrix
so A == U*S*(V-conjugate-transpose)
A.symplectic_form()
A.hessenberg_form()
A.cholesky()

方程式系の解 Solutions to Systems

A.solve_right(B) _left も有
A*X = B の解, ただし X はベクトル or 行列
A = matrix(QQ, [[1,2],[3,4]])
b = vector(QQ, [3,4])
とすると A\b は解 (-2, 5/2) を返す

ORGINAL TEXT

A.solve_right(B) _left too
is solution to A*X = B, where X is a vector or matrix
A = matrix(QQ, [[1,2],[3,4]])
b = vector(QQ, [3,4])
then A\b returns the solution (-2, 5/2)

ベクトル空間 Vector Spaces

U = VectorSpace(QQ, 4) 4 次元, 係数体は有理数体
V = VectorSpace(RR, 4) “係数体” は 53-bit 精度の実数
W = VectorSpace(RealField(200), 4)
“係数体” は 200 bit 精度
X = CC^4 4 次元, 53-bit 精度の複素数
Y = VectorSpace(GF(7), 4) 有限
Y.finite() は True を返す
len(Y.list()) は $7^4 = 2401$ elements を返す

ORGINAL TEXT

U = VectorSpace(QQ, 4) dimension 4, rationals as field
V = VectorSpace(RR, 4) “field” is 53-bit precision reals
W = VectorSpace(RealField(200), 4)
“field” has 200 bit precision
X = CC^4 4-dimensional, 53-bit precision complexes
Y = VectorSpace(GF(7), 4) finite
Y.finite() returns True
len(Y.list()) returns $7^4 = 2401$ elements

ベクトル空間の情報 Vector Space Properties

V.dimension()
V.basis()
V.echelonized_basis()
V.has_user_basis() 標準基底以外の基底を使っている?
V.is_subspace(W) W が V の部分空間なら True
V.is_full() (加群として) 階数が次元と等しいか?
Y = GF(7)^4, T = Y.subspaces(2)
T は Y の二次元部分空間に対する a generator object
[U for U in T] は Y の (2850 個の) 二次元部分空間のリスト

ORGINAL TEXT

V.dimension()
V.basis()
V.echelonized_basis()
V.has_user_basis() with non-canonical basis?
V.is_subspace(W) True if W is a subspace of V
V.is_full() rank equals degree (as module)?
Y = GF(7)^4, T = Y.subspaces(2)
T is a generator object for 2-D subspaces of Y
[U for U in T] is list of 2850 2-D subspaces of Y

部分空間の作成 Constructing Subspaces

span([v1,v2,v3], QQ) 環 QQ 上 v1,v3,v3 で生成される空間
行列 A に対し, 返されるのは
基礎環が体ならベクトル空間
基礎環が体でないなら加群
A.left_kernel() == A.kernel() right_ も有
A.row_space() == A.row_module()
A.column_space() == A.column_module()
A.eigenspaces_right() ベクトルは右, _left も有
固有値と右固有空間の組
もし V と W が部分空間なら
V.quotient(W) V の W による商空間
V.intersection(W) V と W の共通部分
V.direct_sum(W) V と W の直和
V.subspace([v1,v2,v3]) リストのベクトルによる部分空間

ORGINAL TEXT

span([v1,v2,v3], QQ) span of list of vectors over ring
For a matrix A, objects returned are
vector spaces when base ring is a field
modules when base ring is just a ring
A.left_kernel() == A.kernel() right_ too
A.row_space() == A.row_module()
A.column_space() == A.column_module()
A.eigenspaces_right() vectors on right, _left too
Pairs, having eigenvalue with its right eigenspace
If V and W are subspaces
V.quotient(W) quotient of V by subspace W
V.intersection(W) intersection of V and W
V.direct_sum(W) direct sum of V and W
V.subspace([v1,v2,v3]) specify basis vectors in a list

Dense vs Sparse Dense versus Sparse

Note: アルゴリズムは表現の仕方に依存するかもしれない
ベクトルと行列にはふたつの表現がある
Dense: リスト (ベクトル) リストのリスト (行列)
Sparse: Python dictionary
.is_dense(), .is_sparse() でチェックできる
A.sparse_matrix() A と等しい sparse な表現の行列を返す
A.dense_rows() A の dense な行ベクトルを返す
sparse(=True/False) というキーワードを持つコマンドも有.

ORGINAL TEXT

Note: Algorithms may depend on representation
Vectors and matrices have two representations
Dense: lists, and lists of lists
Sparse: Python dictionaries
.is_dense(), .is_sparse() to check
A.sparse_matrix() returns sparse version of A
A.dense_rows() returns dense row vectors of A
Some commands have boolean sparse keyword

環 Rings

Note: 多くのアルゴリズムが基礎環が何かに依存している
<object>.base_ring(R) ベクトル, 行列,... に対し
使用する環を指定する.
<object>.change_ring(R) ベクトル, 行列,... に対し
使用する環 (体) を R に変更する.
R.is_ring(), R.is_field()
R.is_integral_domain(), R.is_exact()
おもな環と体
ZZ 整数 $\mathbb{Z}$, 環
QQ 有理数 $\mathbb{Q}$, 体
QQbar 代数閉包 $\overline{\mathbb{Q}}$, 厳密 (exact)

`RDF` 倍精度実数, 近似 (inexact)
`RR` 53-bit 精度実数, 近似 (inexact)
`RealField(400)` 400-bit 精度実数, 近似 (inexact)
(`CDF`, `CC`, `ComplexField(400)` 複素数も有)
`RIF` 実区間演算, 体
`GF(2)` mod 2, 体, specialized implementations
`GF(p) == FiniteField(p)` p 素数, $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$, 体
`Integers(6)` integers mod 6, $\mathbb{Z}/6\mathbb{Z}$, 環
`CyclotomicField(7)` $\mathbb{Q}$ に 1 の 7 乗根を添加した体
`QuadraticField(-5, 'x')` $\mathbb{Q}$ に x=$\sqrt{-5}$ を添加した体
`SR` ring of symbolic expressions

ORGINAL TEXT

Note: Many algorithms depend on the base ring
`<object>.base_ring(R)` for vectors, matrices,...
to determine the ring in use
`<object>.change_ring(R)` for vectors, matrices,...
to change to the ring (or field), `R`.
`R.is_ring()`, `R.is_field()`
`R.is_integral_domain()`, `R.is_exact()`
Some ring and fields
`ZZ` integers, ring
`QQ` rationals, field
`QQbar` algebraic field, exact
`RDF` real double field, inexact
`RR` 53-bit reals, inexact
`RealField(400)` 400-bit reals, inexact
`CDF`, `CC`, `ComplexField(400)` complexes, too
`RIF` real interval field
`GF(2)` mod 2, field, specialized implementations
`GF(p) == FiniteField(p)` p prime, field
`Integers(6)` integers mod 6, ring only
`CyclotomicField(7)` rationals with 7^{th} root of unity
`QuadraticField(-5, 'x')` rationals adjoin x=$\sqrt{-5}$
`SR` ring of symbolic expressions

ベクトル空間 vs 加群 Vector Spaces versus Modules

加群とは (体では無く) 環上のベクトル空間 “みたいな” もの.
先に述べたコマンド多くは加群にも使うことが出来る.
いくつかの “ベクトル” は実際に加群の元.

ORGINAL TEXT

A module is “like” a vector space over a ring, not a field
Many commands above apply to modules
Some “vectors” are really module elements

もっと助けて More Help

コマンドの一部を書いて “tab-補完”
`<object>.` を書いて “tab-補完” で 関連するコマンドを表示
`<command>?` でコマンドの説明と例
`<command>??` でソースコードを表示

ORGINAL TEXT

“tab-completion” on partial commands
“tab-completion” on `<object.>` for all relevant methods
`<command>?` for summary and examples
`<command>??` for complete source code