Balancing a chemical reaction using linear algebra

Suppose that we wish to balance the following chemical equation:

$$PhCH_3 + KMnO_4 + H_2SO_4 \to PhCOOH + K_2SO_4 + MnSO_4 + H_2O.$$

In python, the user writes this as a string

```
'PhCH3 + KMnO4 + H2SO4 = PhCOOH + K2SO4 + MnSO4 + H2O'
```

In order to balance the equation, we must make sure that each element appears the same number of times on both sides of the equation, by finding positive integer values for the variables $x_0, \dots, x_6$ in

$$x_0\ PhCH_3 + x_1\ KMnO_4 + x_2\ H_2SO_4 \to x_3\ PhCOOH + x_4\ K_2SO_4 + x_5\ MnSO_4 + x_6\ H_2O.$$

Balancing for each element, we get the system of equations:

$$\begin{array}{rc}
Ph: & x_0 = x_3 \\
C: & x_0 = x_3 \\
H: & 3x_0 + 2x_2 = x_3 + 2x_6 \\
K: & x_1 = 2x_4 \\
Mn: & x_1 = x_5 \\
O: & 4x_1 + 4x_2 = 2x_3 + 4x_4 + 4x_5 + x_6 \\
S: & x_2 = x_4 + x_5.
\end{array}$$

Let's write these in a more organized form:

$$\begin{array}{rcccccccc}
Ph: & x_0 & +0x_1 & +0x_2 & -x_3 & +0x_4 & +0x_5 & +0x_6 & = 0 \\
C: & x_0 & +0x_1 & +0x_2 & -x_3 & +0x_4 & +0x_5 & +0x_6 & = 0 \\
H: & 3x_0 & +0x_1 & +2x_2 & -x_3 & +0x_4 & +0x_5 & -2x_6 & = 0 \\
K: & 0x_0 & +x_1 & +0x_2 & +0x_3 & -2x_4 & +0x_5 & +0x_6 & = 0 \\
Mn: & 0x_0 & +x_1 & +0x_2 & +0x_3 & +0x_4 & -x_5 & +0x_6 & = 0 \\
O: & 0x_0 & +4x_1 & +4x_2 & -2x_3 & -4x_4 & -4x_5 & -x_6 & = 0 \\
S: & 0x_0 & +0x_1 & +x_2 & +0x_3 & -x_4 & -x_5 & +0x_6 & = 0.
\end{array}$$

This shows us that we can write this system of equations as a single equation in terms of matrices:

$$\begin{pmatrix}
1 & 0 & 0 & -1 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 \\
3 & 0 & 2 & -1 & 0 & 0 & -2 \\
0 & 1 & 0 & 0 & -2 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 4 & 4 & -2 & -4 & -4 & -1 \\
0 & 0 & 1 & 0 & -1 & -1 & 0
\end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}
=
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

A standardized way of representing this system is called the augmented matrix, which looks like

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

Note that we added the column of zeros representing the right hand sides of the equations. In the case of chemical reactions, these will always be 0, but there are of coruse also linear systems where those are different constants. Sympy has a function for solving systems of linear equations. It takes as input an augmented matrix, and the symbolic variables. Note that the number of symbols needed is one less than the number of columns of the augmented matrix. (The number of rows corresponds to the number of equations; in our case the number of elements appearing in the reaction.)

```
import sympy as sp
[x0,x1,x2,x3,x4,x5,x6]=sp.symbols('x0 x1 x2 x3 x4 x5 x6')

M=sp.Matrix([[1,0,0,-1,0,0,0,0],[1,0,0,-1,0,0,0,0],[3,0,2,-1,0,0,-2,0],
   [0,1,0,0,-2,0,0,0],[0,1,0,0,0,-1,0,0],[0,4,4,-2,-4,-4,-1,0],
   [0,0,1,0,-1,-1,0,0]])

sols=sp.solve_linear_system(M,x0,x1,x2,x3,x4,x5,x6)
```

Sympy allows a faster way to set an arbitrary number n of symbolic variables, and call them as paramaters from a list, as follows

```
from sympy.parsing.sympy_parser import parse_expr
x=[parse_expr('x%d'%i) for i in range(n)]
x=sp.symbols('x0:%d'%n)
sols=sp.solve_linear_system(M,*x)
```

Note the $*$ in the last function: this tells python to use the elements from the list as input parameters, which is different from using the list object as an input parameter. The linear system solves in sympy outputs the solution as a dictionary, where some variables may depend on others. Linear systems come in three types, only one of which will be relevant for this problem:

- **No solutions.** The system $x = 0, x = 1$ has no solutions. In this case, sympy gives **no output**.

```
x,y=sp.symbols('x y')
M=sp.Matrix([[1,0],[1,1]])
sols=sp.solve_linear_system(M,x)
sols
Output:
```

- **One solution.** The system $x = 0, y = 1$ has one solution. Sympy gives the solution as a dictionary, which maps each variable to a number.

```
M=sp.Matrix([[1,0,0],[0,1,1]])
sols=sp.solve_linear_system(M,x,y)
sols
Output:  {x: 0, y: 1}
```

- **Infinite solutions.** This will always be the case for our chemical reactions! The system $x = y$ has infinitely many solutions. Sympy gives the solution as a dictionary, which maps some variables to functions of other variables. The variables that do not appear in `sols.keys()` are the **free variables**. We can set these to be any value, which then determines the values of the variables that are in `sols.keys()`.

```
M=sp.Matrix([[1,-1,0]])
sols=sp.solve_linear_system(M,x,y)
sols
Output:  {x: y}
```

The solution to our chemical reaction looks as follows:

```
{x2:9*x6/14, x1:3*x6/7, x0:5*x6/14, x5:3*x6/7, x3:5*x6/14, x4:3*x6/14}
```

If we set all of our free variables (x_6) to 1, then we get the following solution list:

```
coeffs=[5/14,3/7,9/14,5/14,3/14,3/7,1]
```

You will make use of the `subs()` function to obtain this list. We would like our solution to consist of only positive integers, and for them to be as small as possible. We can always obtain a new solution from an existing one by multiplying all of the coefficients by the same number. (In a chemical reaction, taking twice as much of the ingredients gives twice as much product.) Sympy has its own object class for rational numbers: `sp.Rational`. The solutions will (most likely; it is worth checking) be of this type. This means that we can ask for the numerator and denominator as a list, using `sp.fraction()`:

```
sp.fraction(coeffs[0])
Out:   (5,14).
```

Now, we can extract a list of all the denominators in our solution, and multiply the solution by the Least Common Multiple of the denominators. This removes all fractions, while keeping the solution as small as possible. Another option is to multiply everything by all the denominators, and then divide the resulting solution by the Greatest Common Divisor of the coefficients. In this case, we obtain:

```
f=sp.lcm([14,7,14,14,14,7,1])
coeffs=coeffs*f
coeffs
Out:   [5,6,9,5,3,6,14]
```

This is our solution! Now, we return it to our user as a string (using regular expressions):

```
'5PhCH3 + 6KMnO4 + 9H2SO4 = 5PhCOOH + 3K2SO4 + 6MnSO4 + 14H2O'
```

Exercises

- Is it possible to have a system of equations where all variables are free? What does the solution given by sympy look like in that case?
- Use sympy to solve the following problems[1]
  - The admission fee at a small fair is \$1.50 for children and \$4.00 for adults. On a certain day, 2200 people enter the fair and \$5050 is collected. How many children and how many adults attended?
  - The sum of the digits of a two-digit number is 7. When the digits are reversed, the number is increased by 27. Find the number.
  - Find the equation of the parabola that passes through the points (âĂŞ1, 9), (1, 5), and (2, 12).

[1]From: `http://www.purplemath.com/modules/systprob.htm`