UNDER REVIEW

RESCIENCE C

Reproduction / Biophysics

# [Rp] Structural flexibility in proteins: impact of the crystal environment

**Konrad Hinsen**[1,2]

[1]Centre de Biophysique Moléculaire, CNRS UPR4301, Orléans, France – [2]Synchrotron SOLEIL, Division Expériences, Gif sur Yvette, France

**Edited by**
(Editor)

**Received**
–

**Published**
–

**DOI**
–

## Introduction

The article that is the subject of this reproduction attempt was my first publication for which I provided code as supplementary material [1]. In computational biophysics, that was uncommon back in 2008, and unfortunately it remains the exception until today. The code consisted of two Python scripts plus instructions and was meant to help readers with applying the methods discussed in the article to different proteins, rather than ensure reproducibility which at that time I had never heard of. This explains why the supplied code does not reproduce the figures shown in the paper, but merely produces data files in plain text format, from which the plots were originally assembled by hand and by additional scripts that I did not consider worthy of publication.

## Historical context

The work described in the original article was performed in summer 2007. The scripts make heavy use of two libraries of which I am the principal author: the Molecular Modeling Toolkit (MMTK) [2], then at version 2.5, and ScientificPython, then at version 2.7. These two libraries, first published in 1997, are among the oldest scientific computing packages in the SciPy ecosystem. They were initially written on the basis of Numerical Python [3], the original array package for Python that was published in 1995. With the transition of the ecosystem to its successor NumPy [4], initially released in 2006, I ported MMTK and ScientificPython using NumPy's compatibility module called `oldnumeric`. This combination was used for the original work, but unfortunately I did not write down the exact version of NumPy. My instructions recommend "Numerical Python 23.8.2 or NumPy 1.x", which turned out to be overly optimistic: NumPy is still in the 1.x version range, but frequent breaking changes make it impossible to run my code with recent NumPy releases. In fact, with version 1.9 NumPy removed the `oldnumeric` interface, breaking all of my molecular simulation code. I had envisaged to port the code to the official NumPy API, but decided not to do so because of the high risk of introducing errors. There are in fact several subtle changes in the API, such as using the transpose of a matrix compared to the original Numeric API, which are easy to overlook because incorrect use yields an incorrect result but no error message. With the end of support for Python 2, there is longer any point in such a change, as it makes little difference if my code depends on one or two unmaintained software packages.


Correspondence should be addressed to Konrad Hinsen (konrad.hinsen@cnrs.fr)
The authors have declared that no competing interests exists.
Code is available at https://github.com/khinsen/rescience-ten-year-challenge-paper-3..

The remaining dependencies turned out to be less critical. netCDF, which my scripts use for data storage, was at version 3.4, but any more recent version can be substituted. Python itself, then at version 2.5, caused no problems either up to release 2.7. Python 3 is from my point of view a different language, the two major changes for scientific computing being a very different C API layer and a different definition of division. Porting would involve a significant amount of work because of the large number of extension modules in MMTK, most of which are C code hand-written for the C API of Python 1.4, long before Pyrex [5] and then its fork Cython [6] introduced more convenient ways to write extension modules.

# Reproduction

The GitHub repository for this reproduction contains all the code, input data, and instructions for running the reproduction on a modern GNU/Linux system with the Guix package manager [7, 8]. It also lists the exact version numbers of all the software involved in the reproduction. For the numerical calculations alone, i.e. excluding the less critical tasks of downloading files and generating plots, a total of 254 Guix package must be rebuilt identically to guarantee bit-for-bit reproduction of the results.

## Finding the code and the input data

The project-specific scripts are still available for download from the journal's Web site. All the dependencies have been available in public version-controlled repositories for many years. Obtaining the published code is therefore not a problem. The additional scripts that generated the original plots were not published as explained in the introduction. They have not survived two changes of computers. I still have backups that should contain them, but I have not been able to find a working reader for the DAT/DDS tapes on which these backups are stored. Another backup on a CD-ROM turned out to be unreadable.

The input data for reproducing the figures in the original paper consists of three protein structures from the Protein Data Bank (PDB) [9]. The PDB updates its files from time to time. The file for a specifc entry is intended to represent the original data deposited by its authors, but may be modified to conform to newer versions of the file format, or to fix technical mistakes. There is thus no guarantee that a file downloaded today is the same as in 2008, but the scientific information it contains is supposed to stay the same. In practice, PDB updates have occasionally broken analyses of the annotations they contain, but not analyses of the original experimental data.

## Running the scripts using the Guix package manager

My preferred software management tool for reproducible computations is the Guix package manager [7], which permits the bit-for-bit reconstruction of software environments at any later time. All the required dependencies have already been packaged in Guix, but since Guix only started in 2012, and the Python ecosystem was added even later, the original software versions of 2008 are not available.

The two scripts are meant to be run in sequence, once for each protein structure and crystal size. The first script, `calculate_crystal_fluctuations.py`, computes the normal modes of the crystal and stores them in a netCDF file. The second script, `analyze_crystal_fluctuations.py` extracts the relevant data for the plots from the netCDF file and writes them to text files.

Using the dependencies as defined in Guix in January 2020 (more precisely: commit `7357b3d7a52eb5db1674012c50d308d792741c48`), the first script runs without any apparent problem, but the second one crashes with an error message. This is the consequence of a breaking change in NumPy which modified the rules for the conversion

**Figure 1.** Blank figure included for aligning the figure numbers in this reproduction with the figure numbers in the original paper.

of sequence-like Python objects into arrays. This problem can be fixed by changing a single line of code; however, neither the diagnosis nor the correction are likely to be obvious to someone who is not intimately familiar with NumPy.

I explored the possibility of using an earlier NumPy release in order to run the scripts unmodified. From the release history that is available on NumPy's GitHub repository, it appears that release 1.0.4 was current at the time of submission of my paper in late 2007. Unfortunately, this NumPy release cannot be installed with Python 2.7 because NumPy is distributed with a modified version of the `distutils` package. `distutils` reads a configuration file produced during the installation of Python, whose format has changed between Python 2.5 and 2.7. I briefly envisaged installing Python 2.5, but that would have required backporting the modifications made for Guix, which seemed an unreasonable effort for performing a simple test.

In addition to the fix described above, I modified my scripts for convenience, making them read their originally hard-coded input parameters from command line arguments instead. In fact, the scripts I had used for the original work also accepted command line arguments, but I had hard-coded them in the published versions in order to simplify the usage instructions.

## Reproducing the figures

As explained in the introduction, my goal with publishing the code of my computations was to enable reuse, not reproducibility. The code therefore does not reproduce the full figures, which need to be re-generated by hand from the numerical output. I have limited myself to producing figures that are similar enough to the originals to convince the reader that the results have been reproduced correctly. I also used different plotting software, Gnuplot [10], in replacement of the originally used Grace [11] that seems to be no longer supported.

Figure 1 of the original publication does not contain any computed data and has therefore not been reproduced. The data it shows comes directly from the Protein Data Bank.

Figures 2 and 3 of the original publication compare "single molecule" to "crystal" results for two proteins. Only the latter are computed by the scripts in the supplementary material, and are shown in Figs. 2 and 3. The "single molecule" curves were obtained by a script that was not published, and which I have lost. It would not be difficult to replicate, but that is not the goal of this reproduction attempt.

Figure 4 of the original publication shows the dispersion relations for four different directions of wave propagation in the crystal. Unfortunately, the supplied scripts only produce a single file with points on the dispersion plots from all directions combined. The additional script required to separate these points by direction was not published and has been lost. The dashed lines labelled "elastic medium" were also computed by unpublished and lost scripts. Fig. 4 shows unconnected points that each lie on one of the drawn-out lines of the original Figure 4.

Figure 5 of the original publication is reproduced almost entirely in Fig. 5. One curve from the original plots, labelled "extrapolation", is missing because the script used for doing the extrapolation was not published and has been lost.

Finally, Figure 6 of the original publication has not been reproduced. It contains one curve each from Figs. 2 and 3, combined with experimental data from the Protein Data Bank and a scaled curve using a scaling factor computed by yet another script that was not published and has been lost.

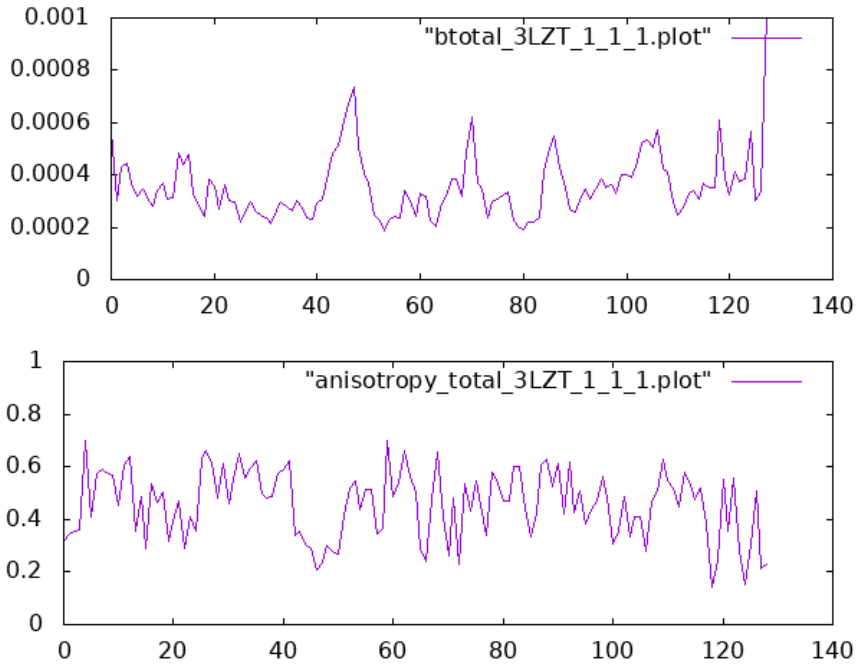


**Figure 2.** Reproduction of Fig. 2 in the original publication.

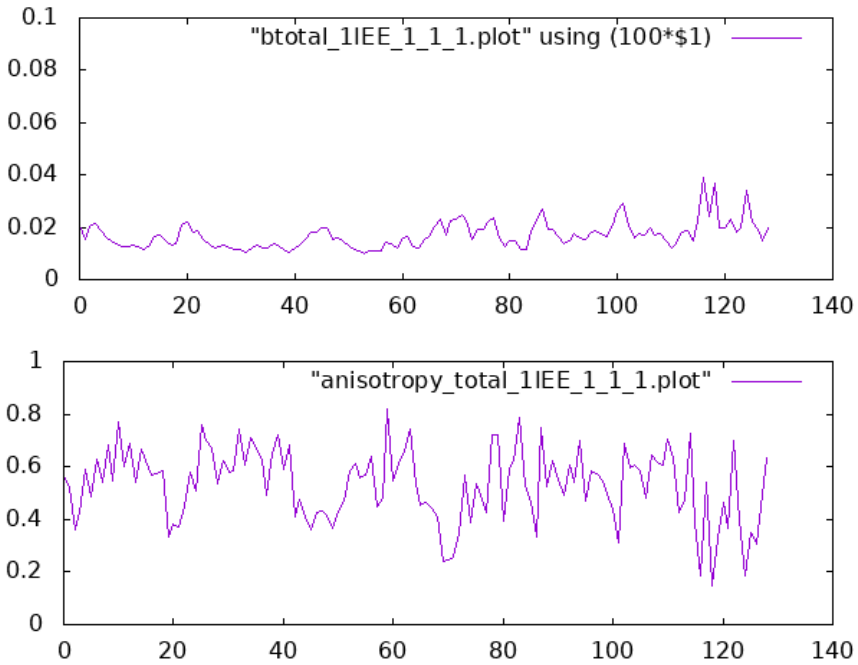


**Figure 3.** Reproduction of Fig. 3 in the original publication.

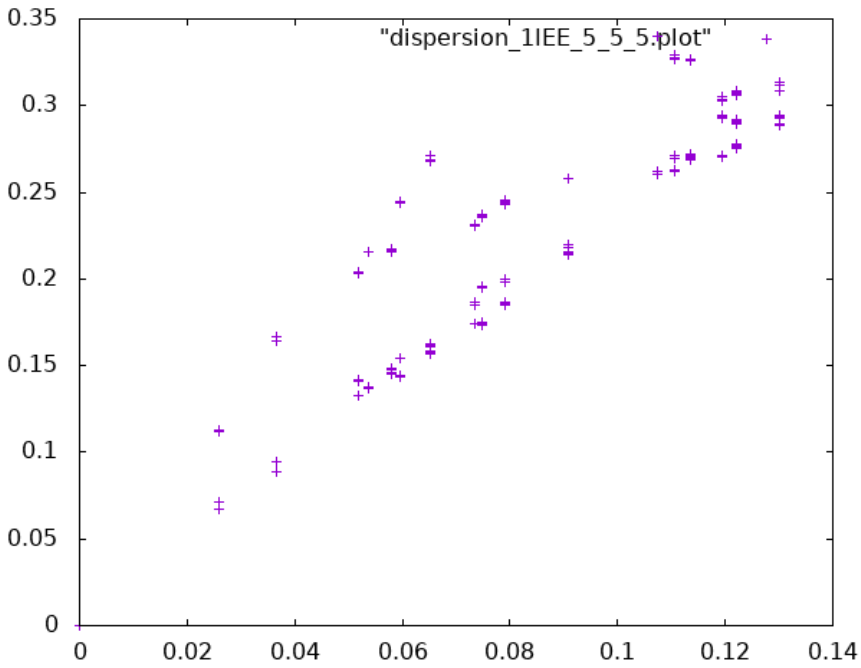


**Figure 4.** Reproduction of Fig. 4 in the original publication.

## Conclusion

The goal of this reproduction attempt was to answer three questions:

1. Can the code published in 2008 still be run today?
2. Does it produce results equivalent to those shown in the original figures?
3. Does the code fully reproduce the original results?

My answers are

1. Yes, but only after modification due to a breaking change in a dependency.
2. Yes.
3. No, because not all the required code was published, and the unpublished code has been lost in the meantime.

The obvious lesson for the future to draw from this exercise is the importance of publishing all the code, up to the automatic generation of figures and tables. Another regrettable omission I made in 2008 is not writing down the precise version numbers of all the code involved. It might have been useful in this case to know the precise version of NumPy used in the original work. In fact, my claim that the modification to a script was required as a consequence of a breaking change in NumPy is based merely on my memories and notes from other projects in which I had to make similar changes.
The final lesson would have to be drawn by a wider community of scientific software developers: breaking changes in widely used infrastructure code such as NumPy can cause a lot of damage in terms of lost reproducibility that may be difficult to diagnose and fix for someone else than the original authors. The open question that the scientific community has to figure out is where to place reproducibility on our scale of values, and for which time spans we consider it desirable. Structural biology is a methodologically

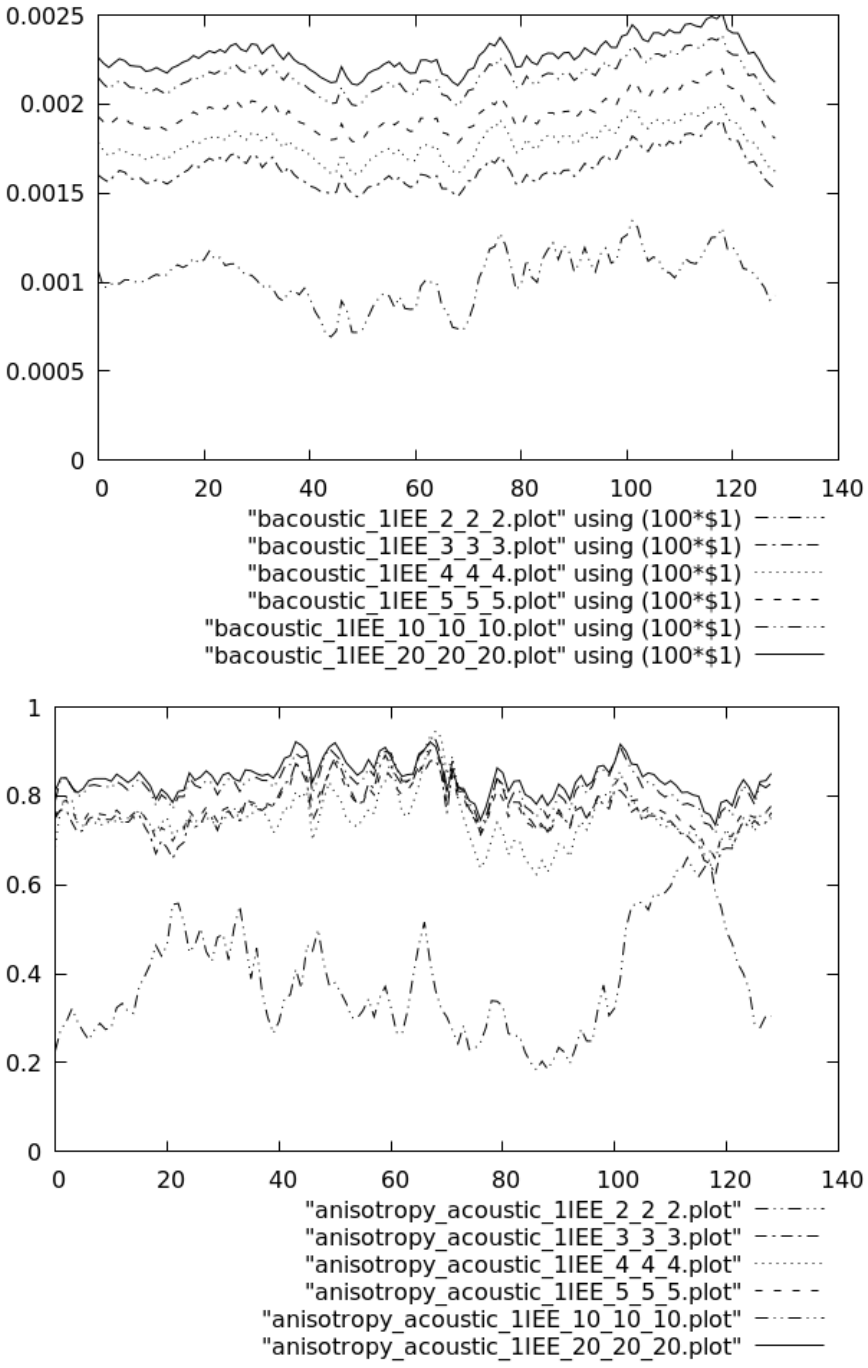


**Figure 5.** Reproduction of Fig. 5 in the original publication.

mature domain of research, in which the main source of progress is not disruptive new methods, but incremental improvements of existing methods in the course of ongoing applications to biologically relevant systems. This is in fact the norm in science and technology [12]. In this context, my methodological work published in 2008 is by no means outdated. The last time I have referred to it myself (in so-far unpublished work) was in 2018. Reproducibility lifetimes of just a few years, corresponding to the current habits in the scientific Python ecosystem, are therefore problematic.

## References

1. K. Hinsen. "Structural Flexibility in Proteins: Impact of the Crystal Environment." In: **Bioinformatics** 24.4 (2008), pp. 521–528.
2. K. Hinsen. "The Molecular Modeling Toolkit: A New Approach to Molecular Simulations." In: **J Comput Chem** 21.2 (2000), pp. 79–85.
3. P. F. Dubois, K. Hinsen, and J. Hugunin. "Numerical Python." en. In: **Computers in Physics** 10.3 (1996), p. 262.
4. T. E. Oliphant. **A Guide to NumPy**. Trelgol Publishing, 2006.
5. G. Ewing. **Pyrex**.
6. S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. "Cython: The Best of Both Worlds." In: **Computing in Science Engineering** 13.2 (Mar. 2011), pp. 31–39.
7. L. Courtès and R. Wurmus. "Reproducible and User-Controlled Software Environments in HPC with Guix." In: **European Conference on Parallel Processing**. Springer, 2015, pp. 579–591.
8. R. Wurmus, B. Uyar, B. Osberg, V. Franke, A. Gosdschan, K. Wreczycka, J. Ronen, and A. Akalin. "PiGx: Reproducible Genomics Analysis Pipelines with GNU Guix." en. In: **GigaScience** 7.12 (Dec. 2018).
9. wwPDB consortium et al. "Protein Data Bank: The Single Global Archive for 3D Macromolecular Structure Data." en. In: **Nucleic Acids Research** 47.D1 (Jan. 2019), pp. D520–D528.
10. T. Williams and C. Kelley. **Gnuplot**. http://www.gnuplot.info/. 2020.
11. G. D. Team. **Grace**. http://plasma-gate.weizmann.ac.il/Grace/. 2008.
12. W. B. Arthur. **The Nature of Technology: What It Is and How It Evolves**. eng. OCLC: 695688782. New York, NY: Free Press, 2009.