Linear Algebra Step by Step in SymPy

Dario Beraldi

March 14, 2015

Contents

Abstract

Excerpts from *Linear Algebra Step by Step* by K. Singh with SymPy implementation.

Linear Equations and Matrices

Solving linear systems

Exercises 1.1 2f

$$\begin{aligned} ex - ey &= 2 \\ ex + ey &= 0 \end{aligned} \tag{1}$$

```
x, y= symbols('x y', real= true)
eq1= Eq(E*x - E*y, 2)
eq2= Eq(E*x + E*y, 0)
sols= solve([eq1, eq2])
```

Solved for:

$$\left\{x : e^{-1}, \quad y : -\frac{1}{e}\right\} \tag{2}$$

Check solutions by substituting them in the original equations 1

```
eq1.subs(sols)
True
eq2.subs(sols)
True
```

Solve the system 1 using matrix notation

```
coeffs= Matrix([[E, -E], [E, E]])
const= Matrix([2, 0])
[coeffs, const]
```

$$\left[\begin{bmatrix} e & -e \\ e & e \end{bmatrix}, \quad \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right] \tag{3}$$

If everything is correct, solutions are consistent with 2:

```
coeffs.solve(const)
```

$$\begin{bmatrix} e^{-1} \\ -\frac{1}{e} \end{bmatrix} \tag{4}$$

Exercises 1.1 4a

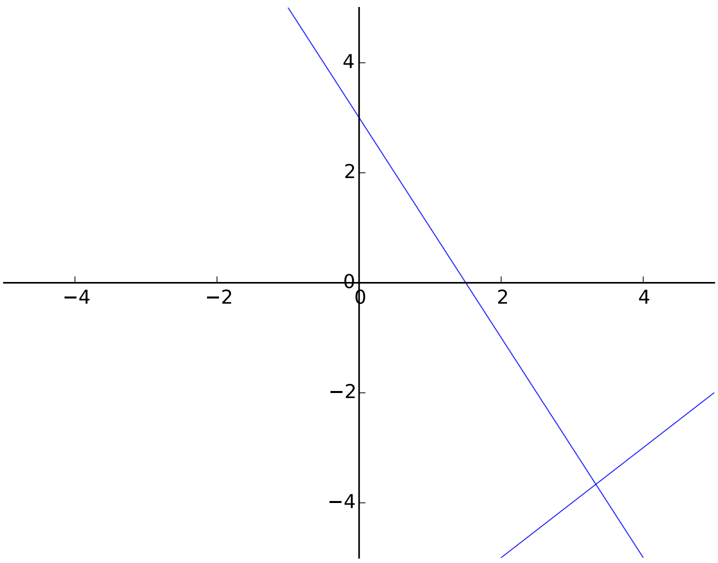
Plot the graphs of these linear equations

$$\begin{gathered} 2x + y = 3 \\ x - y = 7 \end{gathered} \tag{5}$$

```
eq1= Eq(2*x + y, 3)
eq2= Eq(x - y, 7)
```

Plot

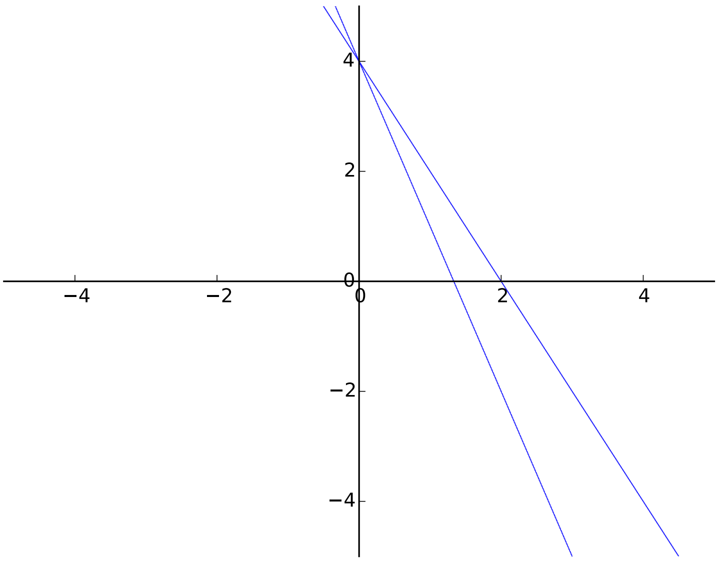
```
pp1= plot_implicit(eq1)
pp2= plot_implicit(eq2)
pp1.extend(pp2)
pp1.save('ex4a.pdf')
```



Exercises 1.1 5b

```
eq1= Eq(12*x + 4*y, 16)
eq2= Eq(8*x + 4*y, 16)

pp1= plot_implicit(eq1)
pp2= plot_implicit(eq2)
pp1.extend(pp2)
pp1.save('ex5b.pdf')
```



Solve by row echelon form

Given a linear system of equations, the solutions can be found by Gaussian elimination:

- Extract the coefficients and constants from the equations and put them in an augmented matrix.
- Transform the augmented matrix in reduced row echelon form (rref). This is the result of the Gaussian elimination process.
- The last columns of the rref matrix lists the solutions of the linear system.

Reminder: rref means that every leading coefficient is 1 and is the only nonzero entry in its column [1].

[1] http://en.wikipedia.org/wiki/Row_echelon_form

Exercises 1.2 2a

$$
\begin{aligned}
x + 2y + 3z &= 12 \\
2x - y + 5z &= 3 \\
3x + 3y + 6z &= 21
\end{aligned}
\tag{6}
$$

```
eq1= Eq(x + 2*y + 3*z, 12)
eq2= Eq(2*x - y + 5*z, 3)
eq3= Eq(3*x + 3*y + 6*z, 21)
```

Represent the linear systen as *augmented* matrix, where the last column holds the constants:

$$
M = \begin{bmatrix} 1 & 2 & 3 & \mathit{12} \\ 2 & -1 & 5 & \mathit{3} \\ 3 & 3 & 6 & \mathit{21} \end{bmatrix} \tag{7}
$$

In SymPy use the `Poly` class to conveniently extract the coefficients at the left hand side of the equations:

```
M= Matrix([
    Poly(eq1.lhs).coeffs(),
    Poly(eq2.lhs).coeffs(),
    Poly(eq3.lhs).coeffs()
])
const= Matrix([eq1.rhs, eq2.rhs, eq3.rhs])
M= const.col_insert(0, M)
```

In reduced row echelon form, with indexes of the pivot variables on the right:

$$
RREF = \left(\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0, & 1, & 2 \end{bmatrix} \right) \tag{8}
$$

Solutions can be read from last column of the RREF $\begin{bmatrix} x:1 \\ y:4 \\ z:1 \end{bmatrix}$. Verify the solutions solve the initial system of equations:

```
rref= M.rref()
sols= rref[0].col(-1)

eq1.subs({x:sols[0], y:sols[1], z:sols[2]}) # True
eq2.subs({x:sols[0], y:sols[1], z:sols[2]}) # True
eq3.subs({x:sols[0], y:sols[1], z:sols[2]}) # True

# Or the same returning dict {x: 1, y: 4, z: 1}
solve([eq1, eq2, eq3])
```

Exercises 1.2 3d

Linear system:

$$\begin{aligned} -2x + 3y - 2z &= 8 \\ -x + 2y - 10z &= 0 \\ 5x - 7y + 4z &= -20 \end{aligned} \tag{9}$$

Augmented matrix:

$$\begin{bmatrix} -2 & 3 & -2 & 8 \\ -1 & 2 & -10 & 0 \\ 5 & -7 & 4 & -20 \end{bmatrix} \tag{10}$$

Reduced row echelon form with solutions in the last column:

$$\left(\begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & \frac{1}{2} \end{bmatrix}, \quad \begin{bmatrix} 0, & 1, & 2 \end{bmatrix} \right) \tag{11}$$

In `SymPy` :

```
eq1= Eq(-2*x + 3*y -2*z, 8)
eq2= Eq(-x + 2*y - 10*z, 0)
eq3= Eq(5*x - 7*y + 4*z, -20)

M= Matrix([
    Poly(eq1.lhs).coeffs(),
    Poly(eq2.lhs).coeffs(),
    Poly(eq3.lhs).coeffs()
])
const= Matrix([eq1.rhs, eq2.rhs, eq3.rhs])
M= const.col_insert(0, M)
rref= M.rref()
sols= rref[0].col(-1)

# Checked: {x: -3, y: 1, z: 1/2}
solve([eq1, eq2, eq3])
```