Topoly workshop June 25, 2021

Bartosz Ambroży Greń

Short introduction

Topoly website: https://topoly.cent.uw.edu.pl.
Topoly website at PyPI: https://pypi.org/project/topoly
According to the PyPI website:
"Topoly is a Python package that collects programs useful for polymer topology analysis.

What you can do with Topoly?

- Find knots, links, lassos, theta-curves, handcuffs and their type.
- Calculate knot/link invariants:
  - Polynomials: Alexander, Jones, Conway, HOMFLY, Yamada, Kauffman, BLM/Ho,
  - Brackets: Kauffman, APS,
  - Other: writhe, Gaussian linking number.
  - Find minimal surface of a loop.
- Simplify polymer structure preserving its topology.
- Generate random polygon structures: walks, loops, lassos, handcuffs.
- Generate knot map (like in KnotProt).
- Calculate sum (U) and product (#) of knots.
- Visualize structures."

Installation

Topoly is supported from Python 3.5 to higher versions. It is a standard Python package, therefore the easiest way to install it using `pip`.

```
pip3 install --upgrade pip
pip3 install topoly
```

We encourage to check Topoly tutorial github repository https://github.com/ilbsm/topoly_tutorial and tutorial at Topoly website https://topoly.cent.uw.edu.pl/tutorial.html.

Usage

Here we will focus on generations of random structures (knots, links, lassos, handcuffs), their topological analysis using Yamada polynomial and calculating polynomials of unions and sums of known structures.

Generation of chain and identification

All functions generate some combination of open and closed equilateral polygons. Open polygons are generated with random walks. Closed polygons are generated using Jason Cantarellas algorithm described in his work titled ”A fast direct sampling algorithm for equilateral closed polygons”.

1. Start with opening python or ipython terminal and writing:

   ```
   import topoly
   ```

2. Lets generate five structures of loops (knots) with length of 20 segments

   ```
   topoly.generate_loop(20,3)
   ```

   In your current location a folder ”`l020`” will be created. Inside this folder there are 5 files: `loop00000.xyz`, `loop00001.xyz` and `loop00002.xyz`. Open any file and check its format.

3. Similarly you can try with other functions: `topoly.generate_link()`, `topoly.generate_lasso()`, `topoly.generate_handcuff()`. E.g.:

   ```
   topoly.generate_lasso(20,30,3)
   topoly.generate_handcuff([20,10],30,3)
   topoly.generate_link([20,10],30,3)
   ```

   Three folders will be created:

   - `l020_t030` with lassos composed of a loop with length 20, and a tail with length 30.
   - `l020_010_t020` with two loops of length 20 and 10, connected by a linker (random walk) of length 30. It can be seen as a lasso with connected loop at the end of its tail.
   - `l020_010_d020` with two loops of length 20 and 10, displaced by a random walk of length 30. It can be seen as a handcuff with linker removed.

   Please open any of created files. Notice that alone `X` sign in a line separates different arcs in a chain and that first and last node in a arc is trivalent.

4. You can change names of created folders and files. To change prefixes use `folder_prefix` and `file_prefix` parameters. To change zero-padding format use `file_fmt` parameter. Play with it e.g. by writing:

   ```
   generate_lasso(20, 30, 3, file_prefix='file_', folder_prefix='folder_',
                  file_fmt=(2,2,1))
   ```

5. For more check documentation:
   https://topoly.cent.uw.edu.pl/documentation.html#random-polygons-generation

6. You can calculate Yamada polynomial to check topology of a given structure by passing filename and its path, e.g.:

   ```
   topoly.yamada('l020_010_t030/hdcf00001.xyz', closure=0)
   ```

   Value `closure=0` is important to inform the program that the chain is already closed. Defaultly program closes every chain in a probabilistic way (200 times).

Unions and sums of knots

1. Lets start with loading Yamada polynomial values of 3_1 knot:

   ```
   topoly.getpoly('Yamada', '3_1')
   ```

   `topoly.getpoly()` returns a list of objects. Lets check fields of these objects:

   ```
   plus31, minus31 = topoly.getpoly('Yamada', '3_1')
   plus31.name
   plus31.inv
   plus31.val
   plus31.val(1)
   ```

   field `val` returns polynomial object from numpy.polynomial package. Passing a number to `val` you can calculate polynomial value at a given point.
2. You can easily calculate sums (unjoined union), and products (conjoined union) of objects created by `topoly.getpoly`, e.g.:

   ```
   plus31 + minus31
   plus31 * minus31
   ```

   In the case of Yamada polynomial there is one more operation possible:

   ```
   plus31 ** minus31
   ```

   which calculates polynomial of conjoined union of two topologies by connecting them by trivalent points. Of course, 3_1 knot does not have such points, therefore try again with handcuffs:

   ```
   plush21, minush21 = topoly.getpoly('Yamada', 'h2_1')
   plush31 ** minush21
   ```