MatricesAndVectors

John Hoggard

Spring 2015

1 Matrices and Vectors

1.1 Variables and Systems of Equations

Sage is able to handle systems of linear equations with symbolic variables, but you must first declare that your variables are _symbolic_. For instance, to use variables x, y, and z, enter the expression in the cell below. (Click on the cell below and hit "shift-return".)

```
var('x, y, z')
```

Now if you want to solve the system

$$\begin{array}{rrrcr} 3x & +2y & & = & 4 \\ x & -y & +z & = & 0 \\ -x & +y & -5z & = & -2 \end{array}$$

you may proceed using the solve command. You will need to enter a list of equations (in square brackets) followed by a list of variables (also in square brackets).

Try it out in the cell below, where I have pre-entered the correct command:

```
solve([3*x+2*y==4, x-y+z==0, -x+y-5*z==-4], [x, y, z])
```

Two important points above: * When entering a variable with a coefficient, you must use an asterisk ('*') to indicate the multiplication. * Equations are entered using a double equals sign ('=='), not a single equals sign.

Hm, I wonder how this deals with systems that do not have a unique solution? You try using Sage to solve the following system, which is Example 2.14 in Section I.2, pp. 18-19 in our text:

$$\begin{array}{rrrcr} x & -y & +z & = & 1 \\ 3x & & +z & = & 3 \\ 5x & -2y & +3z & = & 5 \end{array}$$

If you click on the horizontal line below this cell, it will create a new cell for you to enter a calculation into. Try it now, and see if you can interpret what Sage tells you.

```
solve([x-y+z==1, 3*x+z==3, 5*x-2*y+3*z==-5], [x, y, z])
[]
```

1.2 Matrices and Vectors

Sage is also very adept at handling matrices and vectors. The matrix associated with the left-hand side of the above system is

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

Matrices are created using the matrix command, but you need to tell sage what kinds of entries the matrix should have. We will use rational numbers, indicated by QQ. Then we enter the matrix as a list of lists. Enter the line below to define a matrix m.

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

Sage can print objects it understands with nice formatting using the show command. So to see a nice representation of the matrix you can enter the command show(m). Try it now, by clicking on the line below to get a new cell, then typing the command and pressing shift-return:

```
show(m)
```

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

Next, we'll enter the right-hand side as a vector: (Enter the line below.)

```
v = vector(QQ, [4, 0, -2])
```

```
show(v)
```

$(4, 0, -2)$

(Notice that the above is a row vector, rather than a column vector. Sage is not very picky about the difference in some situations, but if you wanted to force Sage to recognize this as a column vector, you could instead append the method .column() to the end of the command like so: v = vector(QQ, [4, 0, -2])).column(). Sage uses this approach of functions tacked on to the end of an object quite a bit, as we will see shortly.)

Next create an augmented matrix out of the matrix with the column vector v appended:

```
mprime = m.augment(v, subdivide=True)
mprime
[ 3  2  0| 4]
[ 1 -1  1| 0]
```

```
[-1  1 -5|-2]
```

Notice that we included a second mprime in the cell above so that Sage would output the matrix for us to examine. (How can you get a nicely formatted display of your new augmented matrix mprime?)

Now let's work on reducing the matrix. Here are some built-in commands which can be applied to the matrix to help us reduce it. (Each of these operations will replace the matrix mprime with the new matrix which results.) * .swap_rows(row1, row2): This interchanges row1 and row2, where row1 and row2 are row numbers. * .rescale(row, scalar): This multiplies row by the number scalar. * .add_multiple_of_row(targetrow, startrow, scalar): Adds scalar times startrow to targetrow.

Important Note: Sage numbers rows (and entries) starting from zero, so what we call row 1, it calls row 0. (I admit that this is confusing for most people staring out, but it is a standard convention in many computer languages.) So for example, swapping the first two rows can be accomplished by the command

```
mprime.swap_rows(0, 1)
mprime
```

```
[ 1 -1  1| 0]
[ 3  2  0| 4]
[-1  1 -5|-2]
```

```
m.rescale_row()
```

(We entered mprime a second time so that Sage would show us what was in the matrix after the swap.) Now let's add -3 times the first (zeroth) row to the second (first) row to further reduce the matrix:

```
mprime.add_multiple_of_row(1, 0, -3)
mprime
```

```
[ 1 -1  1| 0]
[ 0  5 -3| 4]
[-1  1 -5|-2]
```

Now you complete the row reduction to reduced echelon form. Remember that you can create a new cell by clicking on the horizontal line below. Don't forget to type "mprime" again after each operation to see what is in the matrix.

(If you make a mistake, you can always reverse an operation, because we have proven that all row operations are reversible!)

1.2.1 Reduced Echelon Form

There are other techniques for manipulating matrices built in to Sage. Look up the method ".rref()" for the matrix mprime in Sage's help using the following command: mprime.rref?

```
mprime.rref?
```

Try it out! How does your answer compare to what you got before?

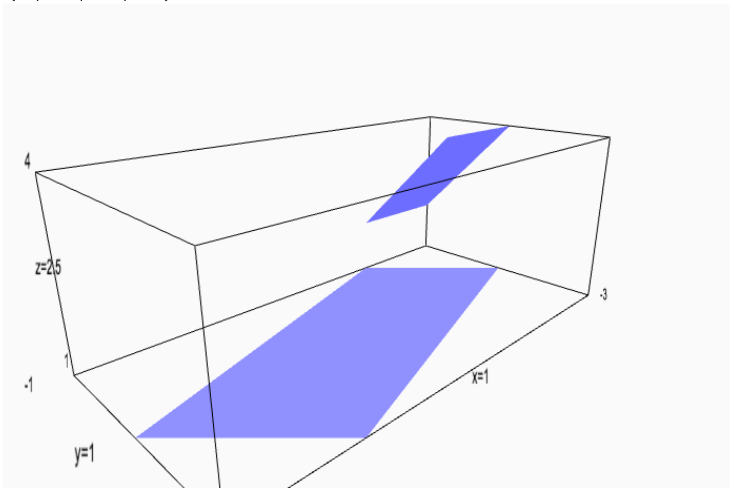
1.3 Parametric Plots

Sage can create many different types of graphs. We will be interested in parametric plots, and usually in three dimensions. On Homework 2, you were asked to find the intersection of the two planes

$$\begin{pmatrix}1\\1\\1\end{pmatrix}+\begin{pmatrix}3\\1\\0\end{pmatrix}s+\begin{pmatrix}1\\-1\\0\end{pmatrix}t, \qquad \begin{pmatrix}-2\\0\\3\end{pmatrix}+\begin{pmatrix}-1\\0\\1\end{pmatrix}m+\begin{pmatrix}0\\1\\1\end{pmatrix}n$$

These are each defined by a set of parametric equations. We can plot these using parametric_plot3d, which takes a list of parametric equations (for x, y, and z), and then a range for each of the parameters. (Don't forget to declare your variables for your parameters.)

```
var('s, t, m, n')
p1 = parametric_plot3d((1 + 3*s + t, 1 + s - t, 1), (s, -1, 1), (t, \
    -1, 1))
p2 = parametric_plot3d((-2 -m, n, 3 + n), (m, -1, 1), (n, -1, 1))
p1+p2
(s, t, m, n)
```



Oops, the parts of the planes we plotted above didn't show the intersection. Try expanding the range of values for s, t, m, and n so that you can see the intersection. (Note that you can grab and move the plot, too.)

Note the last line: Plots are combined by addition in Sage.