MAT 305: Mathematical Computing

Linear algebra

John Perry

University of Southern Mississippi

Fall 2011

Outline

Outline

Vectors

`vector(`*ring*`,` *entries*`)` where

- *ring* is base ring of *entries* (a list)
- default ring: $\mathbb{Z}$

Vectors

`vector(`*ring*`,` *entries*`)` where

- *ring* is base ring of *entries* (a list)
- default ring: $\mathbb{Z}$

Example

```
sage:  u = vector([0, 2, 2, 0])
sage:  v = vector([1, 3, -1, 2])
sage:  u + v
(1, 5, 1, 2)
sage:  u*v
4
sage:  u.norm()
2*sqrt(2)
```

Dot product!

You can plot vectors!

`v.plot()`, with optional arguments:

- `plot_type`: `'arrow'`, `'point'`, `'step'`
- `start`: tuple, list, or vector

You can plot vectors!

`v.plot()`, with optional arguments:

- `plot_type`: `'arrow'`, `'point'`, `'step'`
- `start`: tuple, list, or vector

Example

Illustration of vector arithmetic:

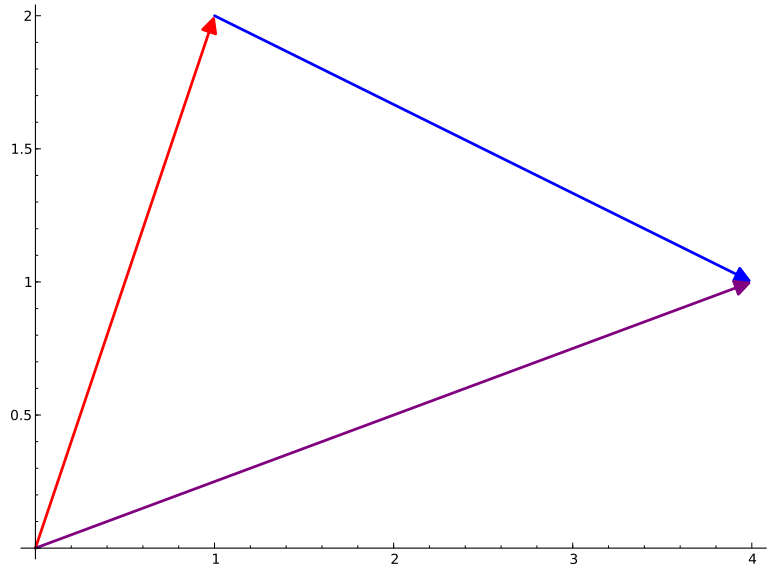
```
sage:  u = vector([1,2])
sage:  v = vector([3,-1])
sage:  u.plot(color='red')
         + v.plot(color='blue',start=u)
         + (u+v).plot(color='purple')
```

You can plot vectors!

Example

Illustration of vector arithmetic:

```
sage:  u = vector([1,2])
sage:  v = vector([3,-1])
sage:  u.plot(color='red')
         + v.plot(color='blue',start=u)
         + (u+v).plot(color='purple')
```



Outline

The `matrix()` command

`matrix(` *ring*, *#rows*, *#cols*, entries`)` where

- *ring* (optional) an algebraic ring
  $\mathbb{N}$, $\mathbb{Z}$ (default), $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, symbolic ring, ...
- *#rows*, *#cols* (optional) number of rows and columns
  (default depends on *entries*; no *entries* $\Longrightarrow$ 0×0 matrix)
- *entries* (optional) is one of
  - a list of entries, from northwest corner to southeast
    (if *#rows*, *#cols* specified)
  - a list of row vectors
  - none specified? all entries 0

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
```

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
sage:  MR = matrix(RR,[[1,2,3],[3,2,1],[1,1,2]])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
```

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
sage:  MR = matrix(RR,[[1,2,3],[3,2,1],[1,1,2]])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
sage:  MS = matrix(SR,[[x**2 + 1, 0, 0],
       [x + I, 1, 0]])
sage:  MS
[x^2 + 1       0       0]
[  x + I       1       0]
```

Help yourself read

Good idea to put rows in different lines

```
sage:  MR = matrix(RR, [
       [1,2,3],
       [3,2,1],
       [1,1,2]
     ])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
```

Accessing matrix entries

Matrix a list of lists $\Longrightarrow$ `M[i,j]` $= M_{i,j}$

Accessing matrix entries

Matrix a list of lists $\Longrightarrow$ `M[i,j]` $= M_{i,j}$

Example

```
sage:  MS[1,0]
x+I
sage:  MS[0,2] = x - I
sage:  MS
[x^2 + 1       0       x - I]
[  x + I       1           0]
```

(counting starts from 0)

Submatrices

- `M.submatrix(`i`,` j`,` m`,` n`)` gives
  - $m \times n$ submatrix of M
  - whose northwest corner is in row i, column j
- `M.augment(`A`)` gives $(M|A)$

Submatrices

- `M.submatrix(`i`,` j`,` m`,` n`)` gives
  - $m \times n$ submatrix of M
  - whose northwest corner is in row i, column j
- `M.augment(`A`)` gives $(M|A)$

Example

```
sage:  MZ[1,1] = 1
sage:  MZ.submatrix(1,1,2,2)
[ 1 0]
[ 0 0]
```

Basic matrix operations

"dot" command	mathematics
`M.det()`	determinant
`M.inverse()`	
`M.transpose()`	
`M.eigenvalues()`	
`M.eigenvectors_right()`	right eigenvectors*
`M.eigenvectors_left()`	left eigenvectors*
`M.echelon_form()`	echelon formof unchanged M
`M.echelonize()`	change M to echelon form
`M.ncols()`	number of columns
`M.nrows()`	number of rows

*"right eigenvectors" are usual "eigenvectors"

Row arithmetic

"dot" command	mathematics
`M.set_row_to_multiple_of_row(i,j,a)`	set row i to a times row j*
`M.add_multiple_of_row(i,j,a)`	add a times row j to row i*
`M.swap_rows(i,j)`	swap rows i, j
`M.swap_columns(i,j)`	swap columns i, j

*row i changes; row j remains the same

Example: find inverse of matrix

Sage has a `.inverse()` command, but suppose you want to see steps. . . ?

Example: find inverse of matrix

Sage has a `.inverse()` command, but suppose you want to see steps...?

Algorithm from High School Algebra II!

algorithm Compute inverse

inputs

M, an invertible matrix over a field

outputs

M^{-1}

do

Let $n = \dim(M)$

Let A be augmented matrix $(M \mid I_n)$

Triangularize A

return rightmost $n \times n$ submatrix of A

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment `MZ` *by* I_4

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment `MZ` *by* I_4

To create I_4*, can set diagonal entries of zero matrix to 1...*

```
sage: I4 = matrix(4,4)
sage: for i in range(4):
          I4[i,i] = 1
sage: I4
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment `MZ` *by* I_4

. . . or use `identity_matrix()` *command*

```
sage:  I4 = identity_matrix(4)
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment `MZ` *by* I_4

. . . or use `identity_matrix()` *command*

```
sage:  I4 = identity_matrix(4)
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
sage:  A = MZ.augment(I4)
sage:  A
[1 2 3 4 1 0 0 0]
[0 2 2 3 0 1 0 0]
[8 3 1 2 0 0 1 0]
[0 1 2 3 0 0 0 1]
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

```
sage: A.add_multiple_of_row(2,0,-8)
sage: A
[   1    2    3    4    1    0    0    0]
[   0    2    2    3    0    1    0    0]
[   0  -13  -23  -30   -8    0    1    0]
[   0    1    2    3    0    0    0    1]
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

```
sage: A.add_multiple_of_row(2,0,-8)
sage: A
[   1    2    3    4    1    0    0    0]
[   0    2    2    3    0    1    0    0]
[   0  -13  -23  -30   -8    0    1    0]
[   0    1    2    3    0    0    0    1]
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

```
sage: A.swap_rows(1,3)
sage: A.add_multiple_of_row(0,1,-2)
sage: A.add_multiple_of_row(2,1,13)
sage: A.add_multiple_of_row(3,1,-2)
sage: A
[  1   0  -1  -2   1   0   0  -2]
[  0   1   2   3   0   0   0   1]
[  0   0   3   9  -8   0   1  13]
[  0   0  -2  -3   0   1   0  -2]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

```
sage:  A.swap_rows(1,3)
sage:  A.add_multiple_of_row(0,1,-2)
sage:  A.add_multiple_of_row(2,1,13)
sage:  A.add_multiple_of_row(3,1,-2)
sage:  A
[   1    0   -1   -2    1    0    0   -2]
[   0    1    2    3    0    0    0    1]
[   0    0    3    9   -8    0    1   13]
[   0    0   -2   -3    0    1    0   -2]
```

Third column: need pivot multiply row 3 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage:  A.set_row_to_multiple_of_row(2,2,1/3)
...
TypeError:  Multiplying row by Rational Field
element cannot be done over Integer Ring, use
change_ring or with_row_set_to_multiple_of_row
instead.
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage:  A.set_row_to_multiple_of_row(2,2,1/3)
...
TypeError:  Multiplying row by Rational Field
element cannot be done over Integer Ring, use
change_ring or with_row_set_to_multiple_of_row
instead.
```

Uh-oh! No multiplicative inverses in default ring! ($\mathbb{Z}$)
Change to $\mathbb{Q}$ and proceed.

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Uh-oh! No multiplicative inverses in default ring! ($\mathbb{Z}$)
Change to $\mathbb{Q}$ and proceed.

```
sage:  A = A.change_ring(QQ)
sage:  A
[   1    0   -1   -2    1    0    0   -2]
[   0    1    2    3    0    0    0    1]
[   0    0    3    9   -8    0    1   13]
[   0    0   -2   -3    0    1    0   -2]
```

Looks the same, but it's not.
Return to regularly-scheduled programming.

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage: A.set_row_to_multiple_of_row(2,2,1/3)
sage: A
[    1     0    -1    -2     1     0     0    -2]
[    0     1     2     3     0     0     0     1]
[    0     0     1     3 -8/3     0   1/3 13/3]
[    0     0    -2    -3     0     1     0    -2]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage:  A.set_row_to_multiple_of_row(2,2,1/3)
sage:  A
[    1     0    -1    -2     1     0     0    -2]
[    0     1     2     3     0     0     0     1]
[    0     0     1     3  -8/3     0   1/3  13/3]
[    0     0    -2    -3     0     1     0    -2]
```

Third column: eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,2,1)
sage:  A.add_multiple_of_row(1,2,-2)
sage:  A.add_multiple_of_row(3,2,2)
sage:  A
[     1      0      0      1  -5/3      0   1/3   7/3]
[     0      1      0     -3   16/3      0  -2/3 -23/3]
[     0      0      1      3  -8/3      0   1/3  13/3]
[     0      0      0      3 -16/3      1   2/3  20/3]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,2,1)
sage:  A.add_multiple_of_row(1,2,-2)
sage:  A.add_multiple_of_row(3,2,2)
sage:  A
[     1      0      0      1  -5/3      0   1/3   7/3]
[     0      1      0     -3   16/3      0  -2/3 -23/3]
[     0      0      1      3  -8/3      0   1/3  13/3]
[     0      0      0      3 -16/3      1   2/3  20/3]
```

Fourth column: need pivot
multiply row 4 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

```
sage:  A.set_row_to_multiple_of_row(3,3,1/3)
sage:  A
[     1      0      0      1  -5/3      0   1/3   7/3]
[     0      1      0     -3  16/3      0  -2/3 -23/3]
[     0      0      1      3  -8/3      0   1/3  13/3]
[     0      0      0      1 -16/9    1/3   2/9  20/9]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

```
sage:  A.set_row_to_multiple_of_row(3,3,1/3)
sage:  A
[     1      0      0      1  -5/3      0   1/3   7/3]
[     0      1      0     -3  16/3      0  -2/3 -23/3]
[     0      0      1      3  -8/3      0   1/3  13/3]
[     0      0      0      1 -16/9    1/3   2/9  20/9]
```

Fourth column: eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,3,-1)
sage:  A.add_multiple_of_row(1,3,3)
sage:  A.add_multiple_of_row(2,3,-3)
sage:  A
[     1      0      0      0    1/9  -1/3    1/9    1/9]
[     0      1      0      0   16/3     1      0     -1]
[     0      0      1      0   -8/3    -1   -1/3   -7/3]
[     0      0      0      1 -16/9   1/3    2/9   20/9]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,3,-1)
sage:  A.add_multiple_of_row(1,3,3)
sage:  A.add_multiple_of_row(2,3,-3)
sage:  A
[     1      0      0      0    1/9  -1/3    1/9    1/9]
[     0      1      0      0   16/3     1      0     -1]
[     0      0      1      0   -8/3    -1   -1/3   -7/3]
[     0      0      0      1 -16/9   1/3    2/9   20/9]
```

Have inverse! extract, test

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Have inverse! extract, test

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Have inverse! extract, test

```
sage:  Minv = A.submatrix(0,4,4,4)
sage:  Minv * M
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Other tools

Need another computation w/M? Remember:

- `M.<tab>` states all tools for M
- `M.<command>?` states help for `command`
- `M.<command>??` lists source code for `command`

Outline

Eigenvectors and eigenvalues

An **eigenvector** $\mathbf{x}$ of a matrix M with **eigenvalue** λ satisfies

$$M\mathbf{x} = \lambda\mathbf{x}$$

Eigenvectors and eigenvalues

An **eigenvector** $\mathbf{x}$ of a matrix M with **eigenvalue** λ satisfies

$$M\mathbf{x} = \lambda\mathbf{x}$$

Example

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

verification in Sage:

```
sage:  M = matrix(2,2,[0,2,2,0])
sage:  v = vector([1,-1])
sage:  M*v
(-2, 2)
```

Easy to find in Sage

```
sage:  M = matrix(2,2,[0,2,2,0])
sage:  M.eigenvectors_left()
[(2, [ (1, 1) ], 1), (-2, [ (1, -1) ], 1)]
```

What does this tell us?

- $\mathbf{e}_1 = (1,1)$ is eigenvec w/eigenval 2, mult 1
- $\mathbf{e}_2 = (1,-1)$ is eigenvec, w/eigenval -2, mult 1

Easy to find in Sage

```
sage:  M = matrix(2,2,[0,2,2,0])
sage:  M.eigenvectors_left()
[(2, [ (1, 1) ], 1), (-2, [ (1, -1) ], 1)]
```

What does this tell us?

- $\mathbf{e}_1 = (1,1)$ is eigenvec w/eigenval 2, mult 1
- $\mathbf{e}_2 = (1,-1)$ is eigenvec, w/eigenval -2, mult 1

In other words,

- $M\mathbf{e}_1 = 2\mathbf{e}_1$
- $M\mathbf{e}_2 = -2\mathbf{e}_2$

Try verifying this in Sage

Neat fact of eigenvectors

Theorem (Eigendecomposition)

Let M be an $n \times n$ matrix with

- *independent eigenvectors* $\mathbf{e}_1, \ldots, \mathbf{e}_n$
- *corresponding to eigenvalues* $\lambda_1, \ldots, \lambda_n$.

We can rewrite M as $M = Q\Lambda Q^{-1}$ where

$$Q = (\mathbf{e}_1 | \mathbf{e}_2 | \cdots | \mathbf{e}_n) \qquad \Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}.$$

Example

With M as defined,

$$Q = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \Lambda = \begin{pmatrix} 2 & \\ & -2 \end{pmatrix}$$

Verify in Sage that $M = Q\Lambda Q^{-1}$

Example

With M as defined,

$$Q = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \Lambda = \begin{pmatrix} 2 & \\ & -2 \end{pmatrix}$$

Verify in Sage that $M = Q\Lambda Q^{-1}$

```
sage:  Q = matrix(2,2,[1,1,1,-1])
sage:  L = matrix(2,2,[2,0,0,-2])
sage:  Q*L*Q**(-1)
[0 2]
[2 0]
```

$\ldots$ recall $M = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$

But how is this useful?

Consider the numbers

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

But how is this useful?

Consider the numbers

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

This is the well-known Fibonacci sequence:

$$f_1 = 1 \quad f_2 = 1 \quad f_n = f_{n-1} + f_{n-2}$$

Can we get a "non-recursive" formula?

Fibonacci matrix

As a matrix equation,

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix} = \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$$

Let's try rewriting the matrix

$$F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Fibonacci matrix

As a matrix equation,

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix} = \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$$

Let's try rewriting the matrix

$$F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Iterative multiplication generates the sequence

```
sage:  F = matrix(2,2,[1,1,1,0])
sage:  f12 = vector([1,1])
sage:  F*f12
[2, 1]
sage:  F^2*f12
[3, 2]
sage:  F^3*f12
[5, 3]
```

...

In short,

$$F^{n-2}\begin{pmatrix} f_2 \\ f_1 \end{pmatrix} = \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$$

and

$$\begin{aligned} F^n &= \left(Q\Lambda Q^{-1}\right)^n \\ &= \underbrace{\left(Q\Lambda Q^{-1}\right)\left(Q\Lambda Q^{-1}\right)\cdots\left(Q\Lambda Q^{-1}\right)}_{n} \\ &= \underbrace{Q\Lambda\left(Q^{-1}Q\right)\Lambda\left(Q^{-1}Q\right)\cdots\left(Q^{-1}Q\right)\Lambda Q^{-1}}_{n} \\ &= Q\Lambda^n Q^{-1} \end{aligned}$$

Since Λ is diagonal, it is easy to compute Λ^n

What to do?

General outline:

- Compute eigenvectors and eigenvalues

  ```
  sage:  F.eigenvectors_right()
  ```

- Construct $Q\Lambda^n Q^{-1}$

  ```
  sage:  Q = matrix(2,2,[...])
  sage:  L = matrix(2,2,[...])
  ```

- Analyze the equation

One "drawback"

- eigenvectors, eigenvalues look inexact

```
sage:  F.eigenvectors_right()
[(-0.618033988749895?,
    [(1, -1.618033988749895?)], 1),
 (1.618033988749895?,
    [(1, 0.618033988749895?)], 1)]
```

One "drawback"

- eigenvectors, eigenvalues look inexact

```
sage:  F.eigenvectors_right()
[(-0.618033988749895?,
    [(1, -1.618033988749895?)], 1),
 (1.618033988749895?,
    [(1, 0.618033988749895?)], 1)]
```

- In fact, we can determine their exact values

```
sage:  F = F.change_ring(SR)
sage:  F.eigenvectors_right()
[(-1/2*sqrt(5) + 1/2,
    [(1, -1/2*sqrt(5) - 1/2)], 1),
 (1/2*sqrt(5) + 1/2,
    [(1, 1/2*sqrt(5) - 1/2)], 1)]
```

Put it together

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

```
sage:  Q = matrix(
           [1, -1/2*sqrt(5) - 1/2],
           [1,1/2*sqrt(5) - 1/2]
         )
sage:  var('n')
sage:  L = matrix(2,2,[
           (-1/2*sqrt(5) + 1/2)^(n-2), 0,
           0, (1/2*sqrt(5) + 1/2)^(n-2)
         ])
sage:  Q*L*Q**(-1)
```

. . . very unpleasant

. . . or is it?

With an "algebraic massage" (omitted), the matrix product tells us that

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

To get this:

- need to work with elements of matrix
- rationalize denominators
- notice change in exponent: a bit of trickery involved

Binet's Formula

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

golden ratio

(kindly observe a moment of reverent awe)

Outline

Summary

- Sage does matrices
  - over fields *and* rings
  - symbolic ring! explore!
  - can change base ring
- You can solve some sophisticated problems using matrices on Sage