UNDER REVIEW

RESCIENCE C

Replication

[Rp] Reproducibility report: Estimating friction coefficients of mixed globular/chain molecules, such as protein/DNA complexes. [Biophys J 69, 840-848 (1995)]

Charles H. Robert[1,2]

[1]CNRS, Université de Paris, Laboratoire de Biochimie Théorique, UPR9080, 13, rue Pierre et Marie Curie, F-75005 Paris, France – [2]Institut de Biologie Physico-Chimique, Fondation Edmond de Rothschild,, PSL Research University, Paris, France

Edited by
(Editor)

Received
–

Published
–

DOI
–

1 Introduction

The chosen article from 1995[1] described a new method for calculating friction coefficients of large, flexible DNA-protein complexes, with the goal of helping infer their structural properties from light scattering or ultracentrifugation data. The present report describes my successful effort to reproduce the results presented in the article, and offers some thoughts on the easier and more difficult parts of the task.
About two meters of DNA is packaged into every human cell nucleus, and a major focus of cell biology is on how eukaryotic cells in general reconcile this degree of compaction with the need to access the genetic message of the DNA. The packaged form of DNA is called chromatin, and it is composed in essence of regions of relatively free DNA interspersed by other regions in which about two turns of the DNA double helix are wrapped neatly around proteins called histones; these globular regions are called nucleosomes. The degree of chromatin compaction is locally dynamic; such remodelling reflects different biological imperatives in the cell cycle and adaptation to different metabolic contexts[2].
The approach taken in the article to modelling biophysical hydrodynamics data for these large molecular systems consisted of decomposing the molecule into globular and chain regions (nucleosomes and free DNA, respectively), then replacing each chain region by a single, calculated hydrodynamic sphere; the average frictional properties of the entire assembly were then established using Kirkwood-Bloomfield modelling[3], together with appropriate chain conformational statistics (see overview shown in Figure 1 in the article). The approach was validated on test geometries and applied to realistic chromatin structural models obtained using a novel mathematical 3D model of the 30 nm chromatin fiber, defined as a function of structural parameters including the number of superhelical turns of each nucleosome and the helical twist of the DNA double helix. General features of this modelling were presented briefly in the appendix.

2 Targets

The targets for reproduction were the following.

- Figure 2. This figure showed a test of the method presented in the article for predicting translational friction coefficients of a series of reference assemblies, whose exact properties had been reported earlier. These assemblies had two shapes (lollipop and dumbbell), and their dimensions were chosen to approximate the dimensions of a nucleosome along with free "linker" DNA.
- Figure 3 compared the translational force calculated using the Kirkwood-Bloomfield bead approximation coupled with the new method, versus the exact force calculated by the Zimm approach[4], as a function of the free DNA length in basepairs (bp).
- Figure 4 showed the translational diffusion coefficient for a system consisting of two nucleosomes positioned on a 354 bp DNA template, calculated using the new method and plotted as a function of a putative change in the number of superhelical turns. The calculations were compared to experimental results from light-scattering experiments on a similar system obtained at two different salt concentrations.
- Figure 5/Table 1. A demonstration of the potential of the new calculation method for evaluating the relevance of different models for DNA compaction. Sedimentation coefficients were calculated for all possible intermediates in a hypothetical saturation of a 624 bp DNA template with up to three positioned nucleosomes, and the distributions obtained for histone binding in a cooperative versus a non-cooperative (random) process.

3 Methods

The calculations reported in the 1995 article were all performed with Mathematica version 2.2 on a 1992 Macintosh IIci (Motorola 68030 clocked at 25 MHz) running macOS 7.1. All Mathematica notebooks for the article remained accessible throughout the intervening years in my project file directories.
The reproducibility tests were performed using two different versions of Mathematica on different platforms:

- Mathematica version 5.2, released in 2005, was first used, running on a late-2018 13-inch Macbook (Intel core i7, 2.7 GHz) running macOS 10.14 (Mojave). Note that Mathematica 5.2 will no longer run on macOS $\geq$ 10.15 (Catalina) because it contains 32-bit code.
- All code was also run using Mathematica version 12.0, released in 2019, running under Raspbian linux on a Raspberry Pi 3 model B+ and 4 model B.

The numerical results presented in the article had been preserved in the original notebooks. Although not targets of this comparison *per se*, the article figures (as opposed to numerical notebook data) had been produced either from the original Mathematica Postscript followed by some figure compositing, or by exporting numerical data for input to Cricket Graph (if memory serves– I can't read those figure files now.)
All original and modified software described here has been made publicly available as a directory "rehydro". In preparing for the reproduction trials, the original notebooks were automatically converted using Mathematica v5.2 to an ascii format. The 1995 notebooks generally contained additional code and results unrelated to the article or to the present tests; all relevant code for each target was saved with the root filename addition "_1995" in the subdirectory "originals". Copies of these files, with added filename code "_repro", were then used for all reproduction attempts and modified as necessary in order to obtain results for comparison. The original library notebook "s6.1_1995.nb",

modified as necessary to "s6.1_repro.nb", plus a notebook "extra.nb" containing additional plotting functions, were stored as Mathematica 5.2 packages with the suffix ".m", and the "Get" command added to the top of the relevant notebooks to execute them.
Running notes: In order to run the "_repro" files, a symbolic link "ReScience" pointing to the rehydro code directory was first placed in Mathematica's default working directory (my home directory in this case). Also, for each target, in order to remove history-dependent effects, the kernel was restarted and all target notebook cells were evaluated (answering "No" to the dialog proposing to evaluate initialization cells only).

4 Results

1. Figure 2: The article's approach was validated by using it to calculate translational frictional coefficients for two series of reference assemblies (points connected by solid lines in the Figure). These were compared to exact values (black circles in the Figure) and to values calculated using an existing approximation method (dotted lines).

   The original notebook contained all the necessary code to calculate the friction coefficients of the reference assemblies as well as the numerical values plotted in Figure 2. The friction coefficients calculated in this notebook were thus the targets for the reproduction.

   The notebook was executed without any errors or warnings. The calculated friction coefficients are identical to the values obtained in the original notebook, differing only in the default rounding (4 figures in the original notebook, 6 in the new notebook).

2. Figure 3: Total force calculated using the Kirkwood-Bloomfield (KB) approximation, used here with the sphere decomposition proposed in the new method, compared to the exact value.

   The notebook was documented and it was fairly easy to find the relevant code. However, execution at first produced no numerical results, and I remembered that a notebook "s6.1" containing library functions was stored alongside the others in the local directory. After adding the necessary code to load the definitions (see Methods), running the notebook then provided values that were identical with those plotted in the article. For convenience, I added a plot of the reproduced values at the end of the "_repro" notebook to visually compare to the original Figure.

3. Figure 4: Calculation of translational friction coefficients and corresponding diffusion coefficients for different conformations of a dinucleosome as a function of the degree of DNA wrapping.

   Again I had to guess that the notebook depended on my library functions. Also, on execution, Mathematica generated a runtime error: "Set::wrsym: Symbol MonteCarlo is Protected." This resulted from my use of a Boolean variable called "MonteCarlo", which collided with a Mathematica option name that was defined in version 5.2; I renamed the variable to all lowercase.

   Although the notebook then executed without error, the calculated values reproduced those in the 1995 notebook only to within 4 significant figures, despite the fact that both the new and the 1995 output provided 6 figures. Digging further revealed that after the creation of this figure I had applied a small correction to the average chain statistics in the "doKirkwood" function in the library s6.1, but which I apparently judged too insignificant to justify re-running the calculations at the time. In support of this hypothesis, I found the earlier version of doKirkwood in my previous version of the library, which I added to s6.1_repro.nb as "doKirkwood0". Calling this function produced identical results to those obtained in the 1995 notebook and plotted in Figure 4 of the article (see **Table 1**).

The small difference between the two columns can be seen to decrease even further as more DNA is wrapped onto the nucleosome and the length of the linker DNA is correspondingly reduced.

Table 1. Dinucleosome sedimentation coefficients in Svedberg units, calculated with and without average distance correction

Superhelical turns	as published	with later correction
1.5	1.95652	1.96035
1.6	2.08147	2.08471
1.7	2.24511	2.2478
1.8	2.49574	2.4978
1.9	2.79165	2.79306
2.0	3.02864	3.02924

4. Figure 5 / Table: Sedimentation coefficients calculated for intermediate species in the saturation of a three-site DNA template during its progressive compaction by nucleosome formation.

   This code again depended on functions defined in the library s6.1 as well as a few other generic functions for plotting, which I identified using grep and metafiledata (here, the file date) to identify the relevant older versions of my general library functions.

   In the article, the value for the free DNA template was calculated using an existing method in order to plot the complete sedimentation coefficient distributions. However, for some reason only the function which returned the hydrodynamic radius remained. I didn't pursue this particular omission further; in any case the method for calculating the friction properties of the free DNA species alone preexisted the approach presented in the article.

   Scrutinizing my original notebooks revealed another problematic aspect of notebook computing. The function doKirkwood coordinates the calculation of the frictional properties of a given macromolecular species and reports them, along with other results obtained by calling additional functions. In the 1995 output, one such derived value was the sedimentation coefficient (s1, the 3rd value in brackets printed for each intermediate species). However, it was not calculated by doKirkwood; only the symbol "s1" was output, and no numerical value. A likely explanation is that before running the loop I manually executed the cell defining doKirkwood in my library notebook, but not the cell defining the "sediment" function (I did not yet know the "Get" command to load a library package). This would explain why the sedimentation coefficients were calculated from the friction coefficients separately, after manually executing the cell defining the "sediment" function. Of course, no concrete trace of this execution order remains, and this example serves as an example of the pitfalls of "hidden state" in notebook computing.

   Regardless, all friction and sedimentation coefficients calculated for the DNA-protein complexes, together with the bar charts representing the sedimentation coefficient distributions for two possible saturation models, were reproduced exactly as in the Table and Figure 5 of the 1995 article.

5 Conclusions

I felt fairly confident that I would be able to reproduce the results included in this article on a modern machine, and this was the case. However, it took a bit longer than I had anticipated.
The mid-nineties notebook code, written in Mathematica version 2.2, ran with very few changes on more recent Mathematica versions (5.2 from 2005 and 12.0 from 2019). The observed incompatibilities were

- Name collision: a variable name in my code was defined as a reserved word in more recent versions of Mathematica
- Change in action of trailing semicolon in later versions of Mathematica (previously it suppressed text output; it now suppresses output of graphical results as well)
- Plotting style option names were changed in more recent versions
- Change in default AspectRatio for some graphics

Only the first involved the calculations directly, and all issues were minor and could be expected in any evolving language.
One might note that a particular consequence of Mathematica's symbolic processing capabilities is that it will not automatically throw an error if a non-numerical result results from a calculation. This is the case if a function has simply not been defined, as was the case for the last example discussed above. Mathematica will happily provide output consisting of the symbolic function name mixed with numerical values for other symbols. Although this behavior is obviously critical for symbolic manipulations, it makes it more difficult to identify omitted function definitions.
A more significant barrier came from missing documentation for the dependencies of the notebook code on functions and definitions made in other notebooks. Mathematica pioneered the use of notebooks on the Macintosh in the 1980's, but it's only been in more recent years that notebook interfaces such as Jupyter have received widespread adoption, (*e.g.*, Rule *et al.*[5]). When using notebooks for significant tasks, the problem of dependencies, or more generally, state, can be particularly dogging, as notebooks implicitly encourage state to be established interactively, as opposed to programmatically. Despite being an early proponent, I could already see the madness notebooks could engender, and as a consequence even in the 1990s I made some efforts to organize my code and to perform rudimentary versioning. Twenty-five years later this allowed me to reconstruct the dependencies used for this article without too much effort. I only wish I had gone further and taken the time to employ resources such as packages already offered in Mathematica to do so.
A vexing aspect of my 1990's code had nothing to do with Mathematica or notebooks in general: the liberal use of global variables. The notebooks were consistent, but coordinating several notebooks required chasing down each variable in different contexts. This unfortunately dampens one's enthusiasm for re-using the code for other applications.
Nevertheless, the calculations presented in the 1995 article could be reproduced, with some effort, in three different 2020 computing environments from the original code. One reason for this is the self-consistency and continuity in the development of Mathematica, which began seven years before the target article's publication and continues to this day. A second type of continuity that might be mentioned is that implied in developing this subject over several years, alternating with other topics, which obliged me to document and modularize my own code (although the result was decidedly far from perfect!). A short-term, one-off project would perhaps have produced a very different result if I had to reproduce it 25 years later. Indeed, while it is clearly in no way optimal, seeing this code running in a modern environment encourages me to continue its

development in light of mounting advances both in quantifying chromatin remodelling and in appreciating its biological importance.

6 Acknowledgements

Thanks to Wolfram Research for making Mathematica available on the Raspberry Pi platform, which allowed me to test using version 12.0, to the French CNRS program 80|PRIME-2019 (Projet "DECRYPTOR"), and more generally to the "Initiative d'Excellence" program from the French State (Grants "DYNAMO", ANR-11-LABX-0011, and "CACSICE", ANR-11-EQPX-0008). Special thanks to Luc Boulesteix, who kindly lent me his Raspberry Pi as one of the test environments.

References

1. C. H. Robert. "Estimating friction coefficients of mixed globular/chain molecules, such as protein/DNA complexes." In: **Biophys J** 69.3 (Sept. 1995), pp. 840–848.
2. Y. Tian, G. Garcia, Q. Bian, K. K. Steffen, L. Joe, S. Wolff, B. J. Meyer, and A. Dillin. "Mitochondrial Stress Induces Chromatin Reorganization to Promote Longevity and UPR(mt)." In: **Cell** 165.5 (May 2016), pp. 1197–1208.
3. J. Garcia de la Torre and V. A. Bloomfield. "Hydrodynamic properties of macromolecular complexes. 1. Translation." In: **Biopolymers** 16 (), pp. 1747–1763.
4. B. H. Zimm. "Chain Molecule Hydrodynamics by the Monte-Carlo Method and the Validity of the Kirkwood-Riseman Approximation." In: **Macromolecules** 13.3 (1980), pp. 592–602. eprint: https://doi.org/10.1021/ma60075a022.
5. A. Rule et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." In: **PLoS Comput Biol** 15.7 (July 2019), e1007007.