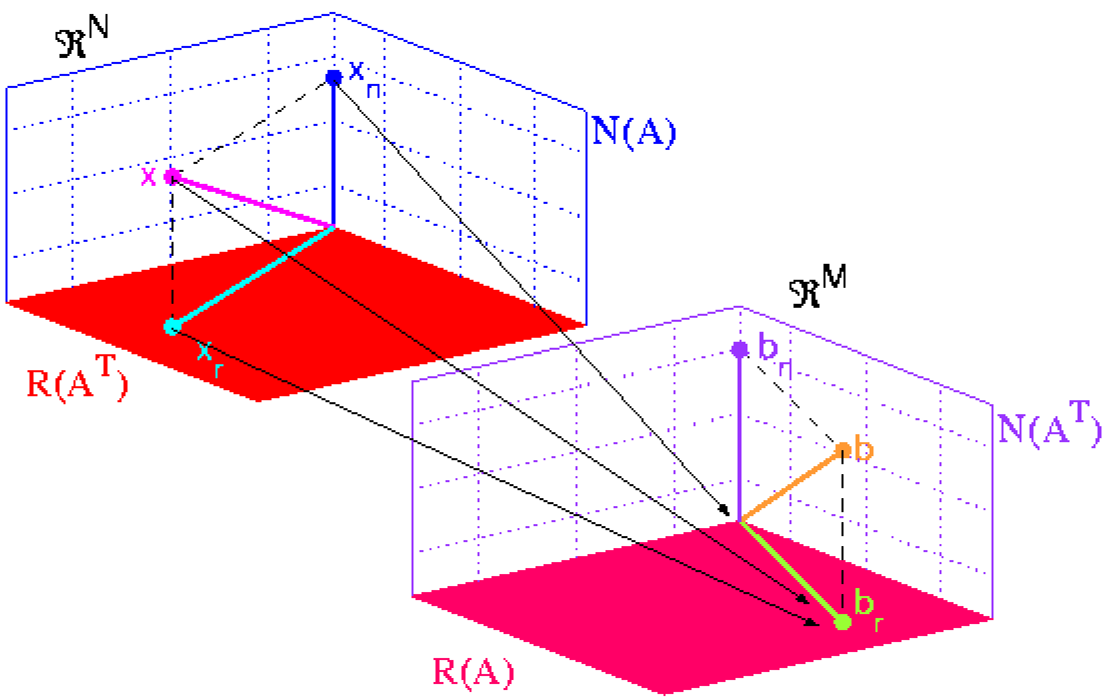
Linear Algebra in Sage

Linear Algebra Tutorial
Sage Days 15
May 16, 2009

Robert A. Beezer

University of Puget Sound



Graphic: Gidon Eshel, Univ. of Chicago

Solving Systems of Equations

Create a matrix and a vector.

A is a 3\times 3 nonsingular matrix.

b is a 3-slot vector.

```
A = matrix([[2,1,1/3],[-1,6,2],[1/2,1,8]])
A
```

```
b = vector([14/3, 2, -6])
b
```

Solve $Ax = b$.

Solve commands ("right" is location of solution vector).

```
A.solve_right(b)
```

```
print A.det()
print A.det() == 0
```

```
A.inverse()
```

Vectors are rows or columns as appropriate, compute $A^{-1}b$

```
A.inverse()*b
```

Can "divide" by a matrix

```
b/A
```

```
b/A.transpose()
```

Augment and row-reduce

```
R = A.augment(matrix(b).transpose())
R
```

```
R.echelon_form()
```

Properties of Matrices

B is a 6\times 5 matrix of rank 4 over the rationals *Unknown control sequence '\QQ'*

```
B = matrix(QQ,[
[10,0,3,8,7],
[-16,-1,-4,-10,-13],
[-6,1,-3,-6,-6],
[0,2,-2,-3,-2],
[3,0,1,2,3],
[-1,-1,1,1,0]
])
B
```

```
B.right_kernel()
```

```
B.right_kernel().basis()
```

```
B.left_kernel().basis()
```

```
B.row_space()
```

```
B.column_space()
```

```
print "Rank", B.rank()
print "Right Nullity", B.right_nullity()
print "Columns", B.ncols()
print
print "Rank", B.rank()
print "Left Nullity", B.left_nullity()
print "Rows", B.nrows()
```

C is a random 50\times 50 matrix over *Unknown control sequence '\QQ'*

```
C = random_matrix(QQ, 50, num_bound=10, den_bound=10)
C
```

Column 5 of the matrix.

```
C[4]
```

Python slicing for entries, show() for looks

```
show(C[20:30, 25:35])
```

```
C.det()
```

```
C.trace()
```

```
C.norm()
```

EigenStuff

D is a 4\times 4 matrix, not diagonalizable

```
D = matrix(QQ, [
[-2,1,-2,-4],
[12,1,4,9],
[6,5,-2,-4],
[3,-4,5,10]
])
show(D)
```

Build a characteristic polynomial for *D* (then simplify)

```
var('t')
S = D - t*identity_matrix(4)
print S
print
p(t)=S.det()
p(t)
```

```
p.find_root(-10, 10)
```

```
show(p.factor())
```

Easier, but less instructive\dots

```
q(t) = D.charpoly('t')
q
```

```
D.eigenvalues()
```

```
diag, evecs = D.eigenmatrix_right()
print "Diagonal matrix with eigenvalues"
print diag
print
print "Matrix with eigenvectors as columns"
print evecs
```

Jordan Canonical Form

```
D.jordan_form()
```

```
J, P = D.jordan_form(transformation = True)
print J
print
print P
```

Check the results

```
P^(-1)*D*P
```

Manipulate the basis for the Jordan form representation

```
Q = P.transpose().gram_schmidt()[0].transpose()
Q
```

Orthogonal?

```
Q.transpose()*Q
```

Resulting representation?

```
Q^(-1)*D*Q
```

Decompositions

LU, QR, SVD, Jordan Canonical Form, Smith Normal Form, Cholesky Decomposition, \dots

Convert D to a matrix E over the reals (double precision), *Unknown control sequence '\RR'*, to obtain QR decomposition

```
E = D.change_ring(RDF)
E
```

```
ortho, triangular = E.QR()
print "Orthogonal"
print ortho
print
print "Upper Triangular"
print triangular
print
```

Checks

```
(ortho.transpose()*ortho - identity_matrix(4)).norm()
```

```
(ortho*triangular - E).norm()
```

Vector Spaces

Sage is so much more than numerical computation.

Can work naturally with vector spaces and modules over a variety of fields and rings.

```
F.<a> = FiniteField(3^2)
```

```
F
```

V is a 3-dimensional vector space over F

```
V=F^3
V
```

```
V.list()
```

"Generator" of all 2-D subspaces of V

```
subs = V.subspaces(2)
```

Python "list comprehension"

```
all_subs = [U for U in subs]
```

Grab one of the subspaces, #42

```
all_subs[42]
```

How many such subspaces?

How many basis matrices in echelon_form?

```
all_subs[86]
```

```
len(all_subs)
```

$= (3^2)^2 + (3^2)^1 + 1$

Accuracy

Octave and Matlab emphasize numerical results - everything is a floating point number.

Their rational form is deceptive. Example by William Stein:

octave:1> format **rat**;
octave:2> a = [-86/17,40/29,-68/43,-20/11;-24/17,-1/38,-2/25,49/17]
a =
    -86/17   40/29   -68/43   -20/11
    -24/17   -1/38   -2/25   49/17
octave:3> rref(a)
ans =
        1   0   155/2122   -725/384
        0   1   -152/173 -6553/795

and in Matlab:

>> format **rat**;
>> a = [-86/17,40/29,-68/43,-20/11;-24/17,-1/38,-2/25,49/17]

a =
    -86/17   40/29   -68/43   -20/11
    -24/17   -1/38   -2/25   49/17

>> rref(a)

ans =
      1   0   13/178   -725/384
      0   1   -152/173   -1426/173

Now in Sage:

```
F = matrix(2,[-86/17, 40/29, -68/43, -20/11, -24/17, -1/38, -2/25,
49/17])
show(F.echelon_form())
```

Entry in lower right corner:

```
print N(-6553/795, digits = 9), "  Octave"
print N(-1426/173, digits=9), "  Matlab"
print N(-30037214/3644069, digits = 9), "  Sage"
```

Speed

Matrices with symbolic entries

```
var('x y')
n=6
entries = [x^i-y^j+i+j for i in range(1,n+1) for j in range(1,n+1)]
G = matrix(SR, n, entries)
G
```

```
G.det()
```

full_output.txt

```
G.det().simplify_full()
```

```
time G.det()
```

full_output.txt

Reals, 53-bit precision

```
H = random_matrix(RR, 10)
time H.det()
```

Reals, 200-bit precision

```
J = random_matrix(RealField(200), 10)
time J.det()
```

Reals, Interval Arithmetic

```
K = (1/17.0)*random_matrix(RIF, 10, bound = 10)
time K.det()
```

Reals, Double Precision

```
L = random_matrix(RDF, 300)
timeit("L.det()")
```

Rationals (Exact)

```
M = random_matrix(QQ, 800, num_bound = 10, den_bound = 10)
time M.det()
```