MATH-C-HEM

PYTHON PACKAGE

FUNCTIONALITY

Calculates matrices:

- Adjacency
- Laplacian
- Signless Laplacian
- Normalized Laplacian
- Distance
- Resistance Distance
- Reciprocal Distance

```
>>> import mathchem as mc
>>> m = mc.Mol('GhCH?_')
>>> m.distance_matrix()

matrix([[0, 1, 2, 3, 4, 5, 3, 4],
        [1, 0, 1, 2, 3, 4, 2, 3],
        [2, 1, 0, 1, 2, 3, 1, 2],
        [3, 2, 1, 0, 1, 2, 2, 1],
        [4, 3, 2, 1, 0, 1, 3, 2],
        [5, 4, 3, 2, 1, 0, 4, 3],
        [3, 2, 1, 2, 3, 4, 0, 3],
        [4, 3, 2, 1, 2, 3, 3, 0]])
```

Calculates topological indices:

- Spectrum of the all matrices above
- Spectral moments
- Energy
- Zagreb M1 Index
- Zagreb M2 Index
- Connectivity index (R)
- Eccentric Connectivity Index
- Randic Index
- Atom-Bond Connectivity Index (ABC)
- Estrada Index (EE) for all matrices
- Distance Estrada Index
- Distance Degree (DD)
- Reverse Distance Degree (rDD)
- (Schultz) Molecular Topological Index (MTI)
- Distance Sum
- Balaban J index
- Kirchhoff Index (Kf) or Resistance
- Wiener Index (W)
- Reverse Wiener Index (RW)
- Hyper-Wiener Index (WW)
- Harary Index (H)

```
>>> import mathchem as mc
>>> m = mc.Mol('GhCH?_')
>>> m.reverse_wiener_index()
72

>>> m.spectrum('distance') # distance matrix used

[17.675869817881818, -0.4268447865902264, -0.5999662634461097,
 -0.8565662710452482,-1.4606244785164448, -2.744728088663583,
 -3.615279075919263, -7.971860853700944]
```

Calculates graph properties:

- Order
- Diameter
- Degree
- Eccentricity
- Connectedness

```
>>> import mathchem as mc
>>> m = mc.Mol('GhCH?_')
>>> m.degrees()

[1, 2, 3, 3, 2, 1, 1, 1]
```

WHAT IS MATHCHEM?

Mathchem is a free open source Python package for calculating topological indices and other invariants of molecular graphs. Currently it has version 0.1.0. This means that the package is on a very early stage of development. It was not fully tested and any feedback is welcome.

The package was tested under Mac OS X. Since the package contains no compiled code it is cross-platform and could be used in any operating system compatible with Python.

USAGE

After successful installation you can immediately use **mathchem** in Python or Sage

```
$ python
>>> import mathchem as mc
>>> m = mc.Mol('GhCH?_')
>>> m

Molecular graph on 8 vertices

>>> m.laplacian_matrix()
```

STRUCTURE

Currently the package consists of two modules: **mathchem** and **utilities**.

The first one contains the class *Mol*. In a current state the only way to initialize the Mol instance object with structure data is to give a *graph6* string as an argument: `m = mc.Mol('GhCH?_')`. Support of *SMILES* and *SDF* formats will be added later.

The second module **utilities** contains some useful functions. Currently there is only one function `batch_process(infile, outfile, function)` which allows to easily read a text file sting by string, pass read data to a function and write returned value to another file. Here is an example of use:

```
import mathchem as mc
import mathchem.utilities as u

def calculate_laplacian_energy(s):
    m = mc.Mol(s)
    return m.energy('laplacian')

u.batch_process('graphs.g6', 'result.txt', calculate_laplacian_energy)
```

After executing this code we can find calculated laplacian energy for all graphs listed in *graphs.g6* file as *graph6* strings in *result.txt*. In fact, we just applied function `calculate_laplacian_energy` to each line of file *graphs.g6* and wrote results in *result.txt*.

CONCEPT OF LAZY CALCULATIONS

For performance reasons we calculate data only when it is needed and then we save the results.

For instance, we want to calculate three indices based on distance, let say, *Eccentric Connectivity Index*, *Balaban J index* and *Wiener Index*. This means that we need a *distance matrix*, which we calculate only once and then use it:

```
>>> import mathchem as mc
>>> m = mc.Mol('GhCH?_')
>>> print m.eccentric_connectivity_index(), m.balaban_j_index(), m.wiener_index()
    52  3.29247815608  68
```

The *distance matrix* will be calculated only once - before calculating the first index.

INSTALLATION

As Python module

For any UNIX-like system the installation process is trivial:

```
$ pip install mathchem
```

Depends on: numpy

As Sage module

Download **spkg** file from
github.com/downloads/hamster3d/mathchem-package/mathchem-0.1.0.spkg

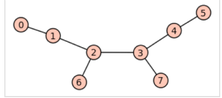
Save it into your **sage** directory

Run sage with a command to install a new package:

```
$ sage -f mathchem-0.1.0.spkg
```

After that you can use **mathchem** in your sage programs:

```
sage: import mathchem as mc
sage: m = mc.Mol('GhCH?_')
sage: s = m.sage_graph()
sage: s.show()
```



SAGE

Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface.

Mission: Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab.

www.sagemath.org

ABOUT

Mathchem package written by Alexander Vasilyev, PhD student of University of Primorska under supervision of prof. Dragan Stevanović.

All contacts details can be found at the homepage of the project:

github.com/hamster3d/mathchem-package

UNIVERZA NA PRIMORSKEM
UNIVERSITÀ DEL LITORALE

github SOCIAL CODING

http://goo.gl/76WNF