hydrosolver

Dmytro Strelnikov

Sep 20, 2021

CONTENTS:

CHAPTER

ONE

WORKING WITH COMPOSITIONS

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

Defining a composition

The most straightforward way to define a compostion 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 defied as follows.

```python from hydrosolver import Composition

**MKP = Composition(** name='Monopotassium phosphate', vector='[0, 0, 0.2276, 0.2837, 0, 0, 0, 0, 0, 0, 0, 0, 0]' )
```

Here *vector* follows the structure of *composition.nutrients_stencil*.

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 *from_dict* which works as follows.

```python MKP = Composition.from_dict(

    {'Monopotassium phosphate': {'P': 0.2276, 'K': 0.2873}} )
```

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.

```python 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.

```python from hydrosolver.utils import load_file

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

## Operations on compositions

Compositions can be added and scaled, i.e. multiplied by scalars. Consider the following use case.

```python KOH = Composition.from_dict(

    {'Potassium hydroxide': {'K': 0.69687}} )

KOH_94 = 0.94 * KOH ```

CHAPTER

# TWO

# TEST

**Warning:** beware!

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search