2014-05-14-linear-algebra.sagews

May 19, 2014

Contents

1 Math 480b Sage Course

1.1 Linear Algebra, part 1

1.2 May 14, 2014

Screencast: (not available, since it Quicktime silently crashed during the lecture)

Plan

- Questions
- Homework:
  - hw7, etc., due Monday morning at 6am
  - talk to Simon about any grading related issues
  - Simon extra office hours: Thursday from 13:00-15:00 in my office, PDL C-430. Come chat if you have any queries about the homework for this week, the grading for last week, or the grading results from the week before!
- Topic: Exact linear algebra
  - solving a system of equations: simple small naive approach
  - creating matrices and vectors
  - solving a matrix equations
  - computing invariants of matrices

1.3 Exact Linear Algebra

Our topic for today is exact linear algebra, by which I mean algebra with matrices whose entries are exact numbers (integers, rational numbers, elements of a finite field, etc.), rather than approximate numbers (floating point numbers).

These are very important in coding theory, combinatorics, much of pure math research, etc. They are not so important in applied math, where one usually works with matrices having floating point (or complex) entries, and the algorithms and issues (e.g., numerical analysis to deal with rounding errors) are much different.

1.4 Solving a system of linear equations

Here is an example to illustrate the naive (painful) direct approach, which is fine for small example and some educational applications.

```
# create 3 symbolic variables:
%var x,y,z

# make a list of 3 linear equations
v = [2*x + 3*y + 5*z == 10,
     7*x - 4*y == 5,
     2*x - 5*y + z == 2]

# solve the equations for x,y,z
s = solve(v, [x,y,z], solution_dict=True)
s
```

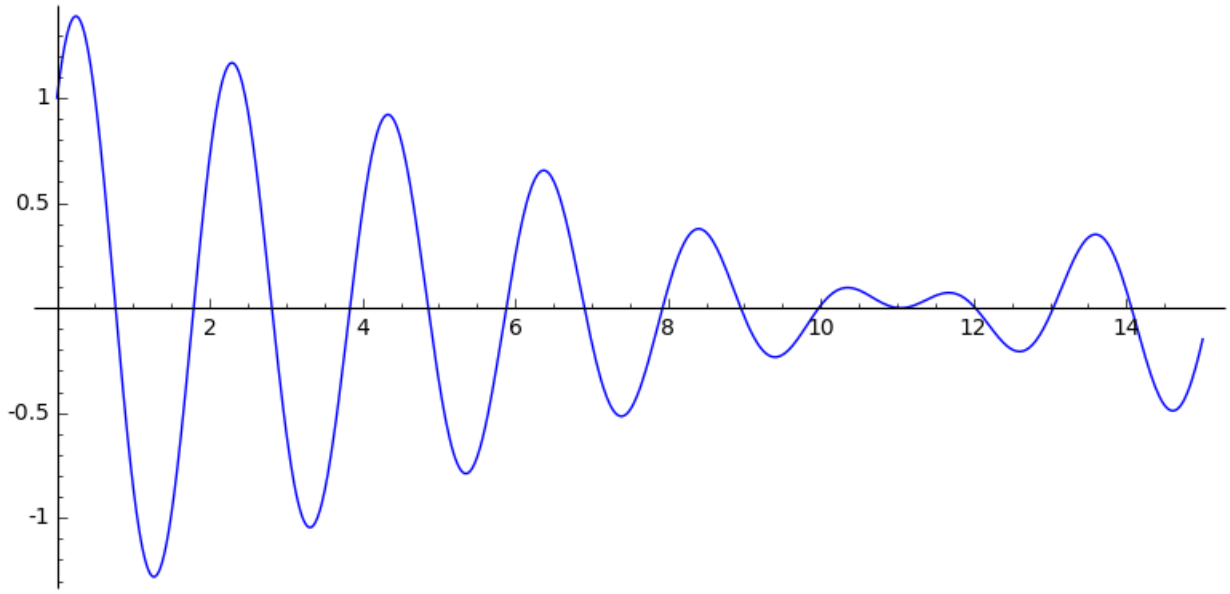
```
[{z: 62/41, x: 35/41, y: 10/41}]
```

```
[cos(pi*x) == -sin(3*x)]
```

```
find_root(sin(3*x)+cos(x*pi), 0, 15)
```

```
14.067002298373158
```

```
plot(sin(3*x)+cos(x*pi), 0, 15)
```




```
# the only solution:
t = s[0]; t
```

```
{z: 62/41, x: 35/41, y: 10/41}
```

```
t[z]
```

```
62/41
```

NOTE: The solve command mostly uses some very generic/general machinery, so you can put some weird nonlinear terms in.

```
v = [2*x^2 + 3*y + 5*z == 10,
     7*x - 4*y == 5,
     2*x - 5*y + z^2 == 2]

solve(v,[x,y,z])
```

```
[[x == (-4.59043682722 - 2.3161925967*I), y == (-9.28326444764 - 4.05333704422*I), z ==
(1.28701382069 - 6.07386640933*I)], [x == (-4.59043682722 + 2.3161925967*I), y ==
(-9.28326444764 + 4.05333704422*I), z == (1.28701382069 + 6.07386640933*I)], [x ==
0.930873621713, y == 0.379028882378, z == 1.42597239649], [x == 3, y == 4, z == -4]]
```

```
# solve a cubic algebraic equation
solve(x^5 + 5*x + 2, x)
```

```
[0 == x^5 + 5*x + 2]
```

1.5 Creating matrices and vectors

```
#2*x + 3*y + 5*z == 10,
#7*x - 4*y == 5,
#2*x - 5*y + z == 2
```

```
m = matrix(3,3, [2,3,5,  7,-4,0,  2,-5,1])

v = vector([10,5,2])
```

```
m[0,0]
```
```
2
```

```
m[0,0] = 10
```

```
m[0]
```
```
(10, 3, 5)
```

```
m.column(0)
```
```
(10, 7, 2)
```

```
m.row(0)
```
```
(10, 3, 5)
```

```
m[0][0] = 20
```
```
Error in lines 1-1
Traceback (most recent call last):
  File "/projects/74af30b7-ad25-4308-a02e-c71fcd84de6e/.sagemathcloud/sage_server.py",
line 733, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in <module>
  File "vector_integer_dense.pyx", line 185, in
sage.modules.vector_integer_dense.Vector_integer_dense.__setitem__
(sage/modules/vector_integer_dense.c:3760)
ValueError: vector is immutable; please change a copy instead (use copy())
```

```
type(m)
```
```
<type 'sage.matrix.matrix_integer_dense.Matrix_integer_dense'>
```

```
m[0] = 10/3
```
```
Error in lines 1-1
Traceback (most recent call last):
  File "/projects/74af30b7-ad25-4308-a02e-c71fcd84de6e/.sagemathcloud/sage_server.py",
line 733, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in <module>
  File "matrix0.pyx", line 1404, in sage.matrix.matrix0.Matrix.__setitem__
(sage/matrix/matrix0.c:7295)
  File "matrix0.pyx", line 1483, in sage.matrix.matrix0.Matrix._coerce_element
(sage/matrix/matrix0.c:8266)
  File "parent.pyx", line 1070, in sage.structure.parent.Parent.__call__
(sage/structure/parent.c:8858)
  File "rational.pyx", line 3795, in sage.rings.rational.Q_to_Z._call_
(sage/rings/rational.c:26796)
TypeError: no conversion of this rational to integer
```

```
m[0]
```

```
(2, 3, 5)
```

```
show(m)
show(v)
```

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

$$(10, 5, 2)$$

```
matrix([[1,2], [3,4/7]])
```

```
[  1   2]
[  3 4/7]
```

```
# a matrix with entries in the symbolic ring
%var x
reset('e')
[[pi,e,sqrt(2)], [1,2,sin(x)]]
```

```
[[pi, e, sqrt(2)], [1, 2, sin(x)]]
```

```
m = matrix([[pi,e,sqrt(2)], [1,2,sin(x)]])
show(m)
```

$$\begin{pmatrix} \pi & e & \sqrt{2} \\ 1 & 2 & \sin\left(x\right) \end{pmatrix}$$

```
m.base_ring()
```

```
Symbolic Ring
```

```
%var x, y
m = matrix(1,[sin(x)/cos(y) == 1])
```

```
m.subs({x:5, y:10})
```

```
[sin(5)/cos(10) == 1]
```

```
# a biger matrix (all entries are zero by default)
m = matrix(25,50)
m
```

```
25 x 50 dense matrix over Integer Ring (type 'print m.str()' to see all of the entries)
```

```
print m.str()
```

```
[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 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 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 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 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 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 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 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 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 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 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]
```

```
# a random matrix with integer entries
random_matrix(ZZ, 5, 10)
```

```
[  0   0  -1   0   0  -1   2  -2   1   0]
[  0   0  -2   0   0   0   2  -1  -1   2]
[ -3  -1  -1   0  -1  -1 369  25   1   0]
[  0  -1  -2  -1   0  -1  13   0   1  -3]
[  1  12  -1   3  -3   9  -5  -1  -2   1]
```

```
ZZ.random_element()
```

```
-2
```

```
random_matrix(ZZ, 5, 10, x=-3, y=3)  # entries chosen uniformly from \
    [-3..3]
```

```
[-3  2 -2 -3  1  2 -3 -1  0  0]
[-1  0 -2  0 -3  1 -3 -3  2  1]
[-1 -3  1  2  1  1 -2 -3 -2 -2]
[ 1 -1 -2 -3 -3  0  2  1 -2 -3]
[-3 -3 -3  0  0  0 -1  0 -2 -1]
```

```
a = random_matrix(QQ, 3); a
b = random_matrix(QQ, 3); b
```

```
[  -1    1   -1]
[-1/2    1    0]
[  -2   -1    2]
[   1    1    2]
[  -2  1/2    2]
[-1/2  1/2    0]
```

```
QQ.random_element()
```

```
-14/3
```

```
a + b
```

```
[   0    2    1]
[-5/2  3/2    2]
[-5/2 -1/2    2]
```

```
a
```

```
[  -1    1   -1]
[-1/2    1    0]
[  -2   -1    2]
```

```
# addition adds to the *diagonal*
a + 10
```

```
[   9    1   -1]
[-1/2   11    0]
[  -2   -1   12]
```

```
# matrix multiplication is matrix multiplication
a * b
```

```
[-5/2   -1    0]
[-5/2    0    1]
[  -1 -3/2   -6]
```

```
a + a.transpose()
```

```
[ -2 1/2  -3]
[1/2   2  -1]
[ -3  -1   4]
```

```
a = matrix(RR, 2,2, [1,2,3,4]); a
```

```
[1.00000000000000 2.00000000000000]
[3.00000000000000 4.00000000000000]
```

```
a = matrix(2, [1,2,3,4]); a
```

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

```
a.change_ring(RR)
```
```
[1.00000000000000 2.00000000000000]
[3.00000000000000 4.00000000000000]
```

```
M = MatrixSpace(RR, 2)
M
```
```
Full MatrixSpace of 2 by 2 dense matrices over Real Field with 53 bits of precision
```

```
M([1,2,3,4])
```
```
[1.00000000000000 2.00000000000000]
[3.00000000000000 4.00000000000000]
```

1.6 Solving a matrix equations

```
m = matrix(3,3, [2,3,5,  7,-4,0,  2,-5,1])
v = vector([10,5,2])
```

```
m
```
```
[ 2  3  5]
[ 7 -4  0]
[ 2 -5  1]
```

```
# solve m*x = v
x = m.solve_right(v); x
```
```
(35/41, 10/41, 62/41)
```

```
m*x == v
```
```
True
```

```
# solve x*m = v
```

```
x = m.solve_left(v); x
```
```
(129/164, 72/41, -317/164)
```

```
x*m
```
```
(10, 5, 2)
```

```
# use matlab notation
x = m \ v; x
```
```
(35/41, 10/41, 62/41)
```

```
preparse('x = m \ v')
```
```
'x = m  * BackslashOperator() * v'
```

Solving gives you back one solution, if there is one, even if there are infinitely many. To get all of them you would add elements of the nullspace (or kernel)

```
m = matrix(3,3, [2,3,5,  7,-4,0,  2,-5,1])
b = random_matrix(QQ, 3,2); b
```
```
[  0  -1]
[ -2 1/2]
```

```
[  2   2]
```

```
# solve m*x == b, where b is a *matrix*, so x is also a matrix
x = m.solve_right(b); x
```
```
[ -24/41  -15/82]
[ -43/82 -73/164]
[  45/82  23/164]
```

```
m*x == b
```
```
True
```

1.7 Computing invariants of matrices

```
m = matrix(3,3, [2,3,5,  7,-4,0,  2,-5,1])
```

```
m.determinant()
```
```
-164
```

```
m.rank()
```
```
3
```

```
m.nullity()
```
```
0
```

```
m.rref()  # watch out -- not the same as m.echelon_form(), in general...
```
```
[1 0 0]
[0 1 0]
[0 0 1]
```

```
m.echelon_form()  # this is the echelon form *over ZZ* -- no dividing \
    allowed
```
```
[  1   0  12]
[  0   1 103]
[  0   0 164]
```

```
m.characteristic_polynomial()
```
```
x^3 + x^2 - 41*x + 164
```

```
m.minimal_polynomial()
```
```
x^3 + x^2 - 41*x + 164
```

```
e = m.eigenvalues(); e
```
```
[-8.30939752116266?, 3.654698760581329? - 2.525839783704967?*I, 3.654698760581329? +
2.525839783704967?*I]
```

```
lamb = e[0]; lamb
```
```
-8.30939752116266?
```

```
# what's up with the "?"?
type(lamb)
```
```
<class 'sage.rings.qqbar.AlgebraicNumber'>
```

```
lamb.minpoly()
```
```
x^3 + x^2 - 41*x + 164
```
```
# really lamb is an *infinite* precision eigenvalue -- you can ask for \
   more (correct) digits...
lamb.numerical_approx(digits=50)
```
```
-8.309397521162656840421334675085550768272082567069 7
```
```
# compute the eigespace decomposition
m.eigenspaces_right()
```
```
[
(-8.30939752116266?, Vector space of degree 3 and dimension 1 over Algebraic Field
User basis matrix:
[                 1 -1.624356993204802? -1.087265308309651?]),
(3.654698760581329? - 2.525839783704967?*I, Vector space of degree 3 and dimension 1 over
Algebraic Field
User basis matrix:
[                                       1   0.824678496602401? + 0.2721211925688051?*I
-0.1638673458451749? - 0.6684406722822763?*I]),
(3.654698760581329? + 2.525839783704967?*I, Vector space of degree 3 and dimension 1 over
Algebraic Field
User basis matrix:
[                                       1   0.824678496602401? - 0.2721211925688051?*I
-0.1638673458451749? + 0.6684406722822763?*I])
]
```
```
# Jordan Canonical Form
m.jordan_form(QQbar)
```
```
[                     -8.30939752116266?|                                       0|
0]
[---------------------------------------+----------------------------------------+-----
-----------------------------------]
[                                     0|3.654698760581329? - 2.525839783704967?*I|
0]
[---------------------------------------+----------------------------------------+-----
-----------------------------------]
[                                     0|
0|3.654698760581329? + 2.525839783704967?*I]
```