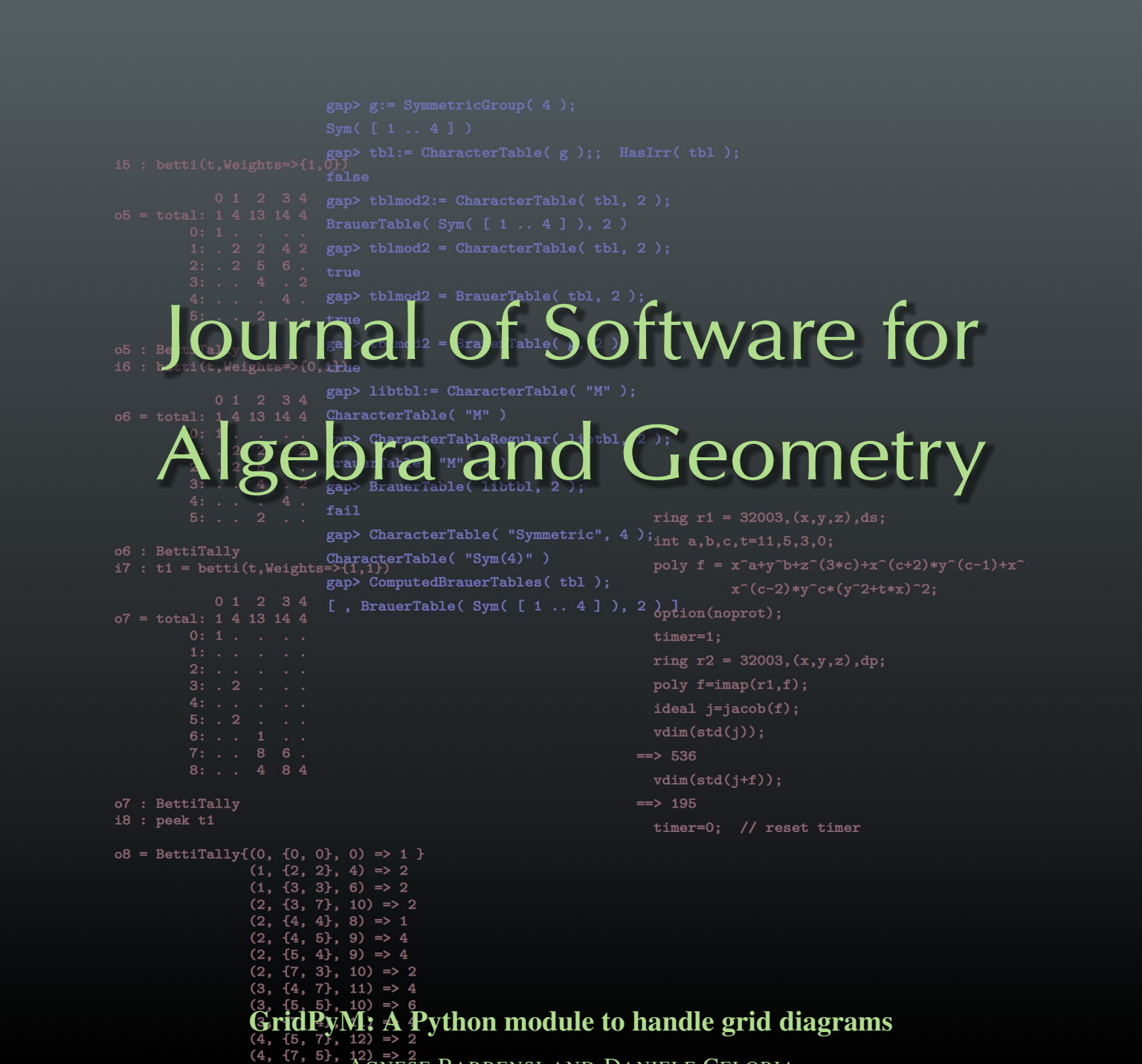

Journal of Software for Algebra and Geometry

GridPyM: A Python module to handle grid diagrams

Agnese Barbensi and Daniele Celoria

vol 14 2024

JSAG **14** (2024), 31–39
https://doi.org/10.2140/jsag.2024.14.31

The Journal of Software for
Algebra and Geometry

GridPyM: A Python module to handle grid diagrams

AGNESE BARBENSI AND DANIELE CELORIA

ABSTRACT: Grid diagrams are a combinatorial version of classical link diagrams, widely used in theoretical, computational and applied knot theory. Motivated by questions from (bio)-physical knot theory, we introduce GridPyM, a Sage compatible Python module that handles grid diagrams. GridPyM focuses on generating and simplifying grids, and on modelling local transformations between them.

1. INTRODUCTION. Grid diagrams are a handy and computationally efficient model for oriented link diagrams. They admit a concise description, can be easily randomised and (unlike standard link diagrams) are able to encode certain local geometric features of links. They were first introduced in a slightly different form in [16], and later on in [9; 13] under the name of *arc presentations*. The latter paper also introduced a set of combinatorial moves, commonly known as Cromwell moves, connecting any two grids representing equivalent link diagrams.

Grid diagrams are ubiquitous in theoretical and applied knot theory. Indeed, they arise naturally in contact geometry, as they represent Legendrian diagrams of links in S^3, endowed with its standard tight contact structure. Further, a certain subset of Cromwell moves bijectively corresponds to Reidemeister moves preserving the Legendrian isotopy class of the link [14; 19].

Grid diagrams can also be interpreted as representing links in a genus one Heegaard decomposition of the 3-sphere. Here, the grid is identified with the Heegaard torus, with horizontal and vertical lines acting as the cores and co-cores of the 1 and 2 handles. This is the ideal setup to define a combinatorial version of knot Floer homology [20; 24], known as *grid homology* [17]; see also [23] for a detailed reference, and [1] for a lens space version. Despite its straightforward definition, a direct computation of grid homology becomes quickly unfeasible, as the chain complex has a number of generators which grows factorially in the size of the grid. It is, however, possible to greatly simplify the grid homology chain complex [5], and there is a relatively efficient algorithm to compute it [2] implemented in the software GridLink [10]. Note that faster computation of knot Floer homology [22] can be achieved using more recent techniques developed in [21].

Thanks to their combinatorial definition, grid diagrams provide a convenient model to investigate on asymptotic properties of (bio)physical knots [12; 27]. As an example, they have been used to quantify the intensity of crossing change-mediated fluxes between different knot types, and to demonstrate the

MSC2020: primary 57K10; secondary 90-04.
Keywords: knot theory, low-dimensional topology, computational topology.
GridPyM version 1.2

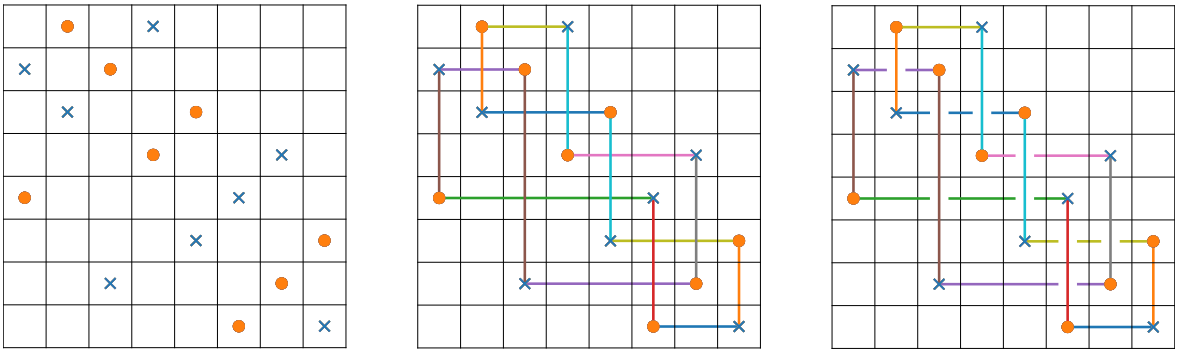

Figure 1. From left to right: an example of a grid diagram, and how to obtain an oriented link diagram from it by resolving each double point as a vertical overpass.

dependency of those fluxes on the local geometry of knotted configurations. In turn, this provided strong evidence that the simplification action of specific DNA enzymes is driven by a geometric selection of sites [4]. One further potential use of grid diagrams is to help with the search of band changes and the determination of Gordian distance between knot types [6; 7; 18; 26].

The most used program currently available to manipulate grid diagrams is GridLink. One of the main features of GridLink is a very efficient simplification function. Here we present GridPyM, a Sage [25] compatible Python module that aims at enhancing certain functionalities present in GridLink, and at extending its manipulation capabilities.

In particular, the module is built for generating large populations of complex grids to be employed for the statistical analysis of properties of the grid model. For this purpose, on one hand it is necessary to be able to quickly generate a large amount of random grid diagrams, possibly with specific properties (e.g., a fixed or bounded number of components). On the other hand, easy ways of efficiently simplifying large samples of grids are needed to accelerate invariants computations. Thus, GridPyM's focus is on generating, simplifying and randomising grids, rather than the computation of invariants, which can be carried out afterwards with efficient existing programs (such as [11; 22; 25]).

2. GRID DIAGRAMS. A *grid diagram* G is a $n \times n$ square plane grid, together with two sets of n markings, conventionally denoted by $\mathbb{X}$ and $\mathbb{O}$. Here $n \geq 2$ is called the *grid number* or dimension of G.

The grid is subdivided into n^2 little squares, and the markings are placed in these squares according to a "sudoku" rule, so that each row and column contains exactly one marking of each kind, and each square contains at most one marking. We adopt the convention according to which the markings are listed as ordered tuples $\mathbb{X} = [X_1, \ldots, X_n]$ and $\mathbb{O} = [O_1, \ldots, O_n]$, where the i-th component denotes the position (from the left, and starting from 0) of the marking on the i-th row, where rows are enumerated from the bottom up. For example, the grid in the left of Figure 1 is described by the pair of markings $\mathbb{X} = [7, 2, 4, 5, 6, 1, 0, 3]$ and $\mathbb{O} = [5, 6, 7, 0, 3, 4, 1, 2]$. Note that the combinatorial requirement on the markings implies that $\mathbb{X}$ and $\mathbb{O}$ are a pair of collision-free permutations of $\{0, \ldots, n-1\}$, where n is the grid number.

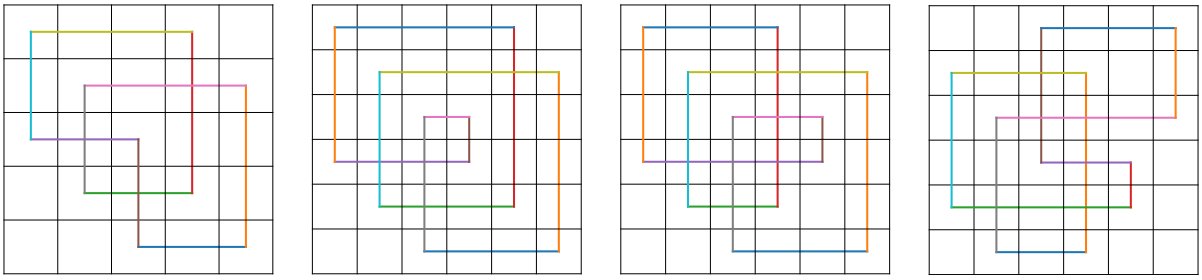

Figure 2. From left to right: a grid diagram representing the (right) trefoil, the effect of applying a stabilisation, the effect of an interleaving commutation on the stabilised diagram and a shift along the vertical axis.

To a grid diagram G we can associate an oriented link diagram as follows: on each row of the grid, connect with a horizontal segment the unique $\mathbb{O}$ marking to the only $\mathbb{X}$, and do the same on columns from $\mathbb{X}$ to $\mathbb{O}$. Resolving each double point, with the convention that vertical lines are always overpasses, yields a diagram representing a link $L(G)$. A schematic outline of the process is shown in Figure 1.

It is easy to realise that any oriented link can be represented by infinitely many grid diagrams. Further, there is a finite set of moves, known as Cromwell moves (see Figure 2 for some examples), with the property that two grid diagrams are related by a finite sequence of such moves if and only if the links they represent are isotopic. These moves are therefore a grid-diagrammatic analogue of the classical Reidemeister moves on link diagrams.

3. GridPyM: Module description and main functions.

In what follows, we give a brief description of the Python module GridPyM, and of some of the main functions implemented.

3.1. *Installation and use.*

GridPyM is stored in the "GridPythonModule" GitHub repository (see [3]), together with some Jupyter notebooks [15], interactively explaining basic usage and functions. We tried to keep the module as self-contained as possible. The only external imports required are the rather standard libraries `sympy`, `random2` and `matplotlib`. As a consequence, there is full compatibility with Sage, and GridPyM can be imported and used in conjunction with Sage's built-in link functions. The module works for all versions of Python greater than or equal to 3. The code will be periodically updated, and we refer to the GitHub repository for the most recent version. GridPyM can be downloaded from its GitHub repository. To install GridPyM, write in a terminal

```
pip install GridPythonModule
```

To import GridPyM and its functions in a Python environment / terminal, write

```
import GridPythonModule
from GridPythonModule import *
```

3.2. *Grid generation.*

GridPyM provides several different ways of generating grids, divided into:

- generating random grids with prescribed features (number of components and grid number) using the `generate_random_grid` commands,

- generating a grid representing specific link types (e.g., torus links),
- loading from a library of low crossing number knots using `load_knot`,
- loading grids representing low crossing number Legendrian knots (taken from the Legendrian knot atlas [8]), using `load_legendrian_knot`.

A list of available knot types can be accessed via, respectively, the commands `available_knots` and `available_legendrian_knots`.

3.3. ***Grid handling and moves.*** GridPyM includes all standard Cromwell moves (cyclic shifts, noninterleaving column/row commutations and (de)stabilisations, see Figure 2). Rows and columns commutations might change the underlying link type of the diagram, depending on the local combinatorics. We include those resulting in a crossing changes [23, Proposition 3.1.13], and band attachments. Band attachments are divided into *coherent* or *uncoherent*, depending on whether they change or not the number of components of the link. Examples on how to perform these moves are shown in the online supplement.

It is also possible to perform some other operations to generate new links from given ones, such as disjoint union, connected sum, and some cables, e.g., taking parallel copies of a knot (see Figure 3 for an example). Other standard knot-theoretic functions include inverting the orientation, taking the mirror image and rotating the grid (note that rotating the grid by $\frac{\pi}{2}$ produces a representative for the mirror of the link). See the online supplement for further examples.

3.4. ***Grid simplification and randomisation.*** The simplest way to reduce the size of a grid diagram is to perform a destabilisation. We implemented the function `destabilise_all` to simplify a grid by recursively destabilising all nontrivial configurations in which two markings are adjacent (described in Figure 4).

The function `destabilise_all` is the key component of the main simplification function of GridPyM: `simplify_grid`. The `simplify_grid` function takes a grid diagram as input and tries (with customisable levels of effort) to simplify it, by performing destabilisations and a (customisable) number of random noninterleaving commutations and cyclic shifts. Note that by the *monotonic simplification theorem* [13], this process is (with probability 1, in the absence of computational issues) guaranteed to return the 2×2 grid diagram, whenever $L(G)$ represents the unknot. See the online supplement for examples. Further options include restricting to Cromwell moves that preserve the Legendrian or transverse class (so that the input and simplified grid represent the same Legendrian or transverse knot type).

The function `scramble_grid` provides the opposite operation: given a grid, it performs a customisable amount of random moves to it, making it on average more complex. Again, there is the option to preserve the Legendrian or transverse isotopy class.

3.5. ***Grid and contact knot invariants.*** As mentioned before, the suite of link invariants was kept to a minimum. We have thus only included invariants that can be efficiently computed from the grid, and that are not easily computable by other compatible programs. In particular, the only available topological invariant of the link type is the number of components. All other functions mentioned below are instead

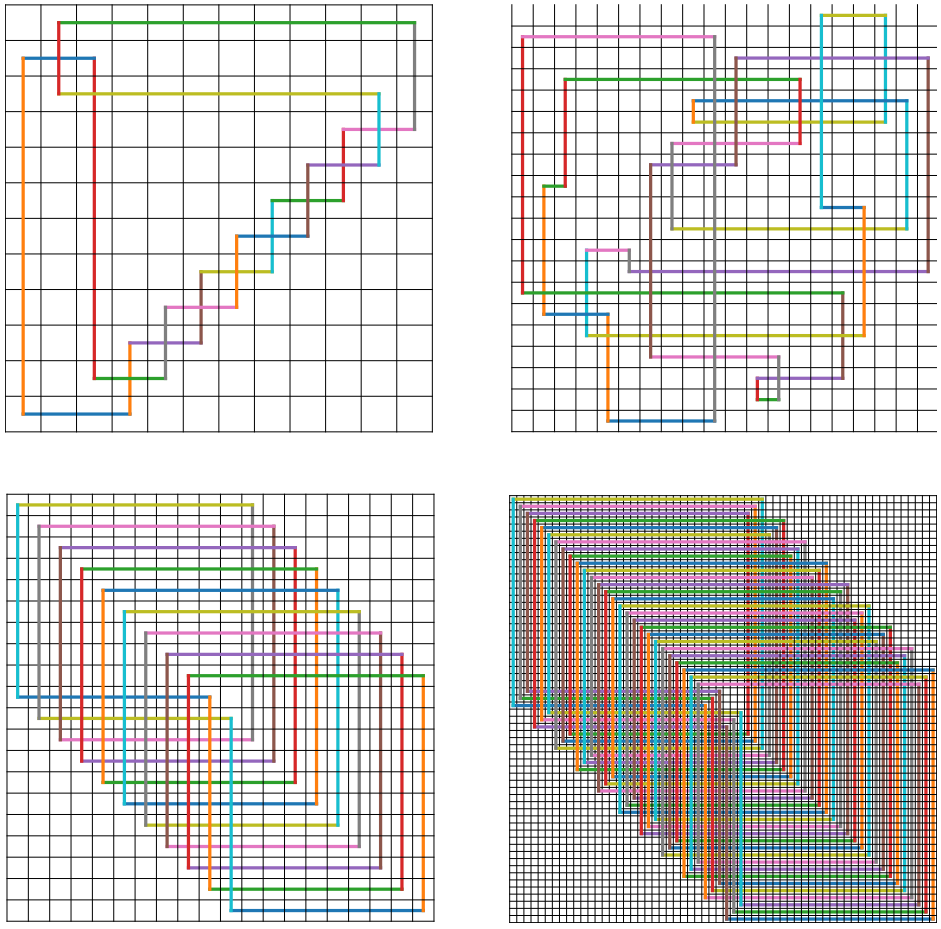

Figure 3. Some examples of grid diagrams obtained with GridPyM's drawing function `draw_grid` (from top to bottom and left to right): the 8 crossings negatively clasped twist knot, a random knot in grid number 20, the (11, 9) torus knot and its flat (3, 1) cable.

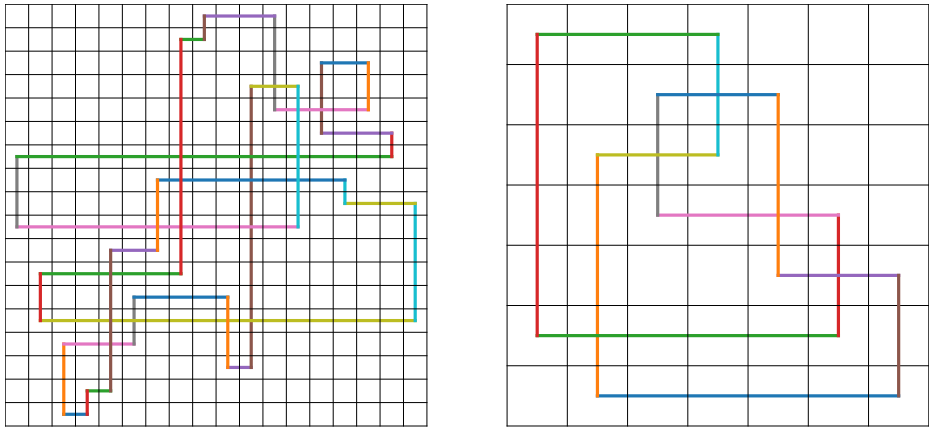

Figure 4. The effect of the `destabilize_all` function on a scrambled grid representing the 5_2 knot. This function performs all possible "generalised" destabilisations on the grid.

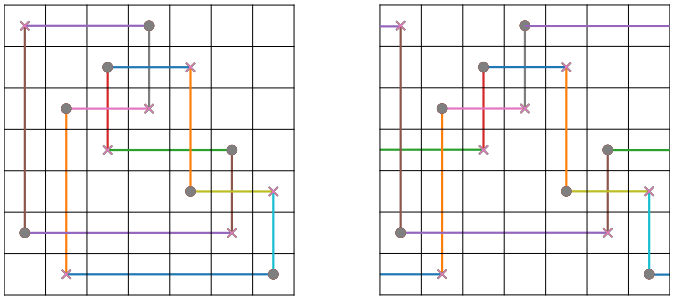

Figure 5. A schematic representation showing one way to associate a braid diagram to a grid diagram.

either invariants of the Legendrian or transverse class, or combinatorial invariants of the grid. In the former category we have the Thurston–Bennequin number, self-linking and rotation number. In the latter we have the grid number (so just the dimension of the given grid), crossing number, and the grid length; this is just the sum of the lengths of the vertical and horizontal segments forming the link.

It is also possible to convert a given grid diagram into a braid (see Figure 5); this is done using the `convert_to_braid` function, whose output is a string containing the standard generators of a braid whose closure represents the link. For example, $[1, 1, 1]$ is the (right) trefoil, while $[1, -2, 1, -2]$ is the figure-eight knot. Note that there is an option to automatically choose a presimplification function; in other words, the function first attempts to simplify the grid before computing the associated braid word.

Another way of passing from grid diagrams to knots is through Gauss codes. The output of the Gauss code and braid conversion functions are chosen to be compatible with Sage, as shown in the example below. Note that the following code needs to be run on Sage:

```
>> G = load_knot('7_3')

>> braid_word = convert_to_braid(G)

>> print(braid_word)
 [5, 4, 3, 2, 5, 4, 3, 2, 1, -2, 1]

>> B = BraidGroup(6)

>> K = Knot(B(braid_word))

>> print(K.alexander_polynomial())
2*t^-2 - 3*t^-1 + 3 - 3*t + 2*t^2

>> g_code = Gauss_code(G)

>> print(g_code)
[[[1, 2, -3, 4, -2, -5, 6, -7, 8, -1, -4, 3, 5, -6, 7, -8]],
[1, -1, 1, 1, 1, 1, 1, 1]]

>> L = Knot(g_code)

>> show(L.plot())

>> print(L.jones_polynomial())
-t^9 + t^8 - 2*t^7 + 3*t^6 - 2*t^5 + 2*t^4 - t^3 + t^2
```

4. SAMPLE COMPUTATIONS. In this final section, we collect some basic results obtained with GridPyM, mostly as a check of some of its functionalities. We analyse the distribution of several invariants for randomly generated grids in the grid number range 5–100 (see Figures 6–8). For related results, see [4; 12].

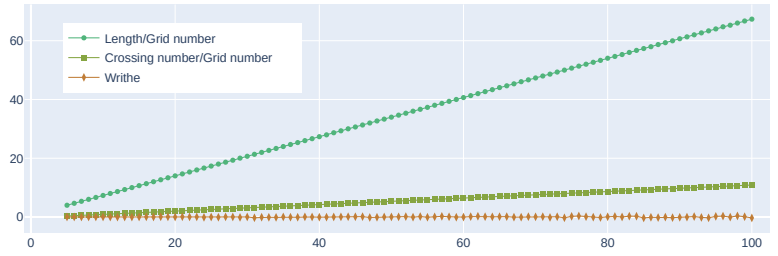

Figure 6. A plot showing the growth of the average length of a grid divided by the grid's dimension, of the average crossing number divided by the grid's dimension and of the average writhe (with respect to the grid number). As expected, the latter is distributed along the $y = 0$ line. On the other hand, both the crossing number and the length exhibit quadratic growth as functions of the grid number. The slope of the linear fit for length / grid number is ~ 0.6667 and ~ 0.1111 for crossing number / grid number.

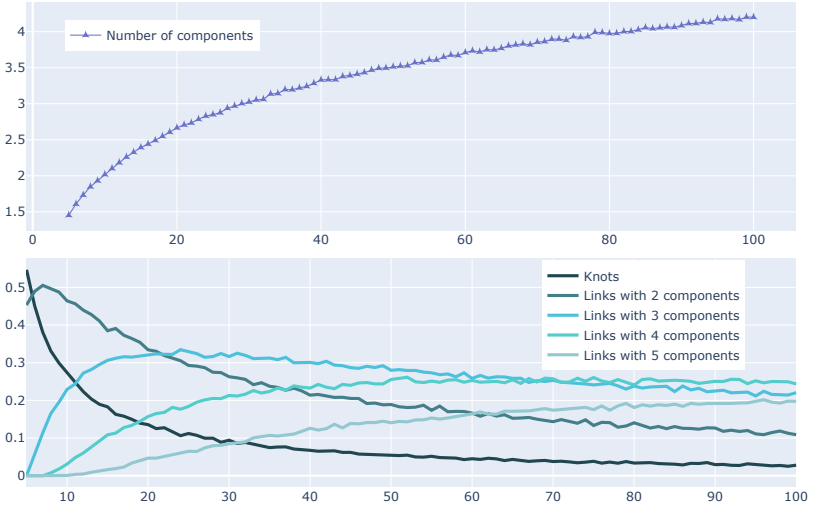

Figure 7. In the top figure, a plot of the average number of components in a sample population of random grid in the grid number range 5–100. The bottom figure shows the occurrence probabilities of grids with a given number of components as the grid number increases, for links with up to five components. Both given with respect to the grid number.

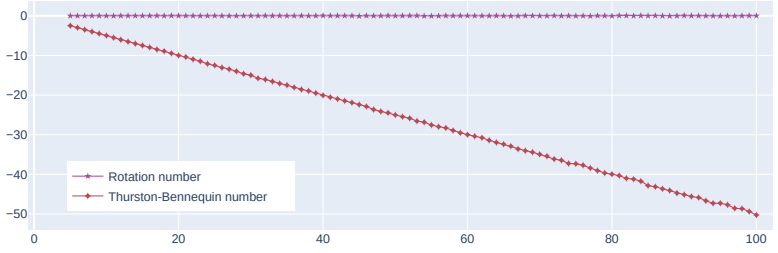

Figure 8. The plot of grid contact invariants (with respect to the grid number); consistently with their definition (see [14]), the average rotation number is constantly 0, while the Thurston–Bennequin number is on a line with slope $-\frac{1}{2}$.

Acknowledgements. Barbensi gratefully acknowledges funding through a MACSYS Centre Development initiative from the School of Mathematics and Statistics, the Faculty of Science and the Deputy Vice Chancellor of Research, University of Melbourne. Celoria was supported by Hodgson–Rubinstein's ARC grant DP190102363, "Classical and quantum invariants of low-dimensional manifolds".

SUPPLEMENT. The online supplement contains version 1.2 of GridPyM.

REFERENCES.

[1] K. L. Baker, J. E. Grigsby, and M. Hedden, "Grid diagrams for lens spaces and combinatorial knot Floer homology", *Int. Math. Res. Not.* **2008**:10 (2008), art. id. rnm024. MR Zbl

[2] J. A. Baldwin and W. D. Gillam, "Computations of Heegaard–Floer knot homology", *J. Knot Theory Ramifications* **21**:8 (2012), art. id. 1250075. MR Zbl

[3] A. Barbensi and D. Celoria, "GridPythonModule", available at https://github.com/agnesedaniele/GridPythonModule.

[4] A. Barbensi, D. Celoria, H. A. Harrington, A. Stasiak, and D. Buck, "Grid diagrams as tools to investigate knot spaces and topoisomerase-mediated simplification of DNA topology", *Sci. Adv.* **6**:9 (2020), art. id. eaay1458. Zbl

[5] A. Beliakova, "A simplification of combinatorial link Floer homology", *J. Knot Theory Ramifications* **19**:2 (2010), 125–144. MR Zbl

[6] Y. Burnier, C. Weber, A. Flammini, and A. Stasiak, "Local selection rules that can determine specific pathways of DNA unknotting by type II DNA topoisomerases", *Nucleic Acids Res. Spec. Publ.* **35**:15 (2007), 5223–5231. Zbl

[7] M. A. Cheston, K. McGregor, C. E. Soteros, and M. L. Szafron, "New evidence on the asymptotics of knotted lattice polygons via local strand-passage models", *J. Stat. Mech. Theory Exp.* **2014**:2 (2014), art. id. P02014. MR Zbl

[8] W. Chongchitmate and L. Ng, "An atlas of Legendrian knots", *Exp. Math.* **22**:1 (2013), 26–37. MR Zbl

[9] P. R. Cromwell, "Embedding knots and links in an open book, I: Basic properties", *Topology Appl.* **64**:1 (1995), 37–58. MR Zbl

[10] M. Culler, "Gridlink: a tool for knot theorists", available at www.math.uic.edu/∼culler/gridlink. Zbl

[11] P. Dabrowski-Tumanski, P. Rubach, W. Niemyska, B. A. Gren, and J. I. Sulkowska, "Topoly: Python package to analyze topology of polymers", *Brief. in Bioinform.* **22**:3 (2021), art. id. bbaa196.

[12] M. I. Doig, "Typical knots: size, link component count, and writhe", preprint, 2020. arXiv 2004.07730

[13] I. A. Dynnikov, "Arc-presentations of links: monotonic simplification", *Fund. Math.* **190** (2006), 29–76. MR Zbl

[14] H. Geiges, *An introduction to contact topology*, Cambridge Studies in Advanced Mathematics **109**, Cambridge Univ. Press, 2008. MR Zbl

[15] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and Jupyter Development Team, "Jupyter Notebooks: a publishing format for reproducible computational workflows", pp. 87–90 in *Positioning and power in academic publishing*: *players*, *agents and agendas*, edited by F. Loizides and B. Schmidt, IOS Press, Amsterdam, 2016.

[16] H. C. Lyon, "Torus knots in the complements of links and surfaces", *Michigan Math. J.* **27**:1 (1980), 39–46. MR Zbl

[17] C. Manolescu, P. Ozsváth, and S. Sarkar, "A combinatorial description of knot Floer homology", *Ann. of Math.* (2) **169**:2 (2009), 633–660. MR Zbl

[18] A. H. Moore and M. H. Vazquez, "Recent advances on the non-coherent band surgery model for site-specific recombination", pp. 101–125 in *Topology and geometry of biopolymers*, edited by E. Flapan and H. Wong, Contemp. Math. **746**, American Mathematical Society, Providence, RI, 2020. MR Zbl

[19] L. Ng and D. Thurston, "Grid diagrams, braids, and contact geometry", pp. 120–136 in *Proceedings of Gökova Geometry–Topology Conference* 2008, edited by T. O. elman Akbulut and R. J. Stern, International Press, Somerville, MA, 2009. MR Zbl

[20] P. Ozsváth and Z. Szabó, "Holomorphic disks and knot invariants", *Adv. Math.* **186**:1 (2004), 58–116. MR Zbl

[21] P. Ozsváth and Z. Szabó, "Bordered knot algebras with matchings", *Quantum Topol.* **10**:3 (2019), 481–592. MR Zbl

[22] P. Ozsváth and Z. Szabó, "Knot Floer homology calculator", available at https://tinyurl.com/knotFloerHom. Zbl

[23] P. S. Ozsváth, A. I. Stipsicz, and Z. Szabó, *Grid homology for knots and links*, Mathematical Surveys and Monographs **208**, American Mathematical Society, Providence, RI, 2015. MR Zbl

[24] J. A. Rasmussen, *Floer homology and knot complements*, Ph.D. thesis, Harvard University, 2003, available at https://www.proquest.com/docview/305332635. MR Zbl

[25] The Sage Developers, *SageMath, the Sage Mathematics Software System*, available at http://www.sagemath.org. Version 9.2. Zbl

[26] R. Stolz, M. Yoshida, R. Brasher, M. Flanner, K. Ishihara, D. J. Sherratt, K. Shimokawa, and M. Vazquez, "Pathways of DNA unlinking: a story of stepwise simplification", *Sci. Rep.* **7** (2017), art. id. 12420.

[27] S. L. Witte, *Link nomenclature, random grid diagrams, and Markov chain methods in knot theory*, Ph.D. thesis, University of California, Davis, 2020, available at https://www.proquest.com/docview/2436788600. MR Zbl

RECEIVED: 18 Oct 2022 REVISED: 20 Nov 2023 ACCEPTED: 8 Jan 2024

AGNESE BARBENSI:

agnese.barbensi@unimelb.edu.au
School of Mathematics and Statistics, University of Melbourne, Australia

DANIELE CELORIA:

dceloria.maths@gmail.com
School of Mathematics and Statistics, University of Melbourne, Australia