TWISTED ALEXANDER POLYNOMIALS OF HYPERBOLIC KNOTS: DATA AND SOFTWARE

NATHAN M. DUNFIELD, STEFAN FRIEDL, AND NICHOLAS JACKSON

ABSTRACT. This document describes how to use the raw data and software we created when writing [DFJ].

CONTENTS

1. SOURCE

The all the files, including these instructions, are available at `http://dunfield.info/torsion/` in the tarball Genus-Comp.tar.gz.

2. RAW DATA

A complete table of $\mathcal{T}_K$ for all knots with at most 15 crossings is included in subdirectory `knotdata` in the files `AlternatingKnots.gz` and `NonalternatingKnots.gz`. Once uncompressed, the format of each line is

$$\text{knot\_name} \qquad \mathcal{T}_K(t) \qquad \Delta_K(t) \qquad \big(\deg(\mathcal{T}_K),\ \text{monic}(\mathcal{T}_K),\ \deg(\Delta_K),\ \text{monic}(\Delta_K)\big)$$

where the four main parts are separated by tabs. While $\mathcal{T}_K$ was computed to 250 digits, only 40 were saved to conserve space.

3. Accessing table data from within Sage

If you have Sage [Sage], here's an easy way to access our table. Begin by starting Sage in the main `Genus-Comp` directory, and then do:

```
sage: ls
conwayKT/ knotdata/ misc/ taskdb/
hypertorsion/ knotgenus/ t3mlite/
sage: import knotdata
sage: data = [d for d in knotdata.data("Test", return_polys=True)]
sage: len(data)
193
sage: knot_name, T_K_data, D_K_data, T_K, D_K = data[0]
sage: knot_name
'8n1'
sage: T_K
t^6 - 2.000000000000000000000000000000000000*t^5
+ (-0.797293258421134089291295944133695953 98 + 0.047347098481845310165745577139987827599*I)*t^4
 + (4.709511025606673812450272570447590111 6 - 1.003651980709914336555553378039249372 2*I)*t^3
+ (-0.797293258421134089291295944133695953 98 + 0.047347098481845310165745577139987827599*I)*t^2
- 2.000000000000000000000000000000000000*t + 1.000000000000000000000000000000000000
sage: D_K
t^4 - 2*t^3 + 3*t^2 - 2*t + 1
```

The above example uses a small sample table, so substitute `AlternatingKnots` or `NonalternatingKnots` for `Test` as the first argument for `knotdata.data` to get the full tables. You can also easily get at Stoimenow's data [Sto] on mutant knots:

```
sage: mutant_groups = knotdata.mutants()
sage: mutant_groups[0]
['11a19', '11a25']
sage: group_sizes = [len(group) for group in mutant_groups]
sage: max(group_sizes)
8
sage: [ (s, group_sizes.count(s)) for s in range(1, 9)]
[(1, 0), (2, 29590), (3, 1276), (4, 3203), (5, 0), (6, 190), (7, 0), (8, 90)]
```

4. Using Sage to compute a new $\mathcal{T}_K$: the hypertorsion module

If you want to know $\mathcal{T}_K$ for some other knot (or one-cusped manifold), you need to have SnapPy [CDW] installed within Sage. See here for detailed instructions on installing SnapPy, and please note that for this part you need Sage version 4.7 or newer.

As a quick check that everything is working, the following computes $\mathcal{T}_K$ for a small sample of knots and compares the result to what's stored in the table. At the command-line in the `Genus-Comp` directory, do:

```
topo ~/a/Genus-Comp % sage -python -m hypertorsion.test
8n1 [0, 0]
8n2 [0, 0]
9n1 [0, 0]
9n2 [0, 0]
9n3 [0, 0]
```

```
9n4 [0, 0]
9n5 [0, 0]
[...]
```

Here, $[0,0]$ means the answer it got for $\mathcal{T}_K$ agrees with the table, both when it used our conventions and also when it used the conventions of Wada's invariant (see Section 2.20 of [DFJ] for the nominal difference here).

Now, here's how to compute $\mathcal{T}_K$ for the knot 5_2 to 100 decimal digits.

```
sage: import snappy, hypertorsion
sage: M = snappy.Manifold("5_2")
sage: T_K = hypertorsion.hyperbolic_torsion(M, 100)
sage: T_K
(2.337641021377626987019545572760951329632797971549133317732992474848586074377226202972650326009106358
- 0.562279512062301243899182144909373061497843002895783939838390456724843663609050834995941556088215512 6*I)*a^2
- 3.999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999997*a
+ 2.337641021377626987019545572760951329632797971549133317732992474848586074377226202972650326009106359
- 0.562279512062301243899182144909373061497843002895783939838390456724843663609050834995941556088215509 9*I
sage: hypertorsion.alexander_polynomial(M)
2*a^2 - 3*a + 2
```

4.1. **Adjoint-based torsion polynomial.** The program can also compute the polynomial $\mathcal{T}_K^{adj}$ discussed in Section 5.2 of [DFJ]. Here, because the final representation has odd dimension, there are more signs to worry about and we just ignore these. As a result, the result is only defined up to a multiple of $\pm a^k$. Also, these torsion polynomials depend on a implicit choice that SnapPy makes for the knot orientation. To continue the previous example:

```
sage: T_adj = hypertorsion.hyperbolic_adjoint_torsion(M, 100)
sage: T_adj
(0.404313580736184811971326605314723373363414731576259873057314071251736357619176278847625788805824909 2
+ 0.759394515009716502410387722428707511511045841769486610949866642590544218760523804401478714486933720 5*I)*a^3
+ (2.903284961389108302142027884744972323744091138181196478392596834414556773827397763205015502475763170
- 4.118538838993551699988263298060630736227221102281528213232190332503477199492144777766793091462257693*I)*a^2
+ (-2.903284961389108302142027884744972323744091138181196478392596834414556773827397763205015502475761709
+ 4.118538838993551699988263298060630736227221102281528213232190332503477199492144777766793091462256964*I)*a
- 0.404313580736184811971326605314723373363414731576259873057314071251736357619176278847625788805826184 1
- 0.759394515009716502410387722428707511511045841769486610949866642590544218760523804401478714486928943 3*I
```

4.2. $\mathrm{SL}(n,\mathbb{C})$ **torsion polynomial.** For each n, there is a unique irreducible n-representation complex representation of $\mathrm{SL}(2,\mathbb{C})$. Composing the holonomy representation with this gives another torsion polynomial depending on the hyperbolic structure; here the above two polynomials are $n=2$ and $n=3$. This package can compute these polynomials as well; here's an example with $n=4$.

```
sage: T_4 = hypertorsion.hyperbolic_SLN_torsion(M, 4, 100)
sage: R.<a> = PolynomialRing(CC)
sage: R(T_4) # Reduces precision for printing
(2.58909881840993 + 3.51266108176133*I)*a^4 + (10.3574038239393 - 13.3784463023755*I)*a^3
+ (-26.8218023631802 + 7.02532216352267*I)*a^2 + (10.3574038239393 - 13.3784463023755*I)*a
+ 2.58909881840993 + 3.51266108176133*I
```

4.3. **Safety checks.** Internally, the `hyperbolic_torsion` function builds the twisted chain complex described in Proposition 2.15 of [DFJ], and checks that ∂^2 is 0 to within a reasonable accuracy given the working precision.

5. Genus and fibering: the knotgenus module

As discussed in Section 6 of [DFJ], building on [DR] we developed tools to find upper bounds on the genus and to find certificates that a (closed) 3-manifold fibers over the circle. This `knotgenus` module unfortunately requires two compiled submodules (at least for the fibering test).

5.1. **Installing fpgroup.** The first, `fpgroup` should be easy to install:

```
topo ~/a/Genus-Comp % cd knotgenus/fpgroup
topo ~/a/Genus-Comp/knotgenus/fpgroup % sage -python setup.py install
```

5.2. **Installing pyregina.** The second, `pyregina` is a wrapper of a small part of Regina [Bur]. You will need to have the basic (non-GUI, non-Python) version of Regina installed. For Linux, you should be able to get a prebuilt package of Regina. For OS X, here is one sample of how to install Regina into a subdirectory of a directory called `/pkgs`, when `libxml2`, and `gmp` have been installed in `/opt` via MacPorts. If you are using 10.5 or earlier or have a 32bit version of Sage, you should delete the configure option `architecture=i686`.

```
topo ~ % cd /pkgs
topo /pkgs (2) wget http://sourceforge.net/projects/regina/files/regina/4.6/regina-4.6.tar.gz/
[...]
topo /pkgs % tar xfz regina-4.6.tar.gz
topo /pkgs % cd regina-4.6/
topo /pkgs/regina-4.6 % mkdir local
topo /pkgs/regina-4.6 % ./configure --disable-kdeui --prefix=/pkgs/regina-4.6/local \
   --with-extra-includes=/opt/local/include   --with-extra-libs=/opt/local/lib \
   architecture=i686
[...]
Good - your configure finished.  Start make now.
topo /pkgs/regina-4.6 % make
topo /pkgs/regina-4.6 % make install
```

Next edit `knotgenus/regina_wrap/setup.py` if necessary to reflect the locations of the Regina library, libxml2 and libgmp. You should now be able to install `pyregina` via `sage -python setup.py install` from within the `knotgenus/regina_wrap/` directory. You can then test it with:

```
topo ~/a/Genus-Comp/knotgenus/regina_wrap % sage -python test.py
4.6
8
(3, [[-1, -1, -2, -2, 3, 3, -2, -2, -2], [2, 2, 2, 1, 2, 3, 3, -2,-2,-2, 1],
     [3, 3, 3, -2, -2, -2]])
```

5.3. **Testing.** Here's a quick test to make sure all the various pieces work together correctly:

```
topo ~/a/Genus-Comp % sage -python -m knotgenus.test
10n1.tri True
10n10.tri True
[...]
3_1_filled 1 1 True
4_1_filled 1 1 True
5_1_filled 2 2 True
5_2_filled 1 1 False
[...]
```

You can look at the file `knotgenus/test.py` to see what it's doing here.

5.4. **An example.** We'll look at the knot 8_{20} from Rolfsen's table. First, let's see what info $\mathscr{T}_K$ gives us:

```
sage: import snappy, hypertorsion, knotgenus
sage: M = snappy.Manifold('8_20')
sage: T_K = hypertorsion.hyperbolic_torsion(M, 100)
sage: T_K.degree(), T_K.leading_coefficient()
(6,0.99999999999999999999999999999999999999...)
```

Thus, we expect $x(8_{20}) = \deg(\mathscr{T}_K)/2 = 3$, that is, $\text{genus}(8_{20}) = 2$ and also that the knot is fibered.

Let's confirm both of these:

```
sage: genus_upperbd, tri, tri_t3m = knotgenus.genus_upper_bound(M, lower_bound=2)
sage: genus_upperbd
2
sage: knotgenus.is_fibered(M, lower_bound=2)
True
```

5.5. **Important note on rigor.** Both the `genus_upper_bound` and `is_fibered` methods assume they are given a knot in S^3. This is because they use Gabai's theorem on Property R to focus instead on the 0-surgery.

As the name suggests, `knotgenus.genus_upper_bound` is not guaranteed to compute the genus, but merely search through various cheap-to-construct Seifert surfaces and report back the smallest genus that it saw. (By default, it tries 100 different triangulations, though it will stop as soon as it finds a surface matching `lower_bound`).

Also, when `is_fibered` returns `False`, this does *not* mean the knot is non-fibered. In contrast, when it returns `True`, this *does* mean it's proved that the manifold fibers. In particular, it splits the 0-surgery open along the simplest homologically nontrivial surface it can find, and then simplifies the fundamental group as much as possible. What `False` means here is that the resulting presentation was not *obviously* that of a surface group.

6. Other files

The `misc` directory contains many of the scripts that were used to generate the data discussed in Sections 5–8. Some of them may not work (due to changes in the main modules subsequent to the script being written), but should still give you an idea of what can be done here. The `conwayKT` directory was used in writing Section 5. The `t3mlite` module is a slimmed down version of [CD], and the `taskdb` is a small module with was used to (coarsely) parallelize the computation of the main table across 12 processors on single machine.

7. Further details

Use the source, Luke! Sage uses Python as its main programming language, and this one of the most readable languages around. If you're new to Sage/Python, the file `__init__.py` usually contain the main source code for its module. Finally, you should email any and all questions or comments to Nathan at the below address.

REFERENCES

[Bur] B. Burton. Regina, a normal surface theory calculator, version 4.6, 2009. http://regina.sourceforge.net/

[CD] M. Culler and N. M. Dunfield. t3m: a Python 3-manifold library, 2010. http://t3m.computop.org

[CDW] M. Culler, N. M. Dunfield, and J. R. Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. http://snappy.computop.org/

[DFJ] N. M. Dunfield, S. Friedl, and N. Jackson. Twisted Alexander polynomials of hyperbolic knots, Preprint 2011.

[DR] N. M. Dunfield and D. Ramakrishnan. Increasing the number of fibered faces of arithmetic hyperbolic 3-manifolds. *Amer. J. Math.* **132** (2010), 53–97. arXiv:0712.3243, MR2597506.

[Sage] W. Stein et al. *Sage Mathematics Software, Version 4.7*. http://www.sagemath.org

[Sto] A. Stoimenow. Knot data tables. http://stoimenov.net/stoimeno/homepage/ptab/

UNIVERSITY OF ILLINOIS, URBANA, IL 61801, USA

E-mail address: nathan@dunfield.info

URL: http://dunfield.info

MATHEMATISCHES INSTITUT, UNIVERSITÄT ZU KÖLN, GERMANY

E-mail address: sfriedl@gmail.com

UNIVERSITY OF WARWICK, COVENTRY, UK

E-mail address: nicholas.jackson@warwick.ac.uk