Eur. Phys. J. E (2018) **41**: 72
DOI 10.1140/epje/i2018-11681-0

THE EUROPEAN PHYSICAL JOURNAL E

Regular Article

KymoKnot: A web server and software package to identify and locate knots in trajectories of linear or circular polymers*

Luca Tubiana[1,a], Guido Polles[2], Enzo Orlandini[3], and Cristian Micheletti[4]

[1] Computational Physics Department, University of Vienna, Sensengasse 8/10, 1090, Vienna, Austria
[2] Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA
[3] Dipartimento di Fisica e Astronomia and Sezione INFN, Università di Padova, Via Marzolo 8, 35131 Padova, Italy
[4] SISSA, International School for Advanced Studies, Via Bonomea 265, I-34136 Trieste, Italy

Received 15 February 2018 and Received in final form 4 May 2018
Published online: 7 June 2018

Abstract. The KymoKnot software package and web server identifies and locates physical knots or proper knots in a series of polymer conformations. It is mainly intended as an analysis tool for trajectories of linear or circular polymers, but it can be used on single instances too, *e.g.* protein structures in PDB format. A key element of the software package is the so-called minimally interfering chain closure algorithm that is used to detect physical knots in open chains and to locate the knotted region in both open and closed chains. The web server offers a user-friendly graphical interface that identifies the knot type and highlights the knotted region on each frame of the trajectory, which the user can visualize interactively from various viewpoints. The dynamical evolution of the knotted region along the chain contour is presented as a kymograph. All data can be downloaded in text format. The KymoKnot package is licensed under the BSD 3-Clause licence. The server is publicly available at `http://kymoknot.sissa.it/kymoknot/interactive.php`.

1 Introduction

Knots emerge inevitably in equilibrated polymers that are sufficiently long [1–3] or densely packed [4–7]. Accordingly, much attention has been, and still is, paid to their occurrence in biopolymers, which can be long and compact, and where topological constraints can interfere with biological functionality. Prototypical systems are DNA plasmids [8], viral DNA [9–14], and proteins [15–23] where, unlike RNAs [24], the presence of knots has been long documented.

Like other forms of entanglement, knots can significantly affect physical aspects of polymers too, particularly their metric, mechanical, dynamical and rheological properties [25–27]. In recent years, this standpoint has motivated the systematic study of knotting in long polymeric filaments (typically DNA) subject to stretching [28–34], elongational flow [35], extensional fields [36, 37], spatial confinement of various dimensions and geometries [25,38–45] and molecular crowding [46].

Our understanding of both these accounts has been much shaped by theoretical approaches based on stochastic simulations, typically by Monte Carlo sampling or molecular dynamics. In these studies a key element is the systematic analysis of each conformation sampled in the trajectories to identify several topological properties: the presence of a knot; the type of knot and the location of the knot along the chain [25,38,47,48].

From a rigorous mathematical point of view, a proper topological state can be defined for closed chains only. For this reason, detecting and identifying knots in open chains (*physical knots*) is inherently ambiguous [49–52]; and so is identifying the shortest chain portion which embodies the knot in either a linear or circular chain. Notwithstanding these difficulties, the notion of physical knots can be established in a robust way by using suitable schemes to close open chains (or portions of circular chains). The one considered here, the algorithmic engine of the *KymoKnot* package and web server, is the minimally interfering scheme. The method, introduced by some of us and discussed in detail in ref. [53], bridges the termini of an arc in one of two possible ways: either directly or via the convex hull, depending on which is shorter. This comparative choice responds to criteria of simplicity of formulation, computational effectiveness and the necessity to minimize

* Contribution to the Topical Issue "Advances in Computational Methods for Soft Matter Systems" edited by Lorenzo Rovigatti, Flavio Romano, John Russo.

[a] e-mail: `luca.tubiana@univie.ac.at`

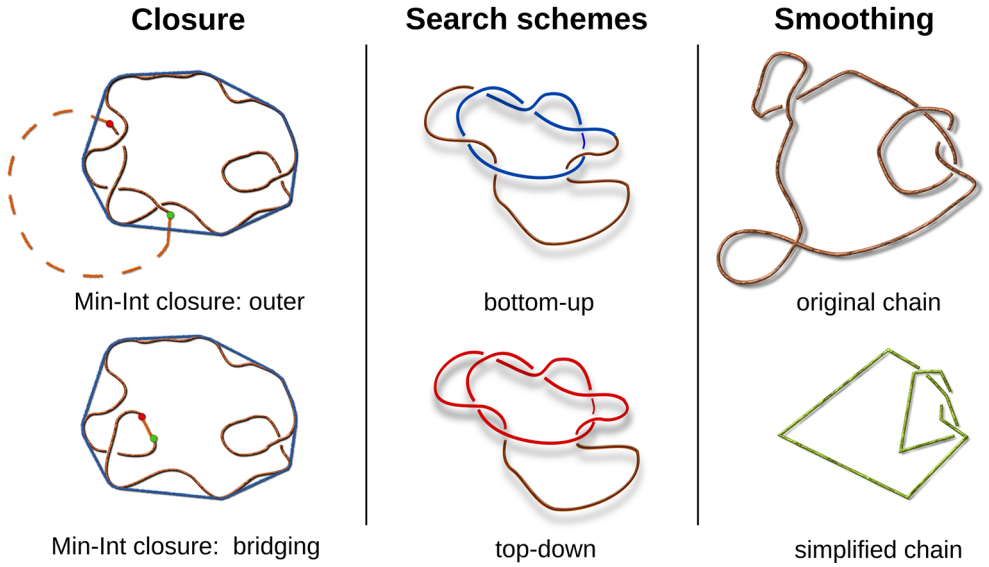

Fig. 1. Left: the minimally interfering closure scheme bridges the chain termini via the convex hull, when the termini are close to the hull, and bridges them directly when they are close to each other. Center: in the trefoil knotted configuration represented here, the bottom-up search identifies the portion highlighted in blue, while the top-down scheme the one highlighted in red. Right: results of the simplification scheme.

the topological "interference" of the auxiliary arc with the rest of the chain, *i.e.* adding the least amount of additional entanglement.

Other schemes, responding to different criteria, have been considered previously in the literature and implemented —some for general use, other for specific biopolymeric applications— in software packages and web servers such as KNOTS [54], pKnot2 [55], KnotProt [19], and one PyMol plugin, PyKnot [56].

The KymoKnot software presented here differs not only for adopting the minimally interfering closure scheme, whose computational effectiveness has been credited for making possible several extensive studies over the past years [27, 38, 48, 57–64], but also for the implementation aimed at analysing trajectories and ensembles of structures.

KymoKnot users will be able to perform such analysis by uploading their trajectories on the web server, where the results can be interactively visualized via a graphical interface or downloaded. For particularly extensive trajectories the analysis must be performed offline with the provided software package, downloadable from `https://github.com/luca-tubiana/KymoKnot`. The KymoKnot source code is released to the public under the BSD 3-Clause licence allowing users to modify it and share its extensions on github.

2 Methods

Algorithms that identify the knotted portion of a chain typically need three components. First, a closure scheme to circularize open portions of a chain. Second, the calculation of a topological invariant to establish the knotted state of the circularized portions. Third, a systematic search scheme to identify the knotted portion of the chain. In addition to these key elements, practical implementations of the search may involve a preliminary (topology-preserving) smoothing procedure where a subset of points of the discretized chain are eliminated to ease the computational cost.

KymoKnot uses the Alexander polynomial evaluated in $t = -1$ and $t = -2$ as topological invariants [65]. The values of these determinants are returned to the user along with the indication of the simplest knot type compatible with them. We recall, in fact, that the same Alexander polynomial can be shared by distinct knot types, and therefore the latter cannot be distinguished by solely using this invariant. Other user-defined topological invariants might be introduced in KymoKnot as outlined in sect. 3.1.

2.1 The minimally interfering closure scheme

KymoKnot adopts the so-called minimally interfering closure scheme [53] to establish the presence of physical knots in open chains. The term "physical" is conventionally used as a reminder that the topological state of an open chain is not mathematically well defined [49–52].

In brief, this scheme compares two distinct ways of bridging the ends of a chain portion: either via their direct connection or via the closest points of the convex hull of the chain portion, see fig. 1. The end-to-end distance is then compared with the summed distances of the termini from the convex hull. If the former is smaller, the direct bridging is used, otherwise the closure is through the convex hull. The criterion of the shortest length is used, heuristically, to choose the closure type that adds the least amount of entanglement to the considered region. Its effectiveness is discussed in ref. [53].

2.2 Search schemes for the knotted region

The knotted region is usually intended as the smallest portion of the chain that, after closure, has the same topology of the entire chain. Different search schemes can be used, and these typically return different knotted regions.

Figure 1 illustrates two such methods that are implemented in KymoKnot.

The first is a bottom-up search, where one starts from very short (hence unknotted) portions of the chain and gradually considers longer ones until two conditions are met: i) the considered portion has a physical knot of the same type as the whole chain and ii) the reminder of the (possibly circularised) chain is physically unknotted. The second condition could be omitted yielding a naive bottom-up search.

The second method is a top-down search, where one starts from the entire (possibly circularised) chain and considers progressively smaller portions until the original knotted state is lost.

2.3 Chain smoothing

The computational cost of systematic searches of the knotted region can become impractical for long and/or densely packed chains. The numerical burden can be eased by resorting to chain smoothing procedures, such as those introduced in refs. [2,15]. These procedures work by picking beads randomly and checking whether they can be made collinear with their flanking beads without introducing or removing chain crossings. Whenever this collinearity is possible, the selected bead is removed. Since the smoothing changes the geometry of the chain, it can modify the localization of the knot. To reduce this effect we introduce a *maximum simplification stride*, controlled by the user, which limits the number of subsequent beads which can be made collinear. The algorithm, whose effects are sketched in fig. 1, is described in detail in ref [53]. The topology-preserving smoothing procedure can significantly reduce the number of projected crossings and hence also the algorithmic complexity of establishing the topological state of the examined chain portion. The latter is always extracted from the original chain, *i.e.* the knot localization search is never performed starting from a smoothed version of the chain. When the chain to be analyzed is linear the rectification scheme is applied only after this is closed into a ring, thus preventing simplification from modifying the chain topology [52].

3 Results

3.1 Software package

This section describes the programs downloadable from `https://github.com/luca-tubiana/KymoKnot` which are the computational engine of the KymoKnot web server, described in sect. 3.2.

The two main programs are `KymoKnot_ring.x`, `KymoKnot_linear.x` and they identify the knotted region in circular and linear chains, respectively.

`KymoKnot_linear.x` is complemented by a third program, `K_close.x`, which takes as input one or more linear chains, closes them using the minimally interfering scheme and outputs the closed configuration.

The KymoKnot suite accepts input files written in the following XYZ format:

```
N
x_1 y_1 z_1
x_2 y_2 z_2
...
x_N y_N z_N
```

where N is the number of vertices/beads of the configuration. Several configurations, even with different values of N, can be appended to the same file. In this way a full trajectory can be analysed at once. Note that, if configurations represent ring polymers, the first and last beads in the corresponding XYZ entry must coincide, and the first line must be equal to $N + 1$. This format is used for the output of `K_close.x` as well.

Both `KymoKnot_ring.x` and `KymoKnot_linear.x` produce a set of files, one for each search scheme performed: *bottom-up*, *top-down*, and *naive bottom-up*, described in "Methods". The corresponding output files are named after the input file, respectively with the prefix: `BU__`, `TD__` and `NBU__`. The header of each file reports the seed used by the random number generator invoked by the stochastic smoothing, the simplification stride (see sect. 2.3) and the starting and ending points for the knot localizations. Each line after the header reports the following information: i) the configuration number, ii) the Alexander determinants in $t = -1$ and $t = -2$, iii) the knot type according to the Rolfsen table (described in `KNT_table.h`), iv) the beginning and ending points of the knot, and v) the knot length including the beginning and ending points. If the chain contains no knot, the last three entries are set to -1. A typical output file is reported below for an input file containing two configurations:

```
#idx  Adet_1 Adet_2 k_type start end len
0       3      7      3_1    83 104 21
1       3      7      3_1    74 95 21
```

To make the knot localisation programs more flexible a series of user-specified options are provided. They are: the smoothing stride, the region in which to look for the knotted portion, the seed for the pseudo random-number generator (prng), and the search scheme to be performed. By default, both programs simplify the chain with a stride equal to 2% of its contour length, set the prng seed from the system process id and time, and performs the bottom-up search only.

For reference, we tested how the run time needed to identify and locate knots scales with the number of edges, N, of knotted freely jointed rings (FJRs). FJRs can form many more and more complex knots than polymer models with the same number of bonds but endowed with excluded volume interactions or with a high bending rigidity [60]. For the purpose of the computational cost for

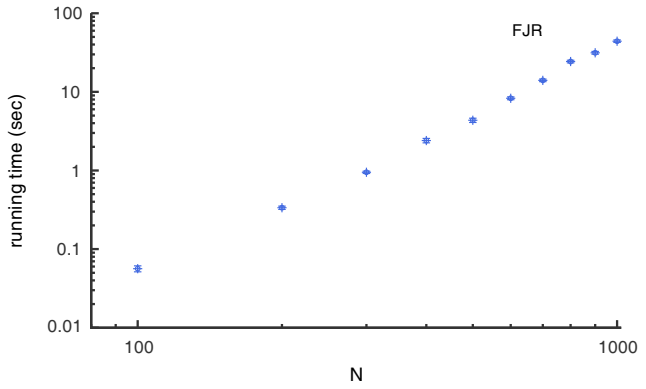

Fig. 2. KymoKnot running time evaluated on freely jointed rings (FJRs) of increasing contour length. Each point represents the average time needed to identify knots and, if present, locates them in a FJR of length N. The analysis was carried out on an Intel Xeon processor at 2.6 GHz. The average value and error bars have been evaluated on 10 separate batches composed of 100 independent configurations sampled at equilibrium.

determining and locating knots, they can theferore be regarded almost as a worst case scenario among common polymer models. With default parameters the average runtime needed by `KymoKnot_ring.x` to identify the topology of the ring and locate the knot, if present, on an Intel Xeon core at 2.6 GHz, was $\sim$ 60 milliseconds for $N = 100$ and $\sim$ 44 seconds for $N = 1000$. These times have been evaluated on batches of 100 uncorrelated conformations generated with a Monte Carlo procedure with crankshaft moves. These preserve bond lengths but not necessarily the ring topology and hence allow one to generate a topology-unresticted sample. The knotting probability of FJR goes from 30% for rings of $N = 100$ to 98% for $N = 1000$. The complete scaling curve of the performance, which is practically the same as `KymoKnot_linear.x`, is shown in fig. 2.

Cumulatively, the most demanding tasks in the pipeline of KymoKnot are: i) the topology-preserving geometrical simplification of the chain (which speeds up the evaluation of Alexander's polymial), and ii) the determination of the convex hull. Both these routines are repetitively referenced for locating the knotted portion. On a set of knotted freely jointed rings of $N = 500$ edges, these two tasks account, respectively, for 85% and 12% of the total running time. Both operations are performed for each chain portion to be analyzed, since we evaluate the topological state of an arc on the unsimplified chain (see sect. 2.3). Their execution times can be considerably reduced by increasing the stride used in the initial topology-preserving geometrical simplification, because this decreases in turn the number of chain portions to be analyzed [53].

The high modularity of KymoKnot allows also users to modify and extend its capabilities, as detailed in the package documentation. Here, as an example, we shortly describe how users can substitute the Alexander determinants with another topological invariant. Mathematical knots are mapped to a structure, `KNTid`, defined in `KNT_table.h`. This contains a unique ID, `(int) k_id`, for each knot type known to KymoKnot as well as the topological invariants needed to identify a mathematical knot. The same header contains a table associating topological invariants to knot IDs. The functions which determine the topological state of a ring are defined in `KNT_identify.h`. The rest of KymoKnot distinguishes different knots only through the value of `KNTid.k_id`. This design makes it possible to substitute the Alexander determinants with any other topological invariant by simply changing the tables in `KNT_table.h` and the functions in `KNT_identify.h`, provided that their interface is not modified and that the structure `KNTid` has a member called `k_id` associated to the knot type.

3.2 Web server

The KymoKnot suite can be used directly from its web server interface, available at
`http://kymoknot.sissa.it/kymoknot/interactive.php`.
The web server accepts both PDB and XYZ files (described in sect. 3.1) and runs remotely the knot search. It does not require installation or a specific platform and runs on recent releases of common browsers (Google Chrome 9+, Firefox 4+, Opera 15+, Safari 5.1+, Internet Explorer 11 and Microsoft Edge).

Data files can be uploaded from the dialog box that pops up either when the page is accessed, or by clicking on the *Load Structure File* link at top left corner the page. If the data file contains a series of structures, the server analyses the full trajectory and produces a kymograph with position and size of the knots as a function of time (see fig. 3). Because jobs are run remotely, we limit the size of the files to 80 MB and the maximum job execution time to one hour. When loading a PDB file, the interface retains only *CA* atoms and ignores the rest. The system separates the chains based on the PDB chain identifier, and shows a chain selector on the right of the visualization window. When loading a XYZ file, the system automatically detects whether the input file represent a trajectory and its number of frames.

The visualization area is based on the three.js WebGL library (`https://threejs.org/`), and displays the loaded structures as a tube. The tube centerline is computed as a Catmull-Rom interpolation of the input coordinates [66], and the tube radius is customarily set to 10% of the average separation between points.

The interface sets some default options which can be modified from a collapsible option panel. A *Run Options* button toggles the visualization of the panel. The default options are the following:

- Approach: the default approach is *bottom-up*.
- Polymer type: the structures are recognized either as rings or open chains. A ring is detected when the first and last coordinates are equal or very close (their difference is less than 10^{-4} for each Cartesian component). If the structure is not recognized as a ring, it is

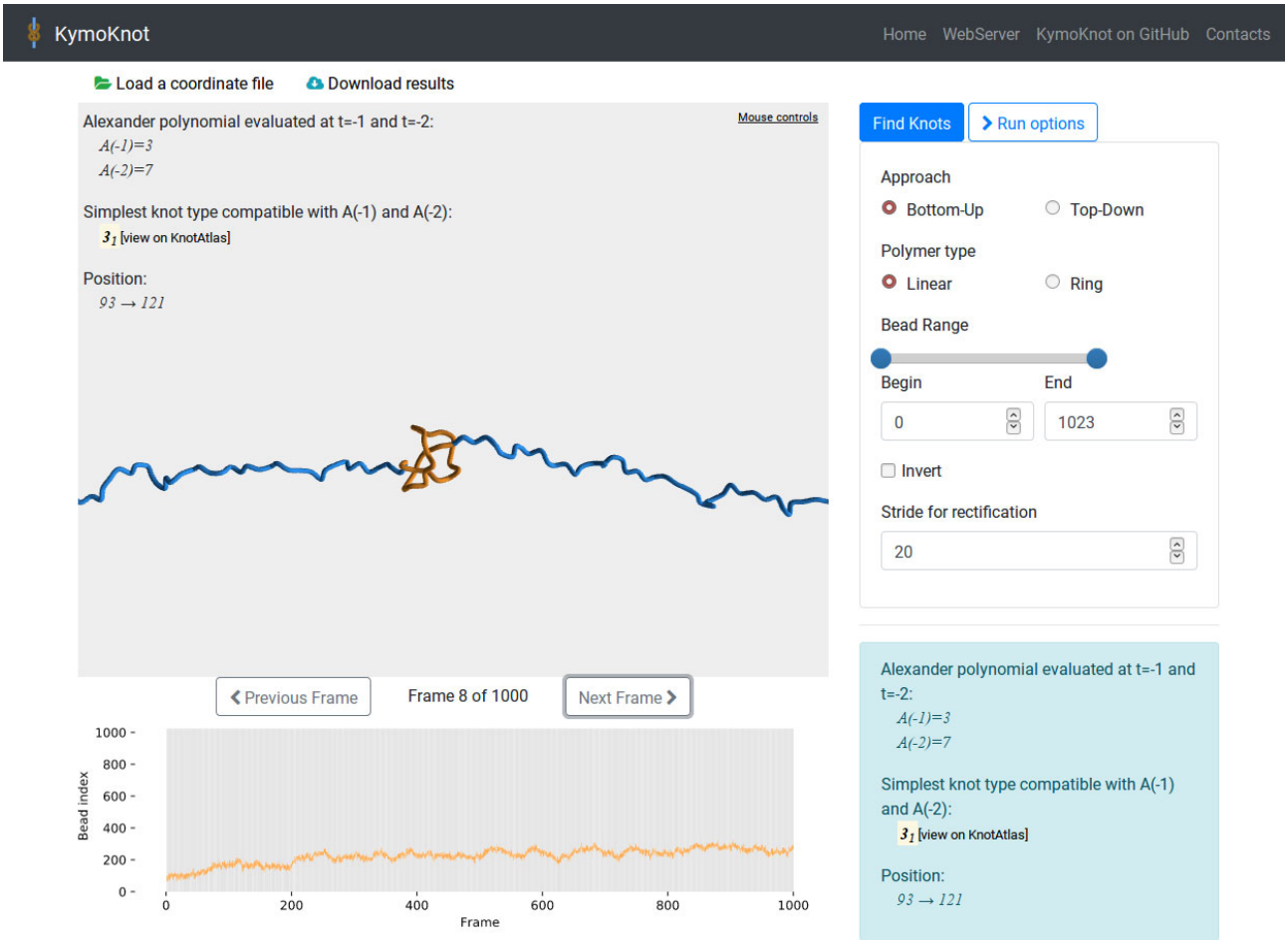

Fig. 3. A screenshot of the KymoKnot server, showing the interactive panel in the middle, with the knot highlighted; the interface for a search on the right-hand side, the kymograph for the knot position on the bottom left angle and the information on the knot on the bottom right corner. The position of the knot refers to the particular frame visualized in the interactive panel.

treated like an open chain by default, and the closure is based on the minimal-interfering algorithm. The default behavior can be overridden in the option panel by selecting "Ring" as polymer type, which connects the first and last bead with a straight segment.
- Bead range: the full chain is selected by default.
- Simplification stride: the default value is set to 2% the total length of the chain. This is a good compromise between speed and precision in most of the cases we tested, but the ideal value depends on the specific configuration and can be specified by the user.

The knot search can be started by clicking the *Find Knots* button. The interface collects the data, uploads it to the server for processing, and polls the server for results. The status of the job appears on the right side of the visualization area, and it is refreshed automatically.

When the search for the knotted region is completed, the web server displays a kymograph highlighting size and position of the knot along the chain and the detailed information on the knot type, along with a link to knot atlas: `www.katlas.org`. The knot is highlighted on the structure in the interactive panel on the center of the page; the panel allows the user to rotate, expand and translate structures using the mouse controls. The output files can be downloaded for offline analysis by clicking on the *Download results* button above the interactive panel.

4 Conclusions

KymoKnot is a software package that processes multiple configurations of polymers, such as Monte Carlo or molecular dynamics trajectories, to identify and locate knots. The analysis can be performed on both circular chains (with proper knots) and open ones (with physical knots). The KymoKnot source code is released to the public under the BSD 3-Clause licence, allowing users to modify it and share its extensions on github.

The web server implementation, available at `http://kymoknot.sissa.it/kymoknot/interactive.php` offers a user-friendly graphical interface for performing the same analysis without the need for package installation. The web server can take as input also individual conformations in PDB format, as a convenient way of analysing protein structures.

Open Access Funding provided by University of Vienna. LT acknowledges also support from the Mahlke-Oberman Stiftung and the European Union's Seventh Framework Programme for research, technological development and demonstration (Grant No. 609431).

Author contribution statement

LT, EO and CM designed the algorithms included in KymoKnot. LT developed the KymoKnot package. GP developed the KymoKnot web server. All the authors were involved in the preparation of the manuscript. All the authors have read and approved the final manuscript.

References

1. D.W. Sumners, S.G. Whittington, J. Phys. A: Math. Gen. **21**, 1689 (1988).
2. Kleanthes Koniaris, M. Muthukumar, Phys. Rev. Lett. **66**, 2211 (1991).
3. E. Orlandini, S.G. Whittington, Rev. Mod. Phys. **79**, 611 (2007).
4. S.G. Whittington, E.J. Janse van Rensburg, J. Phys. A: Math. Theor. **23**, 3573 (1990).
5. M.L. Mansfield, Macromolecules **27**, 5924 (1994).
6. Peter Virnau, Yacov Kantor, Mehran Kardar, J. Am. Chem. Soc. **127**, 15102 (2005).
7. M. Baiesi, E. Orlandini, A.L. Stella, F. Zonta, Phys. Rev. Lett. **106**, 258301 (2011).
8. José M. Sogo, Andrzej Stasiak, María Luisa Martínez-Robles, Dora B. Krimer, Pablo Hernández, Jorge B. Schvartzman, J. Mol. Biol. **286**, 637 (1999).
9. V.V. Rybenkov, N.R. Cozzarelli, A.V. Vologodskii, Proc. Natl. Acad. Sci. U.S.A. **90**, 5307 (1993).
10. J.C. Wang, S.Y. Shaw, Science **260**, 533 (1993).
11. Javier Arsuaga, Mariel Vazquez, Sonia Trigueros, De Witt Sumners, Joaquim Roca, Proc. Natl. Acad. Sci. U.S.A. **99**, 5373 (2002).
12. Javier Arsuaga, Mariel Vazquez, Paul McGuirk, Sonia Trigueros, De Witt Sumners, Joaquim Roca, Proc. Natl. Acad. Sci. U.S.A. **102**, 9165 (2005).
13. Davide Marenduzzo, Enzo Orlandini, Andrzej Stasiak, De Witt Sumners, Luca Tubiana, Cristian Micheletti, Proc. Natl. Acad. Sci. U.S.A. **106**, 22269 (2009).
14. Davide Marenduzzo, Cristian Micheletti, Enzo Orlandini *et al.*, Proc. Natl. Acad. Sci. U.S.A. **110**, 20081 (2013).
15. W.R. Taylor, Nature **406**, 916 (2000).
16. Peter Virnau, Leonid A. Mirny, Mehran Kardar, PLoS Comput. Biol. **2**, e122 (2006).
17. Anna L. Mallam, Joseph M. Rogers, Sophie E. Jackson, Proc. Natl. Acad. Sci. U.S.A. **107**, 8189 (2010).
18. R. Potestio, C. Micheletti, H. Orland, PLoS Comput. Biol. **6**, e1000864 (2010).
19. Michal Jamroz, Wanda Niemyska, Eric J. Rawdon, Andrzej Stasiak, Kenneth C. Millett, Piotr Sułkowski, Joanna I. Sulkowska, Nucl. Acids Res. **43**, D306 (2014).
20. Nicole C.H. Lim, Sophie E. Jackson, J. Phys.: Condens. Matter **27**, 354101 (2015).
21. Sophie E. Jackson, Antonio Suma, Cristian Micheletti, Curr. Opin. Struct. Biol. **42**, 6 (2017).
22. Pawel Dabrowski-Tumanski, Joanna I. Sulkowska, Polymers **9**, 454 (2017).
23. Patrícia F.N. Faísca, Comput. Struct. Biotechnol. J. **13**, 459 (2015).
24. Cristian Micheletti, Marco Di Stefano, Henri Orland, Proc. Natl. Acad. Sci. U.S.A. **112**, 2052 (2015).
25. Cristian Micheletti, Davide Marenduzzo, Enzo Orlandini, Phys. Rep. **504**, 1 (2011).
26. D. Meluzzi, D.E. Smith, G. Arya, Annu. Rev. Biophys. **39**, 349 (2010).
27. Danielle J. Mai, Charles M. Schroeder, Curr. Opin. Colloid Interface Sci. **26**, 28 (2016).
28. O. Farago, Y. Kantor, M. Kardar, Europhys. Lett. **60**, 53 (2002).
29. X.R. Bao, H.J. Lee, S.R. Quake, Phys. Rev. Lett. **91**, 265506 (2003).
30. Lei Huang, Dmitrii E. Makarov, J. Phys. Chem. A **111**, 10338 (2007).
31. R. Matthews, A.A. Louis, J.M. Yeomans, EPL **89**, 20001 (2010).
32. Peter Poier, Christos N. Likos, Richard Matthews, Macromolecules **47**, 3394 (2014).
33. Michele Caraglio, Cristian Micheletti, Enzo Orlandini, Phys. Rev. Lett. **115**, 188301 (2015).
34. S. Najafi, L. Tubiana, R. Podgornik, R. Potestio, EPL **114**, 50007 (2016).
35. C. Benjamin Renner, Patrick S. Doyle, Soft Matter **11**, 3105 (2015).
36. Vivek Narsimhan, Alexander R. Klotz, Patrick S. Doyle, ACS Macro Lett. **6**, 1285 (2017).
37. Alexander R. Klotz, Vivek Narsimhan, Beatrice W. Soh, Patrick S. Doyle, Macromolecules **50**, 4074 (2017).
38. Luca Tubiana, Enzo Orlandini, Cristian Micheletti, Phys. Rev. Lett. **107**, 188302 (2011).
39. Jing Tang, Ning Du, Patrick S. Doyle, Proc. Natl. Acad. Sci. U.S.A. **108**, 16153 (2011).
40. A. Rosa, M. Di Ventra, C. Micheletti, Phys. Rev. Lett. **109**, 118301 (2012).
41. Cristian Micheletti, Enzo Orlandini, Macromolecules **45**, 2113 (2012).
42. Cristian Micheletti, Enzo Orlandini, Soft Matter **8**, 10959 (2012).
43. Calin Plesa, Daniel Verschueren, Sergii Pud, Jaco van der Torre, Justus W. Ruitenberg, Menno J. Witteveen, Magnus P. Jonsson, Alexander Y. Grosberg, Yitzhak Rabin, Cees Dekker, Nat. Nanotechnol. **11**, 1093 (2016).
44. Antonio Suma, Cristian Micheletti, Proc. Natl. Acad. Sci. U.S.A. **114**, E2991 (2017).
45. Liang Dai, Johan R.C. van der Maarel, Patrick S. Doyle, ACS Macro Lett. **1**, 732 (2012).
46. Giuseppe D'Adamo, Cristian Micheletti, Macromolecules **48**, 6337 (2015).
47. B. Marcone, E. Orlandini, A.L. Stella, F. Zonta, Phys. Rev. E **75**, 041105 (2007).
48. Liang Dai, C. Benjamin Renner, Patrick S. Doyle, Macromolecules **48**, 2812 (2015).
49. S.G. Whittington, D.W. Sumners, J. Phys. A: Math. Gen. **23**, 1471 (1990).
50. Marc L. Mansfield, Nat. Struct. Mol. Biol. **1**, 213 (1994).
51. M.L. Mansfield, Macromolecules **31**, 4030 (1998).

52. A. Stasiak K. Millett, A. Dobay, Macromolecules **38**, 601 (2005).
53. Luca Tubiana, Enzo Orlandini, Cristian Micheletti, Prog. Theor. Phys. Suppl. **191**, 192 (2011).
54. Grigory Kolesov, Peter Virnau, Mehran Kardar, Leonid A. Mirny, Nucl. Acids Res. **35**, W425 (2007) (Suppl. 2).
55. Yan-Long Lai, Chih-Chieh Chen, Jenn-Kang Hwang, Nucl. Acids Res. **40**, W228 (2012).
56. Rhonald C. Lua, Bioinformatics **28**, 2069 (2012).
57. Cristian Micheletti, Enzo Orlandini, ACS Macro Lett. **3**, 876 (2014).
58. Ivan Coluzza, Peter D.J. van Oostrum, Barbara Capone, Erik Reimhult, Christoph Dellago, Phys. Rev. Lett. **110**, 075501 (2013).
59. L. Tubiana, A. Rosa, F. Fragiacomo, C. Micheletti, Macromolecules **46**, 3669 (2013).
60. Luca Tubiana, Phys. Rev. E **89**, 052602 (2014).
61. Liang Dai, C. Benjamin Renner, Patrick S. Doyle, Phys. Rev. Lett. **114**, 037801 (2015).
62. Miguel A. Soler, Antonio Rey, Patrícia F.N. Faísca, Phys. Chem. Chem. Phys. **18**, 26391 (2016).
63. Raffaello Potestio, Luca Tubiana, Soft Matter **12**, 669 (2016).
64. Saeed Najafi, Rudolf Podgornik, Raffaello Potestio, Luca Tubiana, Polymers **8**, 347 (2016).
65. E. Orlandini, S.G. Whittington, Rev. Mod. Phys. **79**, 611 (2007).
66. Edwin Catmull, Raphael Rom, *A class of local interpolating splines*, in *Computer Aided Geometric Design*, edited by Robert E. Barnill, Richard F. Riesenfeld (Academic Press, 1974) pp. 317–326.