49　行列計算

行列の計算は，数値計算の分野の中で重要な位置を占めています。線形微分方程式の解を計算する問題は，その多くの部分が行列計算に帰着され，Google の検索システムでは Web ページのランク付けのために，巨大な行列の固有ベクトルが計算されています。

49.1　ベクトルと行列の生成

まずは，Sage を使って手軽に行列計算を行う方法を紹介します。

ベクト行列の定義

Sage ではベクトルと行列はそれぞれ次のように表します：

(1) $\begin{pmatrix}1\\2\end{pmatrix}$ = `vector([1,2])`

(2) $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$ = `matrix([[1,2,3],[4,5,6]])`

(3) $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$ = `matrix(2,3,[1,2,3,4,5,6])`

上では，2 種類の行列の作り方を紹介しました。一つは上の (2) のように 2 重のリストによって表現する方法で，二つ目は (3) のようにリストを型で区切って行列を生成する方法です。(3) では 2,3 と指定することで 2 行 3 列の行列を生成しています。

具体的にベクトルや行列を作ってみましょう：

```
1 a = vector([1,2])      # aをベクトル(1,2)とする
2 print a
```

実行結果の例

```
(1, 2)
```

```
1 b = matrix([[1,2],[3,4]])  # 行列bを定義
2 print b
```

実行結果の例

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

```
1 c = matrix(2,3,[1,2,3,4,5,6])    # 行列cを定義
2 print c
```

実行結果の例

```
[1 2 3]
[4 5 6]
```

上のように，行列の成分をリストの形に具体的に書いて全て指定することによって，行列を作ることもできますが，行列の成分が何らかのパターンを持つ場合には，次のように関数を使って行列を作ります。

例えば $f(i,j)=i+j$ を (i,j) 成分に持つような 5×5 行列を作ってみます。

```
1 def elem(i,j):       # 関数elemの定義
2     return i+j^2     # i+j^2を返す
3
4 aa = matrix(5,5, elem)     # i,j成分が elem(i-1,j-1) となる行列
```

```
5 print aa
6 print aa[3,4]
```

実行結果の例

```
[ 0  1  4  9 16]
[ 1  2  5 10 17]
[ 2  3  6 11 18]
[ 3  4  7 12 19]
[ 4  5  8 13 20]
19    # aa の 4 行 5 列目の成分
```

Sage では行列の成分は 0 から数えるので，通常の行列の 1,1 成分は Sage では 0,0 成分となります。

49.2　ベクトルと行列の積

ベクトル同士の内積，行列とベクトルの積，行列同士の積は*で計算します。

ベクトルや行列の積

- ベクトル u と v の内積： u*v
- 行列 A とベクトル v の積： A*v
- 行列 A と B の積: A*B

たとえばベクトル $(3,-3,5)$ と $(1,1,-2)$ の内積は $3\times 1+(-3)\times 1+5\times(-2)=-10$ ですが，これを Sage にやらせるには

```
1 print vector([3,-3,5]) * vector([1,1,-2])
```

実行結果の例

```
-10
```

のようにします。ただし，内積は複素内積ではないので注意しましょう：

```
1 print vector([I,1]) * vector([I,1])      # I は虚数単位を表す。
```

実行結果の例

```
0          # I*I + 1*1 = -1+1=0 となる。複素共役はとらない。
```

同様に行列の積は次のように計算します：

```
1 print matrix([[1,2],[-4,2]]) * matrix([[2,-1],[-3,2]])
```

実行結果の例

```
[ -4   3]
[-14   8]
```

49.3　行列に対する基本的な操作

Sage では行列に対して様々な操作を行うことができますが，代表的なものは次のものです：

名前	Sage での記号	数学的意味
行列式	det(A)	$\det(A)$
逆行列	A^-1	A^{-1}
転置行列	transpose(A)	A^t
トレース	A.trace()	$\mathrm{tr}(A)$
$Ax=v$ を解く	A.solve_right(v)	$x=A^{-1}v$
階数	A.rank()	$\mathrm{rank}(A)=\dim \mathrm{Im}A$
核の次元	A.nullity()	$\dim\ker(A)$

行列式，転置はそれぞれ A.det() と A.transpose() で計算することもできます。

では，実際に上の命令を試してみましょう。

```
var('a b c d')      #a, b, c, dを不定元とする
A= matrix(2,2,[a,b,c,d]) # 行列Aを定義する
print A.transpose()      # Aの転置行列
print ''    # 改行
print det(A)             # Aの行列式
print ''    # 改行
print A.trace()          # Aのトレース
print ''    # 改行
print A^-1               # Aの逆行列
```

実行結果の例

```
[a c]
[b d]                                                    # 転置

-b*c + a*d                                               # 行列式

a + d                                                    # トレース

[1/a - b*c/(a^2*(b*c/a - d))          b/(a*(b*c/a - d))]    # 逆行列
[          c/(a*(b*c/a - d))             -1/(b*c/a - d)]
```

一般に行列 n 次正方行列 A の像 $\mathrm{Im}A$ と核 $\ker A$ の次元の間には

$$n = \dim \ker A + \mathrm{rank}A$$

という関係成り立つ事が知られています。実際の行列で試してみましょう。

```
mat = matrix(6,6, lambda i,j : i+j) # 行列を定義
print mat                           # 定義した行列をプリント
print mat.rank()                    # rank
print mat.nullity()                 # nullity
```

実行結果の例

```
[ 0  1  2  3  4  5]
[ 1  2  3  4  5  6]
[ 2  3  4  5  6  7]
[ 3  4  5  6  7  8]
[ 4  5  6  7  8  9]
[ 5  6  7  8  9 10]
2     #階数
4     #ker(mat) の次元
```

上の行列の定義では，無名関数を使うために lambda を使いましたが，先ほどの説明した関数 elem を使って行列を定義するのと同じ事です。

49.4　行列と係数環・係数体

コンピューターによる計算は，大きく分けて二つに分けられます。一つは厳密な計算で，もう一つは数値計算（近似計算）です。例えば，前者は数学的興味のために行われ，後者は応用上のために行われます。これら二つは目的が異なるため，同じ数学的概念 —例えば行列— を対象としていても，計算上は異なる取り扱いをしなければなりません。一般的に，厳密な計算には時間がかかるため，近似的な数値だけが必要なのであれば，厳密さを無視して近似計算をするようにプログラムしてやる必要があります。さいわい，Sage の行列計算では，簡単な指定をするだけで，厳密な計算か数値計算かを区別して計算を行わせることができます。

Sage で行列計算の固有値や固有ベクトル，ジョルダン標準形などを計算させるときには，どこの係数環（または体）で計算をするのか意識する必要があります。行列の成分がどの環・体であるかによって計算できること，できないことが異なってきます。ここで設定できる環の代表的なものには次のようなものがあります：

名前	Sage の記号	Sage の英語名	意味	精度
整数	ZZ	Integer Ring	整数のつくる環 $\mathbb{Z}$	厳密
有理数	QQ	Rational Field	有理数体 $\mathbb{Q}$	厳密
代数的数	QQbar	Algebraic Field	$\mathbb{Q}$ の代数閉包	厳密
形式的な環	SR	Symbolic Ring	形式的な環（ほぼ体）	厳密
$\mathbb{Z}/n\mathbb{Z}$	Integers(n)	$\mathbb{Z}/n\mathbb{Z}$	n を法とした整数	厳密
位数 n の有限体	GF(n)	Finite Field of size n	位数 n の有限体	厳密
実数 (53bit)	RR	Real Field with 53 bits of precision	53 ビットの実数	近似
実数 (53bit)	RDF	Real Double Field	倍精度浮動小数点実数	近似
複素数 (53bit)	CC	Complex Field with 53 bits of precision	53 ビットの複素数	近似
実数 (400bit)	RealField(400)		400 ビットの実数	近似

行列が定義されている環を Sage では base ring といいます。例えば，整数の範囲で行列計算を行うのであれば base ring は整数にします。実数値で近似計算するのであれば，base ring は RR または RDF とします。より精度が必要であれば，RealField(400) を指定して高精度で計算します。行列の係数がすべて実数値であっても，その固有値には複素数が現れることがあるので，そのような計算をする場合には，base ring として CC などを指定します。QQbar は $\sqrt{2}$ や $\sqrt{3}$ などを含む体の事です[15]。

行列の持っているメソッド base_ring() を使って，行列の base ring として何が指定されているのか確かめることができます：

```
1 mat = matrix(2,2,[1,2,3,4])       # 行列を定義
2 print mat.base_ring()              # matのbase ringを表示
```

実行結果の例

```
Integer Ring
```

上の行列の base ring は整数環 $\mathbb{Z}$ になっています。成分に一つでも有理数があると base ring は自動的に有理数体になります：

```
1 mat = matrix(2,2,[1/2,2,3,4])     # 行列の要素に有理数1/2がある
2 print mat.base_ring()
```

実行結果の例

```
Rational Field
```

base ring をはじめから指定して行列を定義したい場合には，matrix の直後に書いて指定します。たとえば base ring を SR (symbolic ring=形式的な和・積・スカラー倍が定義された環) にしたい場合は次のようにします：

```
1 mat = matrix(SR, [[1,2],[3,4]])       # 行列をSR上で定義
2 print mat.base_ring()
```

実行結果の例

```
Symbolic Ring
```

行列を定義したあとで base ring を変える場合は a を行列として a = matrix(SR, a) とします：

```
1 a = matrix([[1,2],[3,4]])      # 行列aを定義
2 print a.base_ring()            # 行列aのbase ringをプリント
3 a = matrix(SR, a)              # aのbase ringをSRに変更
4 print a.base_ring()            # 行列aのbase ringをプリント
```

[15] QQbar 同士の算術計算は厳密なのですが，QQbar を base ring とする行列の固有値の計算では近似計算になるようです

実行結果の例

```
Integer Ring
Symbolic Ring
```

49.5 行列に対する命令

行列に対して実行できる演算には次のようなものがあります：

名前	Sage での記号	数学的意味	補足
固有多項式 (特性多項式)	`A.charpoly()`	$\det(xI - A)$	多項式の変数は `x`
固有値	`A.eigenvalues()`	$\det(\lambda I - A) = 0$ の解 λ	
固有ベクトル	`A.eigenvectors_right()`	$Av = \lambda v$ を満たす v	
最小多項式	`A.minimal_polynomial()`		多項式の変数は `x`
指数関数	`A.exp()`	e^A	
対角化	`A.eigenmatrix_right`	$P^{-1}AP$ は対角行列	`P^-1AP` と `P` の組

49.6 固有値・固有ベクトル

正方行列 A があるベクトル v と定数 λ に対して

$$Av = \lambda v, \quad (v \neq 0)$$

を満たすとき，λ を固有値，v を対応する固有ベクトルといいます。上の方程式を $(A - \lambda I)v = 0$ と書き直すと，これが非自明な解を持つための必要十分条件は $\det(A - \lambda I) = 0$ であることがわかります。つまり多項式 $\det(A - \lambda I) = 0$ の解が固有値です。Sage で固有値をする場合，`base_ring` の設定によって答えの表示が異なります。それでは，具体的に行列

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

の固有値を計算してみましょう。この行列の固有値多項式は $f(\lambda) := (\lambda - 1)(\lambda - 4) - 6$ なので $f(\lambda) = 0$ の解は $(5 \pm \sqrt{33})/2$ です。これを Sage で計算します。

```
mat = matrix(SR, 2,2,[1,2,3,4])     #base_ringはSRに設定
print mat.eigenvalues()             # 固有値のリストをプリント
```

実行結果の例

```
[-1/2*sqrt(33) + 5/2, 1/2*sqrt(33) + 5/2]
```

一つ目の結果では厳密な値が表示されました。しかし，固有多項式が具体的に解けない場合には，固有値は表示されません。たとえば，

```
mat5 = matrix(SR, 5,5, lambda i,j : 1/(i+j+1)) # 環をSRとして行列を定義
print mat5                                     # 行列をプリント
print mat5.eigenvalues()                       # 固有値をプリント
```

と行列を定義してその固有値を計算させたとしても，実行結果は

実行結果の例

```
[  1 1/2 1/3 1/4 1/5]
[1/2 1/3 1/4 1/5 1/6]
[1/3 1/4 1/5 1/6 1/7]
[1/4 1/5 1/6 1/7 1/8]
[1/5 1/6 1/7 1/8 1/9]
Traceback (click to the left of this block for traceback)
...
ArithmeticError: could not determine eigenvalues exactly using symbolic
matrices; try using a different type of matrix via self.change_ring(),
if possible
```

となって，固有値を具体的に決定できなかったことが分かります[16]。この行列の固有値は 5 次方程式であり，一般解を具体的に表すことはできません。しかし，数値的には固有値を求めることはできます[17]。

そこで，例えば上で作った 2×2 行列の係数体を CDF にして計算します：

```
mat = matrix(CDF,mat)     # base_ringをCDFに変更
print mat.eigenvalues()
```

実行結果の例

```
[-0.372281323269, 5.37228132327]
```

固有値の数値のリストが表示されました。はじめから数値結果だけ得られればよい場合には SR を使わずに RDF や CDF を使いましょう。そのほうが SR で固有値を計算するよりもずっと高速に計算できます。

固有ベクトルは行列に対するメソッド eigenvectors_right() で求めます。

```
mat2 = matrix(SR, 2,2,[1,2,3,4])     # base_ringはSRに設定
eigv = mat2.eigenvectors_right()      # 固有ベクトルの組を定義
print eigv
```

実行結果の例

```
[(-1/2*sqrt(33) + 5/2, [(1, -1/4*sqrt(33) + 3/4)], 1), (1/2*sqrt(33) +
5/2, [(1, 1/4*sqrt(33) + 3/4)], 1)]
```

これは，次を意味しています：

[（第 1 固有値,[対応する固有ベクトル], 多重度),（第 2 固有値,[対応する固有ベクトル], 多重度）]

なので，少し面倒ですが，固有ベクトルを取り出すには，上のプログラムに続けて次のようにします。

```
print eigv[0][1][0]     # 一つ目の固有ベクトル
print eigv[1][1][0]     # 二つ目の固有ベクトル
```

実行結果の例

```
(1, -1/4*sqrt(33) + 3/4)
(1, 1/4*sqrt(33) + 3/4)
```

49.7　対角化とその応用

49.7.1　行列の対角化

行列の対角化は，行列の冪や指数関数を計算するために必要な手順の一つです。行列 A を対角化するとは，ある正則行列 P を見つけて

$$P^{-1}AP = \mathrm{diag}[\lambda_1, \cdots, \lambda_n] : \text{対角行列}$$

を計算することです。通常，対角化行列 P を作るためには，A の固有ベクトルを計算すればよいのですが，Sage では，eigenmatrix_right() で対角化を簡単に行うことができます。

行列 $\begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix}$ を対角化してみましょう。

```
A = matrix(SR, [[1,4],[2,3]] )      # 行列の定義
A.eigenmatrix_right()               # 固有値と対角化行列を求める
```

実行結果の例

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

[16] 5 × 5 行列であるところを 4 × 4 行列にすると厳密な固有値が得られます。
[17] もちろん結果は誤差を含む近似値になりますが。

これは A の固有値が $5, -1$ で

$$P^{-1}AP = \begin{pmatrix} 5 & 0 \\ 0 & -1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 1 \\ 1 & -1/2 \end{pmatrix}$$

となることを意味しています。実際

$$\begin{pmatrix} 1 & 1 \\ 1 & -1/2 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1/2 \end{pmatrix} = \begin{pmatrix} 1/3 & 2/3 \\ 2/3 & -2/3 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1/2 \end{pmatrix} = \begin{pmatrix} 5 & 0 \\ 0 & -1 \end{pmatrix}$$

となり，ちゃんと対角化されているのが分かります。

49.7.2 行列の指数関数と線形微分方程式

行列 A に対して，その指数関数 $\exp(tA)$ を

$$\exp(tA) = \sum_{n=0}^{\infty} \frac{t^n}{n!} A^n, \qquad t \in \mathbb{C} \tag{30}$$

によって定義します。Sage で行列の指数関数を計算することができますが，以下ではこれを応用して微分方程式を解きます。

最も簡単な例に対して，具体的な計算を行ってみましょう。ばねにつながれた質量 m からなる 1 次元の力学系を考えます。ばねのつり合いの位置を原点としましょう。このとき運動方程式から

$$m\frac{d^2x(t)}{dt^2} = -kx(t)$$

が成り立ちます。ここで k はバネ定数です。粒子は時刻 0 に位置 X にいて，速度 V であるとします。ここでの目標は時刻 t での位置 $x(t)$ を求めることです。この微分方程式は定数変化法などにより解くことができますが，いまは行列を用いて解いてみます。まず方程式を線形化します。$\sqrt{k/m} = w$ と置きます。$v(t) = dx(t)/dt$ とすると上の微分方程式は

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ -w^2x(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -w^2 & 0 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$

と同値です。つまり

$$A = \begin{pmatrix} 0 & 1 \\ -w^2 & 0 \end{pmatrix}$$

と置けば，

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = A \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$

です，これはベクトル $(x(t), v(t))^t$ を一回微分するごとに行列 A が前に出てくることを意味しているので，$(x(0), v(0)) = (X, V)$ を思い出すと

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \exp(tA) \begin{pmatrix} X \\ V \end{pmatrix}$$

となります。微分方程式を解く問題は行列の指数関数 $\exp(tA)$ を求める問題に置き換わりました。Sage で行列 T の指数関数 $\exp(T)$ を求める命令が `T.exp()` です。具体的に Sage に行列を計算させて微分方程式を解いてみましょう：

```
1 var('t w')                          # t wを変数とする
2 A = matrix([[0,1],[-w^2,0]])        # 行列Aを定義する
3 B = t*A                             # BをtAとする
4 expo = B.exp()                      # expoをexp(B)とする
```

これで，$B = \exp(tA)$ が求まりました。つぎに

```
var("X V")              # X, Vを変数とする
y = vector([X,V])       # y をベクトル(X,V)とする
sol = expo*y            # sol をexp(tA) を(X,V)に作用させたものとする
```

とします。最後得られた量 sol[0] は $(x(t), v(t))$ なのでその最初の成分（第 0 成分）が時刻 t の位置を与えます。つまり微分方程式の解 $x(t)$ は sol[0] です。得られる関数は実なので実部を取って（これは何の変化ももたらしません）full_simplify を用いて整理します。上のプログラムに続けて

```
print real_part(sol[0]).full_simplify()
```

と入力すると最終的な答え

```
(X*w*cos(t*w) + V*sin(t*w))/w
```

が得られます。これが微分方程式の解 $x(t)$ です。通常の数学の記法で書けば，微分方程式の解は

$$x(t) = X\cos(wt) + \frac{V}{W}\sin(wt),$$
$$x(0) = X, \quad x'(0) = V$$

となるという事を意味しています。

49.7.3 行列の対角化とその応用：フィボナッチ数列の一般項

フィボナッチ数列 $a_1 = 1, a_2 = 1,$

$$a_{n+2} = a_{n+1} + a_n, \qquad n = 1, 2, 3, \cdots \tag{31}$$

を考えます。まず，上の関係式は

$$\begin{pmatrix} a_{n+2} \\ a_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{n+1} \\ a_n \end{pmatrix}$$

となる事に注意します。右辺の行列を A とします。すると

$$\begin{pmatrix} a_{n+1} \\ a_n \end{pmatrix} = A \begin{pmatrix} a_n \\ a_{n-1} \end{pmatrix} = A^2 \begin{pmatrix} a_{n-1} \\ a_{n-2} \end{pmatrix} = \cdots = A^{n-1} \begin{pmatrix} a_2 \\ a_1 \end{pmatrix} = A^{n-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{32}$$

なので A^n が具体的に計算できれば，数列の一般項 a_n が求まる事になります。A^n を求めるために，A を対角化してみましょう。

```
A = matrix( SR, [ [1,1],[1,0] ])
sol = A.eigenmatrix_right()
P = sol[1]
D = P^(-1)*A*P
print 'P = ', P,  ''
print 'D = P^(-1)AP ='
print D.expand()
```

実行結果の例

```
P =
[                 1                  1]
[-1/2*sqrt(5) - 1/2  1/2*sqrt(5) - 1/2]

D = P^(-1)AP =
[-1/2*sqrt(5) + 1/2                  0]
[                 0  1/2*sqrt(5) + 1/2]
```

ここで D を単純化するために expand() を行いました。この事から A を対角化する行列は

$$P = \begin{pmatrix} 1 & 1 \\ -(\sqrt{5}+1)/2 & (\sqrt{5}-1)/2 \end{pmatrix}$$

であり，$P^{-1}AP = \mathrm{diag}[(-\sqrt{5}+1)/2, (\sqrt{5}+1)/2]$ となることが分かります。この式の両辺を n 乗すると

$$\begin{pmatrix} (\frac{-1+\sqrt{5}}{2})^n & 0 \\ 0 & (\frac{\sqrt{5}+1}{2})^n \end{pmatrix} = P^{-1}A^nP$$

となります。したがって A^n は

$$A^n = P\begin{pmatrix} (\frac{-1+\sqrt{5}}{2})^n & 0 \\ 0 & (\frac{\sqrt{5}+1}{2})^n \end{pmatrix} P^{-1} \tag{33}$$

と計算する事ができます。これをつかって A^{n-1} を計算させてみましょう：

```
var('n')
Dn = matrix([ [D[0][0]^(n-1), 0], [0, D[1][1]^(n-1)]])    # Dのn-1乗を作る
An = P * Dn * P^(-1)                    # これがAのn-1乗
print expand(An)                        # Anを少し単純化してプリント
```

実行結果の例

```
[ 1/5*sqrt(5)*(1/2*sqrt(5) + 1/2)^n/(sqrt(5) + 1) +
1/5*sqrt(5)*(-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1) + (1/2*sqrt(5) +
1/2)^n/(sqrt(5) + 1) - (-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1)
2/5*sqrt(5)*(1/2*sqrt(5) + 1/2)^n/(sqrt(5) + 1) +
2/5*sqrt(5)*(-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1)]
[
2/5*sqrt(5)*(1/2*sqrt(5) + 1/2)^n/(sqrt(5) + 1) +
2/5*sqrt(5)*(-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1)
-1/5*sqrt(5)*(1/2*sqrt(5) + 1/2)^n/(sqrt(5) + 1) -
1/5*sqrt(5)*(-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1) + (1/2*sqrt(5) +
1/2)^n/(sqrt(5) + 1) - (-1/2*sqrt(5) + 1/2)^n/(sqrt(5) - 1)]
```

D^{n-1}, A^{n-1} を Dn, An と定義しました。複雑ですが正しく計算できています。上の行列にベクトル

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{34}$$

を掛けて得られるベクトルの第 2 成分が一般項 a_n となります。上のプログラムに続けて次を実行します。

```
assume(n,'integer')                     # nを整数と仮定する
v = vector([1,1])       # ベクトル(1,1)をvと定義
aa = An * v               # A^nをvに掛けると
aa[1].simplify_full()    # その第2成分が一般項となる
```

実行結果の例

```
-1/5*((sqrt(5) - 1)^n*(-1)^n - (sqrt(5) + 1)^n)*sqrt(5)/2^n
```

最後の結果が，フィボナッチ数列の一般項 a_n となります。これを整理すれば，

$$a_n = -\frac{\left(5^{\frac{1}{2}n}(-1)^n\left(\sqrt{5}-1\right)^n - \left(\sqrt{5}+5\right)^n\right)5^{-\frac{1}{2}n-\frac{1}{2}}}{2^n} = \frac{1}{\sqrt{5}}\left\{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right\}$$

となります。

実際，最後の出力の結果を n=1,2,... に対して表示させると

```
def fib(n):
    return -1/5*((sqrt(5) - 1)^n*(-1)^n - (sqrt(5) + 1)^n)*sqrt(5)/2^n

for k in range(1,20):
    print fib(k).simplify_full()
```

実行結果の例

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

となり，フィボナッチ数列に一致していることがわかります。

フィボナッチ数列に自然数しか現れないのに，その一般項を表すには平方根が必要であるというのは面白い事実です。

49.8 ジョルダン標準形

Sage でジョルダン標準形を求めるには `jordan_form` をつかいます。次の行列

$$\begin{pmatrix} 3 & 2 & -3 \\ -2 & -1 & 1 \\ 5 & 3 & -5 \end{pmatrix}$$

の Jordan 標準形を求めるには次のようにします：

```
A = matrix(QQ, [[3,2,-3],[-2,-1,1],[5,3,-5]])
A.jordan_form(transformation=True)
```

実行結果の例

```
(
[-1  1  0]  [-3  4  1]
[ 0 -1  1]  [-3 -2  0]
[ 0  0 -1], [-6  5  0]
)
```

これは，

$$P = \begin{pmatrix} -3 & 4 & 1 \\ -3 & -2 & 0 \\ -6 & 5 & 0 \end{pmatrix} \quad J = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} = P^{-1}AP$$

となることを意味しています。A の Jordan 標準形が J で変換行列する為の行列が P です。

49.9 練習問題

49.9.1 大きな行列の固有値のグラフの表示

$N = 20$ と実数 g に対して行列

$$Q^{+}(g) := \begin{pmatrix} 1 & \sqrt{1}g & 0 & 0 & 0 & 0 & 0 & 0 \\ \sqrt{1}g & 0 & \sqrt{2}g & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2}g & 3 & \sqrt{3}g & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3}g & 2 & \sqrt{4}g & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2}g & 5 & \sqrt{5}g & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5}g & 4 & \ddots & \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \sqrt{N}g \\ 0 & 0 & 0 & 0 & 0 & & \sqrt{N}g & N + (-1)^N \end{pmatrix} + g^2 I$$

$$Q^{-}(g) := \begin{pmatrix} -1 & g & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 2 & \sqrt{2}g & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2}g & 1 & \sqrt{3}g & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3}g & 4 & \sqrt{4}g & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{4}g & 3 & \sqrt{5}g & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5}g & 6 & \ddots & \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \sqrt{N}g \\ 0 & 0 & 0 & 0 & 0 & & \sqrt{N}g & N - (-1)^N \end{pmatrix} + g^2 I$$

を定義する。I は単位行列。$Q^{\pm}(g)$ の固有値を小さい順からそれぞれ $\lambda_1^{\pm}(g), \lambda_2^{\pm}(g), \lambda_3^{\pm}(g), \cdots, \lambda_{N+1}^{\pm}(g)$ とする。固有値のリスト

$$
\begin{aligned}
L1 &= (\lambda_1^{+}(j/30))_{j=-90}^{90}, \\
L2 &= (\lambda_2^{+}(j/30))_{j=-90}^{90}, \\
L3 &= (\lambda_3^{+}(j/30))_{j=-90}^{90}, \\
L4 &= (\lambda_4^{+}(j/30))_{j=-90}^{90}, \\
L5 &= (\lambda_5^{+}(j/30))_{j=-90}^{90}, \\
K1 &= (\lambda_1^{-}(j/30))_{j=-90}^{90}, \\
K2 &= (\lambda_2^{-}(j/30))_{j=-90}^{90}, \\
K3 &= (\lambda_3^{-}(j/30))_{j=-90}^{90}, \\
K4 &= (\lambda_4^{-}(j/30))_{j=-90}^{90}, \\
K5 &= (\lambda_5^{-}(j/30))_{j=-90}^{90}
\end{aligned}
$$

を作り `list_plot` でそれぞれのリストをプロットせよ。グラフはオプション `plotjoined=True` により隣り合う点同士を線で繋ぐこと。すべてのグラフを重ねて一つのグラフにせよ（固有値ごとに色を変えておくとよい）[*18]。

49.9.2 ランダム行列の固有値の分布

ランダムな数値を要素に持つ行列をランダム行列という。ランダム行列は，原子核の研究から現れたものであるが，最近になってその数学的研究は飛躍的に発展した。また，ランダムなポテンシャルを持つシュレディンガー作用素の固有値や固有ベクトルの挙動は絶縁体のメカニズムの数学的な説明を与える。

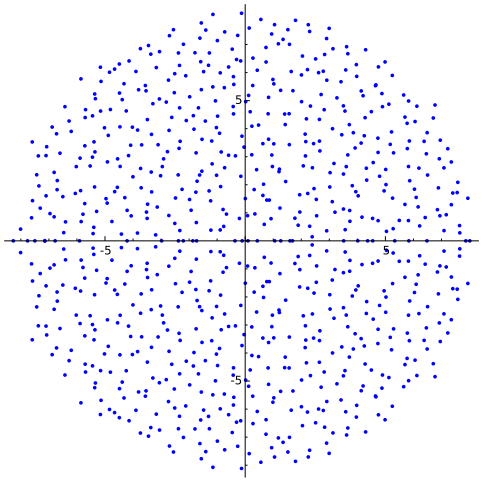
_____ *** _____

問題 $-1/2$ から $1/2$ までのランダムな数を成分に持つ行列に持つ行列を作り，その固有値を複素平面上にプロットせよ。行列のサイズは 800×800 とし，縦横比は 1(`aspect_ratio=1`) とする[*19]。

そのような行列の固有値はある半径の円の中に均一に分布する事が知られている (circular law)。わかりやすい現象だが，これを数学的に証明するのは難しい。

[*18] 注：このような行列は 2 準位を持つ原子とレーザー光との相互作用を表すモデルのハミルトニアンを記述しており Rabi モデルと呼ばれる。対応する固有値はその量子系のエネルギー準位を表している。

[*19] 800×800 のサイズの行列の計算ではエラーが起こることがあるが，そのような場合には，400×400 のように行列のサイズを減らして計算してください。



ちなみに，Sage では `random()` で $(0,1)$ の中の数をランダムに返す：

```
for i in range(4):
    print random(),
```

実行結果の例

```
0.329694823321 0.831787904723 0.214856702912 0.615524329254
```

したがって，`random()-1/2` で $(-1/2,1/2)$ の一様な乱数を得ることができる。

```
for i in range(4):
    print random() - 1/2,
```

実行結果の例

```
-0.0357016806704 0.0999712876727 -0.169974222529 0.39728018763
```

行列の base ring は何も指定しないか (その場合は `RDF`)，もしくは `CDF` にするとよい。グラフの描画は `list_plot` を使うのがよい。