
hydrosolver

Dmytro Strelnikov

Oct 17, 2021

TUTORIAL:

1	Working with compositions	1
1.1	Defining a composition	1
1.2	Loading and dumping compositions	2
1.3	Operations on compositions	2
2	Working with solutions	3
2.1	Defining a solution	3
2.2	Operations on solutions	5
2.3	Correcting solutions	7
3	Optimizing solutions	9
3.1	Using optimizer	9

WORKING WITH COMPOSITIONS

The simplest entity in hydrosolver is `Composition`. Compositions can be defined on the go or loaded from a file, added and scaled.

1.1 Defining a composition

The most straightforward way to define a composition is using its constructor `Composition(name, vector)`. The simplest composition which does not contain any of the nutrient elements of our interest would be `Composition(name='Pure water')`.

The monopotassium phosphate can be defined as follows:

```
>>> from hydrosolver.composition import Composition
>>> MKP = Composition(
...     name='Monopotassium phosphate',
...     vector=[0, 0, 0.2276, 0.2873, 0, 0, 0, 0, 0, 0, 0, 0, 0],
... )
>>> MKP
Composition: Monopotassium phosphate
```

Nutrient	Ratio	Amount mg/kg
P	0.2276	227600
K	0.2873	287300

Here `vector` follows the structure of `composition.nutrients_stencil`. Let us check the result.

It is hard to not notice that this kind of definition is cumbersome and can be barely used by humans. Therefore class `Composition` contains an alternative constructor `Composition.from_dict()`, so the same result could be achieved in the following way:

```
>>> MKP = Composition.from_dict(
...     {'Monopotassium phosphate': {'P': 0.2276, 'K': 0.2873}}
... )
>>> MKP
Composition: Monopotassium phosphate
```

Nutrient	Ratio	Amount mg/kg
P	0.2276	227600
K	0.2873	287300

1.2 Loading and dumping compositions

It makes sense to save frequently used composition into a database and further load it from there. Here is an example:

```
import yaml

with open('database.yaml', 'w') as database:
    database.write(yaml.dump(MKP.as_dict()))
```

Multiple compositions can be loaded at once from a file:

```
from hydrosolver.utils import load_file

compositions = load_file('compositions/pure.yaml')
```

1.3 Operations on compositions

Compositions can be added and scaled, i.e. multiplied by scalars. You will typically not need to add or subtract compositions, but consider the following use case for scaling:

```
>>> KOH = Composition.from_dict(
...     {'Potassium hydroxide': {'K': 0.69687}}
...     )
>>> KOH_94 = 0.94 * KOH
>>> KOH_94
Composition: 0.94 * (Potassium hydroxide)
```

Nutrient	Ratio	Amount mg/kg
K	0.655058	655058

WORKING WITH SOLUTIONS

A more advanced entity in hydrosolver is `Solution`. Solutions consist of a few compositions and can be constructed in different ways. Solutions can be added, scaled, extended and merged.

2.1 Defining a solution

To define a solution we must first define the compositions constituting it. Let us consider a simple example:

```
>>> from hydrosolver.composition import Composition
>>> from hydrosolver.solution import Solution
>>> water = Composition('Pure water')
>>> CN = Composition.from_dict(
...     {'Calcium nitrate tetrahydrate':
...      {'N (NO3-)': 0.1186, 'Ca': 0.1697}}
...     )
>>> solution_CN_10 = Solution(
...     compositions=[CN, water],
...     formulation=[0.1, 0.9],
...     )
>>> solution_CN_10
```

Composition	Amount in kg	Amount in g
Calcium nitrate tetrahydrate	0.1	100
Pure water	0.9	900
Total:	1	1000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
N (NO3-)	0.01186	11860
Ca	0.01697	16970

Here we just defined a 10% (by mass) aqueous solution of calcium nitrate tetrahydrate. It's total mass is given by `solution_CN_10.mass` and equals to 1 [kg]. However, if the solution to construct consists of multiple compositions, it becomes more difficult to adjust the mass of the water. For this purpose there is an alternative constructor `Solution.dissolve()`:

```
>>> solution_CN_10 = Solution.dissolve(
...     mass=1,
```

(continues on next page)

(continued from previous page)

```

...     water=water,
...     compositions_=[CN],
...     formulation_=[0.1],
...     )
>>> solution_CN_10

```

Composition	Amount in kg	Amount in g
-----	-----	-----
Calcium nitrate tetrahydrate	0.1	100
Pure water	0.9	900
Total:	1	1000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO3-)	0.01186	11860
Ca	0.01697	16970

As one can see, for dissolve we first pass the desired total mass of the solution, then the composition which will be used for aligning (typically the water) and the truncated lists of compositions and their amounts without the last element, which will be substituted with water. This way fits more for defining solutions consisting of many compositions:

```

>>> MS = Composition.from_dict(
...     {'Magnesium sulfate heptahydrate':
...      {'Mg': 0.0986, 'S': 0.1301}}
...     )
>>> my_solution = Solution.dissolve(
...     mass=1,
...     water=water,
...     compositions_=[CN, MS],
...     formulation_=[0.002, 0.001],
...     )
>>> my_solution

```

Composition	Amount in kg	Amount in g
-----	-----	-----
Calcium nitrate tetrahydrate	0.002	2
Magnesium sulfate heptahydrate	0.001	1
Pure water	0.997	997
Total:	1	1000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO3-)	0.0002372	237.2
Mg	9.86e-05	98.6
Ca	0.0003394	339.4
S	0.0001301	130.1

2.2 Operations on solutions

The available operations on solutions can be split into two categories.

2.2.1 Operations preserving compositions

Any solution can be multiplied by a scalar. Two solutions defined in the same basis (i.e. consisting of the same compositions listed in the same order) can be added (and hence subtracted):

```
>>> 100 * my_solution
```

Composition	Amount in kg	Amount in g
Calcium nitrate tetrahydrate	0.2	200
Magnesium sulfate heptahydrate	0.1	100
Pure water	99.7	99700
Total:	100	100000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
N (NO3-)	0.0002372	237.2
Mg	9.86e-05	98.6
Ca	0.0003394	339.4
S	0.0001301	130.1

```
>>> solution_CN_20 = Solution.dissolve(1, water, [CN], [0.2])
>>> 5 * solution_CN_20 + 10 * solution_CN_20
```

Composition	Amount in kg	Amount in g
Calcium nitrate tetrahydrate	3	3000
Pure water	12	12000
Total:	15	15000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
N (NO3-)	0.02372	23720
Ca	0.03394	33940

Another operation preserving the compositions is `align()`. It adjusts the total mass of the solution to the specified value by changing the amount of the last composition (typically water):

```
>>> solution_CN_20.align(10)
>>> solution_CN_20
```

Composition	Amount in kg	Amount in g
Calcium nitrate tetrahydrate	0.2	200
Pure water	9.8	9800
Total:	10	10000

(continues on next page)

(continued from previous page)

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO ₃ -)	0.002372	2372
Ca	0.003394	3394

2.2.2 Operations extending compositions

An existing solution can be modified by adding another composition in the specified amount:

```
>>> MAP = Composition.from_dict(
...     {'Monoammonium phosphate':
...      {'N (NH4+)': 0.12177, 'P': 0.26928}}
... )
>>> my_solution.add(MAP, 0.001)
>>> my_solution
```

Composition	Amount in kg	Amount in g
-----	-----	-----
Calcium nitrate tetrahydrate	0.002	2
Magnesium sulfate heptahydrate	0.001	1
Monoammonium phosphate	0.001	1
Pure water	0.996	996
Total:	1	1000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO ₃ -)	0.0002372	237.2
N (NH ₄ ⁺)	0.00012177	121.77
P	0.00026928	269.28
Mg	9.86e-05	98.6
Ca	0.0003394	339.4
S	0.0001301	130.1

This operation does not return a new solution but always modifies the given one in place. Notice that by default the aligning operation is performed when add is called.

Any solutions can be merged which will result in a new solution:

```
>>> solution_a = Solution.dissolve(1, water, [CN], [0.002])
>>> solution_b = Solution.dissolve(1, water, [MS, MAP], [0.001, 0.001])
>>> solution_a.merge(solution_b)
```

Composition	Amount in kg	Amount in g
-----	-----	-----
Calcium nitrate tetrahydrate	0.002	2
Magnesium sulfate heptahydrate	0.001	1
Monoammonium phosphate	0.001	1
Pure water	1.996	1996
Total:	2	2000

(continues on next page)

(continued from previous page)

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO3-)	0.0001186	118.6
N (NH4+)	6.0885e-05	60.885
P	0.00013464	134.64
Mg	4.93e-05	49.3
Ca	0.0001697	169.7
S	6.505e-05	65.05

2.3 Correcting solutions

2.3.1 Adjusting the pH level

It is a common task to adjust the pH level of an existing nutrient solution by adding some acid (typically either nitric acid or phosphoric acid) or some base (typically potassium hydroxide). For this purpose one needs to weight the pH corrector and add it to the solution:

```
>>> solution_ms = Solution.dissolve(1, water, [MS], [0.002])
>>> KOH_94 = 0.94 * Composition.from_dict(
...     {'Potassium hydroxide': {'K': 0.69687}}
... )
>>> solution_ms.add(KOH_94, 0.000120)
>>> solution_ms
```

Composition	Amount in kg	Amount in g
-----	-----	-----
Magnesium sulfate heptahydrate	0.002	2
0.94 * (Potassium hydroxide)	0.00012	0.12
Pure water	0.99788	997.88
Total:	1	1000

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
K	7.86069e-05	78.6069
Mg	0.0001972	197.2
S	0.0002602	260.2

OPTIMIZING SOLUTIONS

3.1 Using optimizer

Hydrosolver includes mathematical optimization for solutions based on projected gradient descent method on a simplex. The following example utilizes a high-level enduser interface `hydrosolver.optimization.optimize` which takes over the formulation of the optimization problem with the standard weighted least squares objective functional and runs the optimization process with default parameters.

```
>>> from hydrosolver.solution import Solution
>>> from hydrosolver.composition import Composition
>>> from hydrosolver.optimization import optimize
>>> from hydrosolver.database import pure, compo, chelates, howard_resh
```

```
>>> composition_target = howard_resh['Resh composition for peppers']
```

```
>>> compositions = [
...     compo['Hakaphos Basis 2'],
...     pure['Calcium-ammonium nitrate decahydrate'],
...     pure['Magnesium sulfate heptahydrate'],
...     chelates['Fe-EDTA 13.3%'],
...     chelates['Zn-EDTA 15%'],
...     pure['Boric acid'],
... ]
>>> solution_init = Solution.dissolve(
...     150,
...     Composition(name='RO water'),
...     compositions,
... )
>>> solution_optimal = optimize(solution_init, composition_target)
>>> solution_optimal
```

Composition	Amount in kg	Amount in g
Hakaphos Basis 2	0.153874	153.874
Calcium-ammonium nitrate decahydrate	0.148834	148.834
Magnesium sulfate heptahydrate	0.0579563	57.9563
Fe-EDTA 13.3%	0.00390307	3.90307
Zn-EDTA 15%	0.000175686	0.175686
Boric acid	0.000194851	0.194851
RO water	149.635	149635
Total:	150	150000

(continues on next page)

(continued from previous page)

Composition: Resulting composition

Nutrient	Ratio	Amount mg/kg
-----	-----	-----
N (NO3-)	0.000172246	172.246
N (NH4+)	1.28593e-05	12.8593
P	4.02928e-05	40.2928
K	0.000340636	340.636
Mg	6.28412e-05	62.8412
Ca	0.000184008	184.008
S	5.02674e-05	50.2674
Fe	4.99947e-06	4.99947
Zn	3.2956e-07	0.32956
B	3.29649e-07	0.329649
Mn	5.12913e-07	0.512913
Cu	2.05165e-07	0.205165
Mo	1.02583e-08	0.0102583