Bayesian protein superposition using Hamiltonian Monte Carlo

Lys Sanz Moreta[1*], Ahmad Salim Al-Sibahi[1*], and Thomas Hamelryck[1,4*]

[1] Department of Computer Science. University of Copenhagen, Denmark
[4] The Bioinformatics Centre. Section for Computational and RNA Biology. University of Copenhagen, Denmark

[*] Corresponding authors.
moreta@di.ku.dk (Lys Sanz Moreta),
ahmad@di.ku.dk (Ahmad Salim Al-Sibahi),
thamelry@bio.ku.dk (Thomas Hamelryck)

***Abstract*—Optimally superimposing protein structures is essential to study their structure, function, dynamics and evolution. We present THESEUS NUTS (No U-Turn Sampler), a Bayesian version of the THESEUS model [1]–[3] which relies on maximum likelihood estimation. The probabilistic model interprets each protein as a rotated and translated noisy observation of a latent mean structure. Unlike conventional methods [4], THESEUS takes into account the differences in correlations between the atoms in the structure. This paper extends the previous THESEUS MAP (Maximum A Posteriori) model, [5] to full Bayesian inference by making use of the iterative NUTS [6], a Hamiltonian Monte Carlo method. The model delivers consistent results and is computationally efficient thanks to its implementation in the probabilistic programming language NumpPyro [7], [8] which in turn relies upon JAX [9], a system for high-performance machine learning.**

***Index Terms*—protein superposition, Bayesian modelling, probabilistic programming, NUTS, Hamiltonian Monte Carlo, protein structure superposition**

I. Introduction

Superimposing proteins optimally is crucial to compare their structures. The standard method minimizes the root mean squared deviation (RMSD) of the distances between paired atoms. This approach centers the proteins to the origin and calculates the rotation matrix by singular value decomposition [4] or quaternion algebra [10], [11]. These methods presume that all atoms positions share equal variance (homoscedasticity). This becomes problematic in the case of proteins with flexible loops or flexible terminal regions, where some of the atoms can exhibit high variance. The THESEUS model [1], [2] solved this issue by allowing identification of atoms with higher or lower variance (heteroscedasticity), corresponding to variable and conserved regions.

Previously we implemented the THESEUS model in the probabilistic programming language Pyro [8] (THESEUS MAP). This made it possible to calculate a maximum a posteriori (MAP) estimate using automated stochastic variational inference (SVI) and suitable priors [5].

Here, we present a fully Bayesian version of THESEUS based on iterative NUTS sampling. The model is implemented in the deep probabilistic programming language NumPyro which facilitates high-performance machine learning research thanks to its JAX backend [9]. JAX is an extensible system for composable function transformations which allows the implementation of sophisticated algorithms with high performance in Python.

II. Methods

A. Algorithm summary

THESEUS considers the observed structures, X_1 and X_2, as distributed according to a multivariate matrix normal distribution with latent mean structure $\mathbf{M}$, with Nx3 size, and covariance matrices $\mathbf{U}$ and $\mathbf{V}$. $\mathbf{X}_2$ is rotated ($\mathbf{R}$) and translated ($\mathbf{T}$) to achieve the optimal superposition 2. The rotation matrix is represented by quaternions in order to formulate a uniform prior over the space of rotations.

$$\mathbf{X}_1 \sim \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V}). \tag{1}$$

$$\mathbf{X}_2 \sim \mathcal{MN}(\mathbf{MR} - \mathbf{1}_N\mathbf{T}, \mathbf{U}, \mathbf{V}). \tag{2}$$

In practice, the matrix-normal distributions of $\mathbf{X}_1$ and $\mathbf{X}_2$ reduce to a product of multivariate normal distributions. For details on the model and its prior distribution we refer to [5].

B. Hamiltonian Monte Carlo methods

Hamiltonian Monte Carlo (HMC) is a Markov chain Monte Carlo (MCMC) algorithm that replaces random walk dynamics with Hamiltonian dynamics which rely on gradient information to perform the sampling. The No-U-Turn Sampler (NUTS) [12] is an HMC extension that allows automatically tuning the required hyperparameters (the step size and the number of steps).

For our model, we rely on the iterative NUTS implementation in Numpyro to perform the sampling. Iterative NUTS [6] improves on conventional NUTS [12] by replacing recursion with iteration in the construction of the binary tree used in the proposal. The iterative approach can be implemented efficiently in JAX [9]. JAX can in turn construct a graph representing the computation, and compile it for efficient execution on modern hardware using optimizations based on linear algebra. These optimizations were critical in making NUTS sampling scale to high-dimensional protein systems with hundreds of atoms, and allow us to retrieve a large number of samples for the posterior distribution in seconds.

C. Inference details

The posterior distribution was approximated by 1000 samples after a burn-in period of 500 samples. The parameters of the inference method were chosen in order to optimize speed: the tree depth was restricted to 10 nodes, the acceptance probability of the samples was set to 0.8 or higher and the chain was initialized by a prior based on the median of 15 samples. We ran a single Markov chain.

III. Materials

The algorithm was tested on several proteins from the RCSB protein database [13] whose structures were determined by Nuclear Magnetic Resonance (NMR). Such files typically contain several models of the same protein. These models represent the structural dynamics of the protein in an aqueous medium, and thus provide structural snapshots that reveal both conserved and variable regions. This makes them challenging targets for conventional RMSD superposition. We used the following structures: 1AB2, 1AHL, 1JE3, 1RLP, 1ZWG, 2HF5, 2KHI and 2LMP, which vary from sizes of 35 to 576 residues.

IV. Results

The computations where carried out on an Intel© Xeon © Gold 6136 CPU @ 3.00GHz and an Nvidia graphic card (Quadro RTX 6000). The model was implemented both in Pyro and Numpyro, re-running them for comparison. The latter implementation was on average approximately 1000 times faster due to the implementation of iterative NUTS within the JAX environment. As the computational speed of the Numpyro implementation was much higher on the CPU than on the GPU, we focus here on the former . The computational times register in I are an average compilation of 10 runs for the Numpyro model. The Pyro version was only run once due to its low computational speed.

An example of a pair of superimposed structures is shown in 1. For comparison, the superposition resulting from the conventional RMSD method [4], calculated using Biopython [14], is shown on 1a. The THESEUS MAP superposition is shown in 1b. Finally, the THESEUS NUTS superposition is displayed in 1c. These images illustrate the differences between these different methods and their abilities to handle regions with high and low variability. Supplementary figures of superimposed structures exhibiting different degrees of variability can be found in the Appendix.

TABLE I

Results of applying THESEUS NUTS to the test structures. First column: PDB identifier. Second column: the number of C_α atoms used in the superposition. Third column: the model identifiers. Fourth column: CPU running time for the Numpyro model. Fifth column: GPU running time for Pyro model.

PBD ID	Length C_α	Protein Models	CPU Time (Numpyro)	GPU Time (Pyro)
1AB2	109	0 and 3	24.3s	1h07m35s
1AHL	49	0 and 2	19.1s	1h05m44s
1JE3	97	0 and 1	22s	1h05m35s
1RLP	65	0 and 5	26s	58m48s
1ZWG	35	0 and 3	20s	52m30s
2HF5	68	0 and 3	30.9s	1h04m37s
2KHI	95	0 and 1	25.1s	1h05m24s
2LMP	576	0 and 3	154.5s	57m25s

V. Conclusion

Here, we present a Bayesian model for protein structure superposition implemented in the deep probabilistic programming language Numpyro [7], [8] with JAX [9] as its back-end . This is the first time that the full Bayesian posterior over the parameters of protein superposition is inferred.

The results achieved using Bayesian inference suggest that THESEUS could be potentially used as a suitable error model for probabilistic protein structure prediction. The THESEUS model has the potential to distinguish among partially correct and utterly incorrect predictions when used as a potential likelihood model. Our results show that it is capable of delivering a rich posterior with multiple or unique superposition solutions, as seen for example in Fig. 1 or Fig. 4 in the Supplementary section, respectively.

Contributions and Acknowledgements

Implemented algorithm in NumPyro and Numpy JAX: *LSM* and *ASA*. Wrote article: *LSM*, *TH*, *ASA*. Performed experiments: *LSM*. Designed experiments: *TH*. *LSM* and *ASA* acknowledge support from the Independent Research Fund Denmark (grant: "Resurrecting ancestral proteins in silico to understand how cancer drugs work") and "Deep Probabilistic Programming for Protein Structure Prediction" grant from Independent Research Fund Denmark(DFF), respectively. We thank the Numpyro team for bench-marking the model and further suggestions on how to increase the speed performance.

Data and Software Availability

NumPyro [7], [8] and Numpy JAX [9] based code is a available at https://github.com/LysSanzMoreta/BayesTheseus-PP

References

[1] D. L. Theobald and D. S. Wuttke, "Empirical Bayes hierarchical models for regularizing maximum likelihood estimation in the matrix Gaussian Procrustes problem," *PNAS*, vol. 103, pp. 18 521–18 527, 2006.

[2] D. L. Theobald and P. A. Steindel, "Optimal simultaneous superpositioning of multiple structures with missing data," *Bioinformatics*, vol. 28, pp. 1972–1979, 2012.

[3] K. Mardia and I. Dryden, "The statistical analysis of shape data," *Biometrika*, vol. 76, no. 2, pp. 271–281, 1989.

[4] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Cryst. A*, vol. 34, pp. 827–828, 1978.

[5] L. S. Moreta, A. S. Al-Sibahi, D. Theobald, W. Bullock, B. N. Rommes, A. Manoukian, and T. Hamelryck, "A probabilistic programming approach to protein structure superposition," in *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, July 2019, pp. 1–5.

[6] N. P. Du Phan, "Iterative NUTS," https://github.com/pyro-ppl/numpyro/wiki/Iterative-NUTS, 2019.

[7] D. Phan, N. Pradhan, and M. Jankowiak, "Composable effects for flexible and accelerated probabilistic programming in numpyro," *arXiv preprint arXiv:1912.11554*, 2019.

[8] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-403.html

[9] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

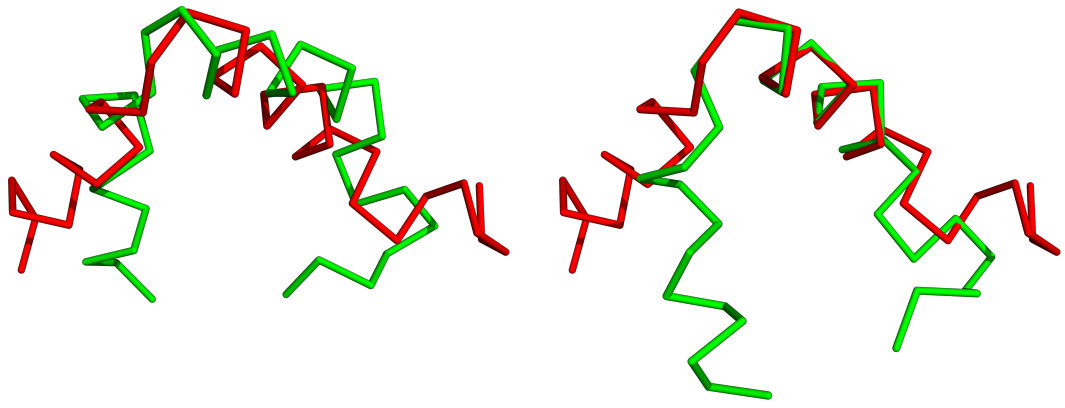
[10] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, pp. 629–642, 1987.

[11] E. Coutsias, C. Seok, and K. Dill, "Using quaternions to calculate rmsd," *J. Comp. Chem.*, vol. 25, pp. 1849–1857, 2004.

[12] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, 2014.

[13] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

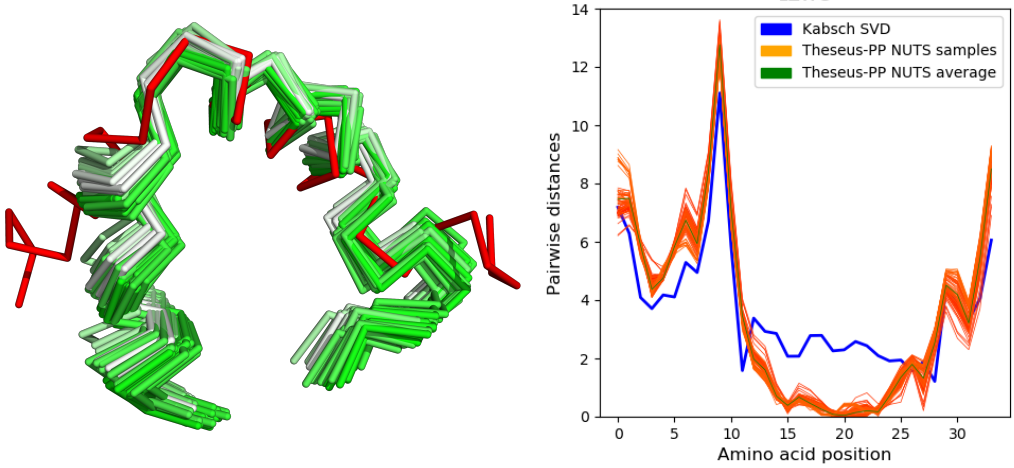
[14] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.



(a) Kabsch-RMSD

1ZWG

(b) Theseus MAP

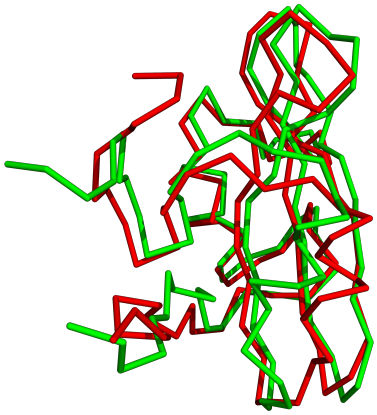


(c) Theseus NUTS

(d) Pairwise Distances NUTS

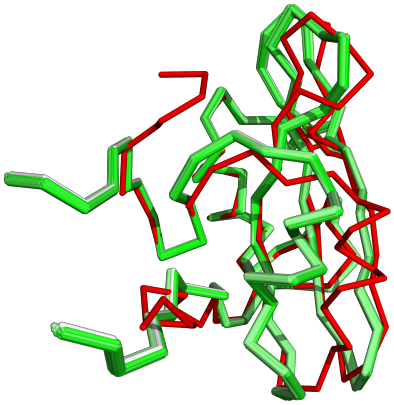
Fig. 1. Protein superposition for two conformations of protein 1ZWG obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.

VI. Appendix

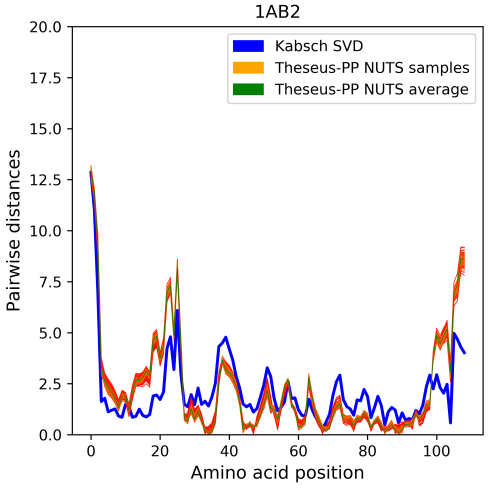


(a) Kabsch-RMSD

(b) Theseus MAP

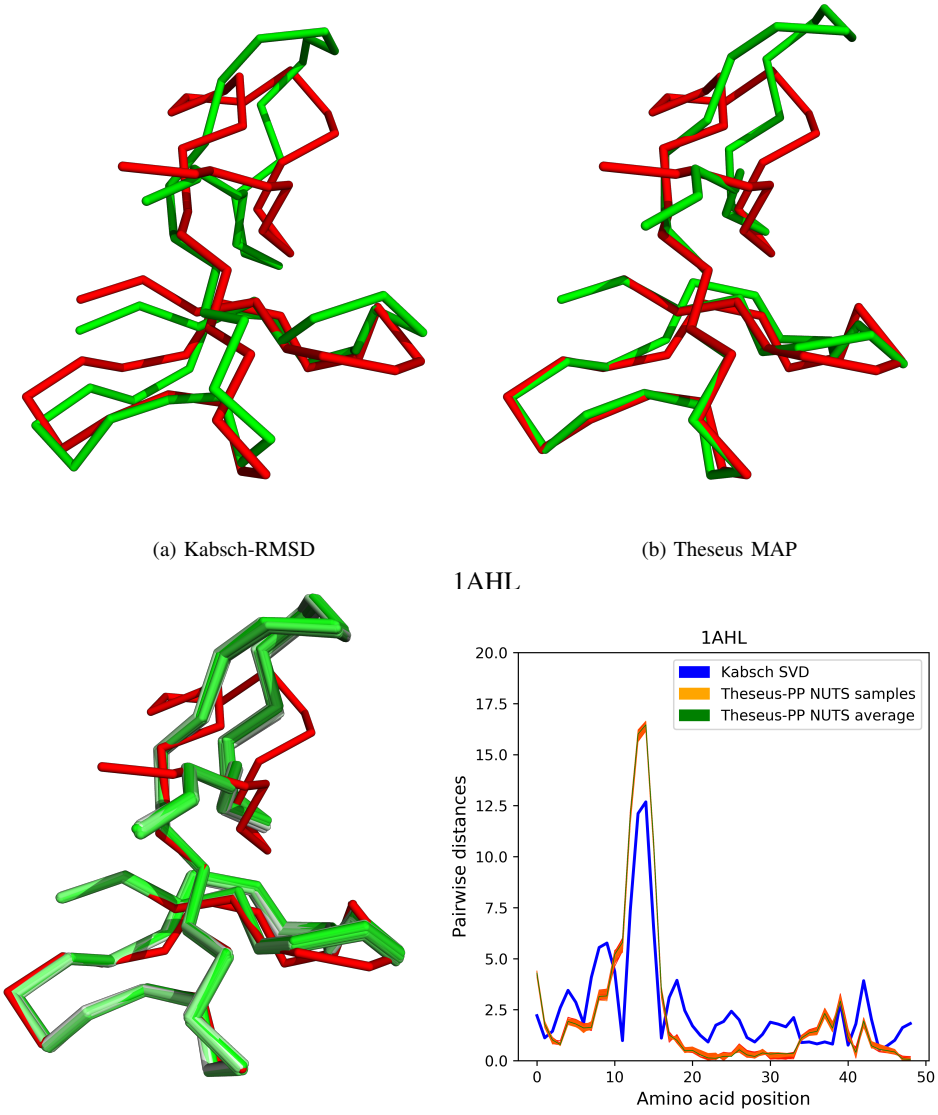


(c) Theseus NUTS



(d) Pairwise Distances NUTS

Fig. 2. Superposition of two conformations of protein 1AB2 obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.



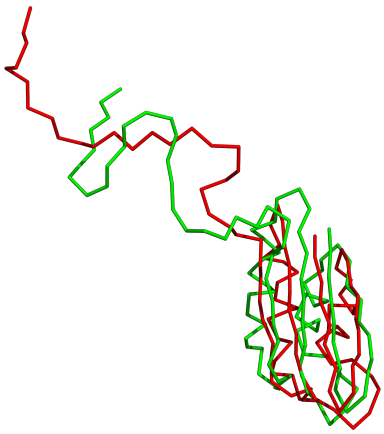
(a) Kabsch-RMSD

(b) Theseus MAP

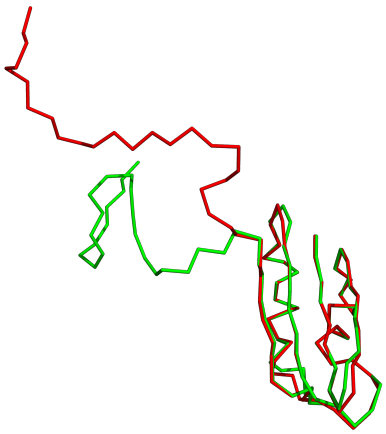
(c) Theseus NUTS

(d) Pairwise Distances NUTS

Fig. 3. Protein superposition of two conformations of protein 1AHL obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.

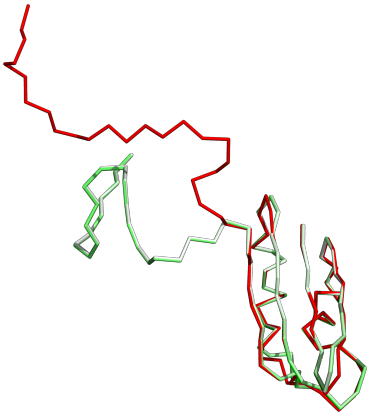


(a) Kabsch-RMSD

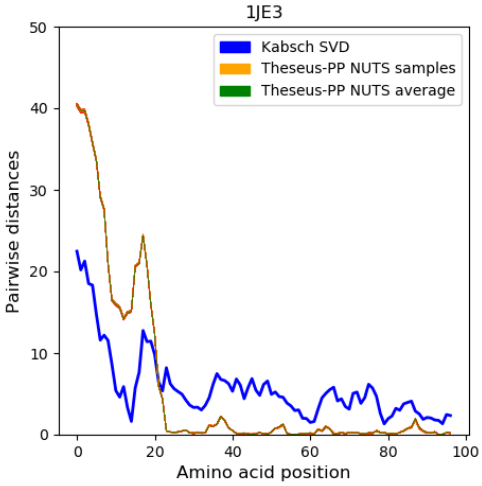


(b) Theseus MAP

1JE3



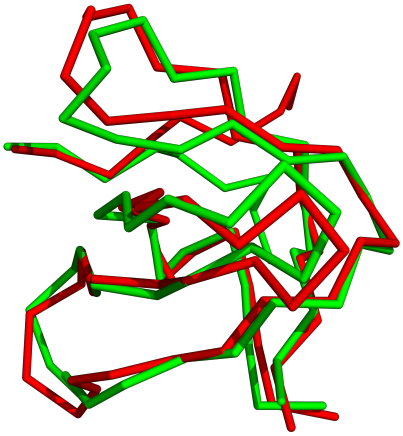
(c) Theseus NUTS



(d) Pairwise Distances NUTS

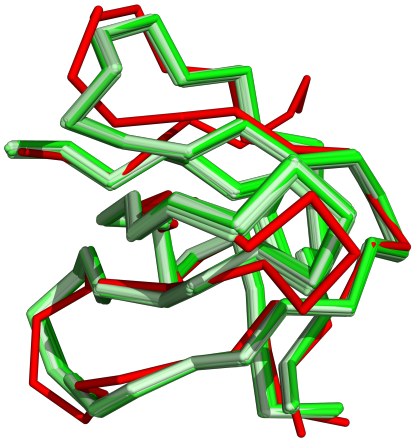
Fig. 4. Superposition of two conformations of protein 1JE3 obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.

(a) Kabsch-RMSD

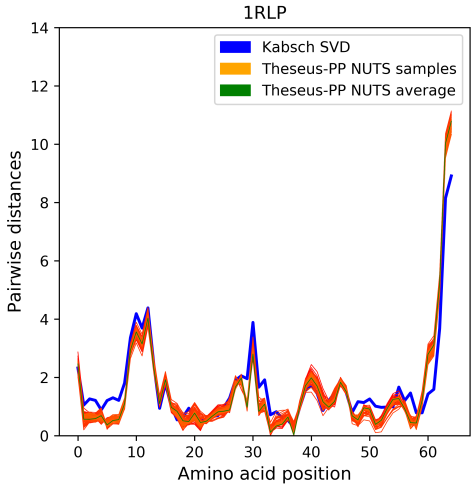


(b) Theseus MAP

1RLP

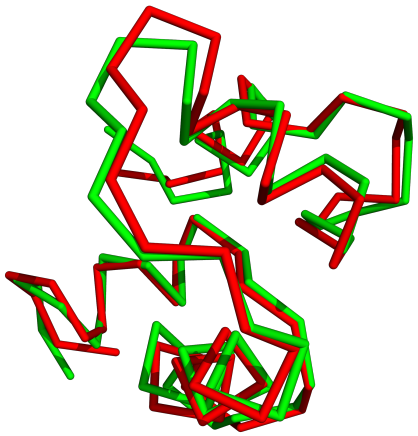


(c) Theseus NUTS

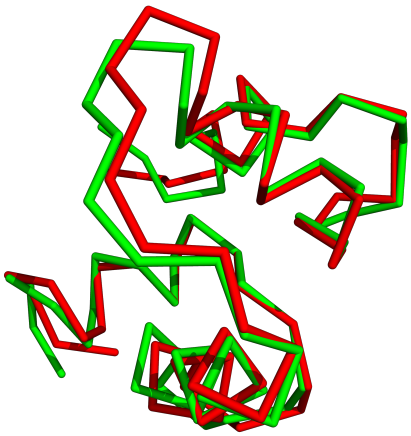


(d) Distances NUTS

Fig. 5. Superposition of two conformations of protein 1RLP obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.

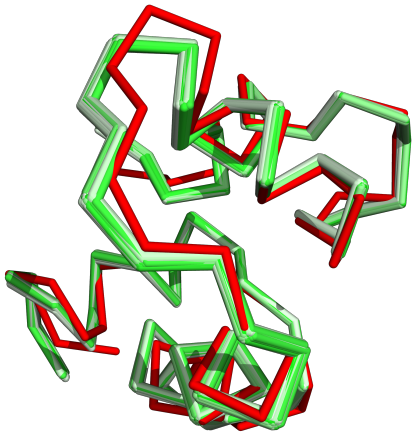


(a) Kabsch-RMSD

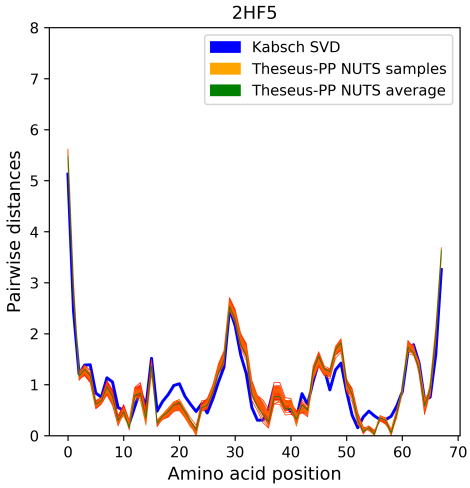


(b) Theseus MAP

2HF5

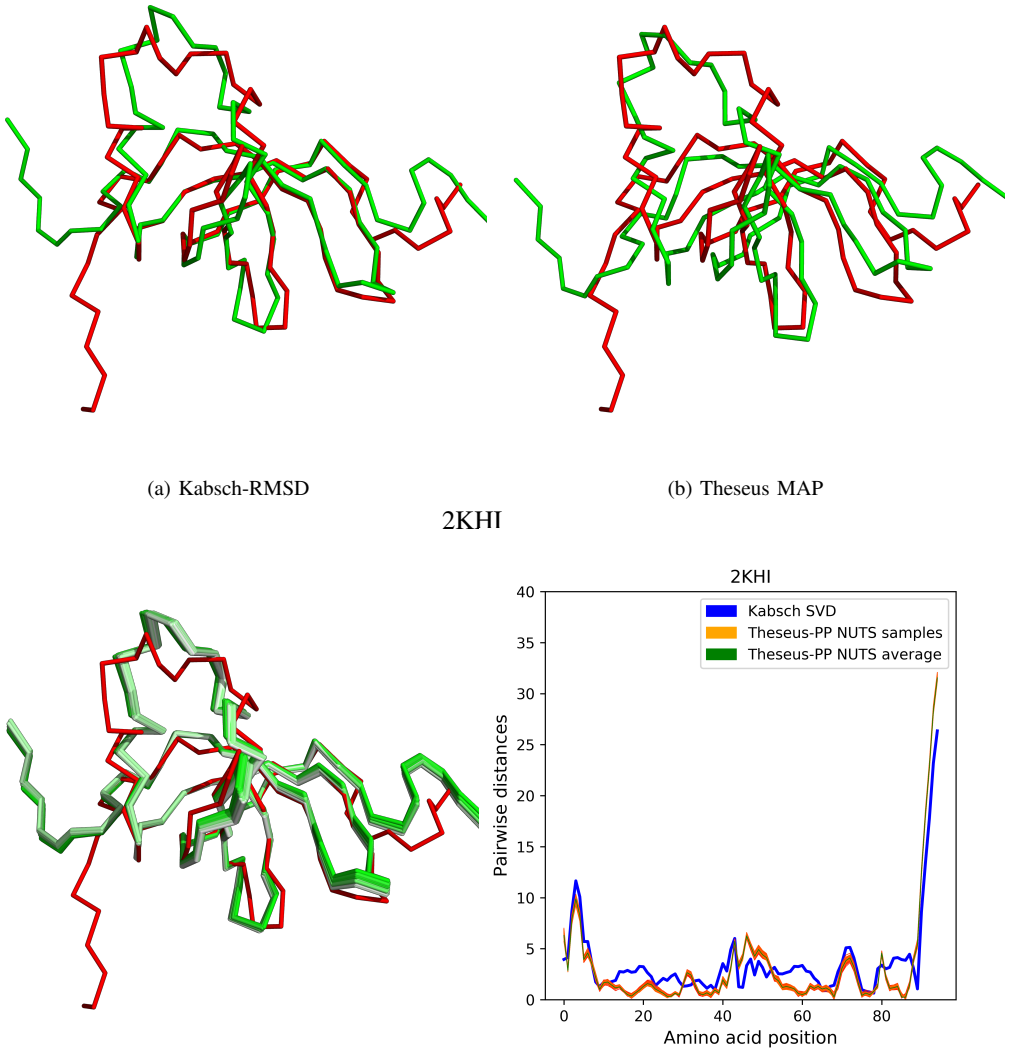


(c) Theseus NUTS



(d) Distances NUTS

Fig. 6. Superposition of two conformations of protein 2HF5 obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.



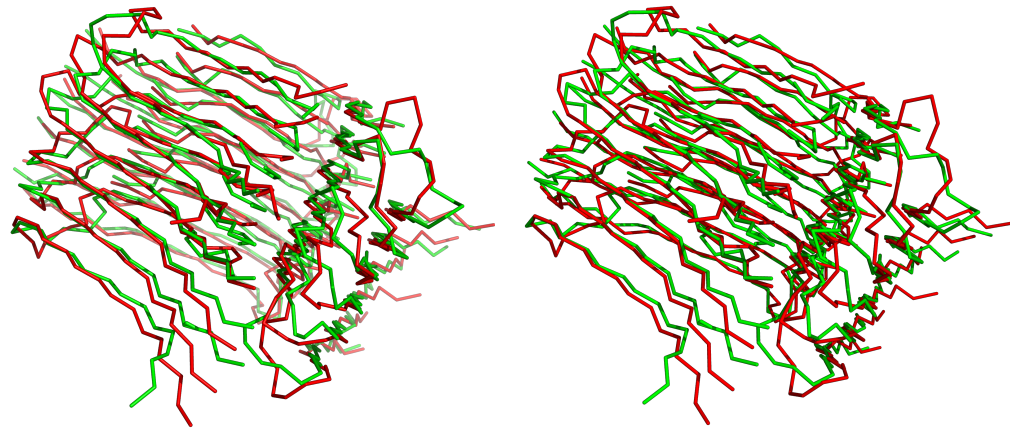
(a) Kabsch-RMSD

(b) Theseus MAP

(c) Theseus NUTS

(d) Distances NUTS

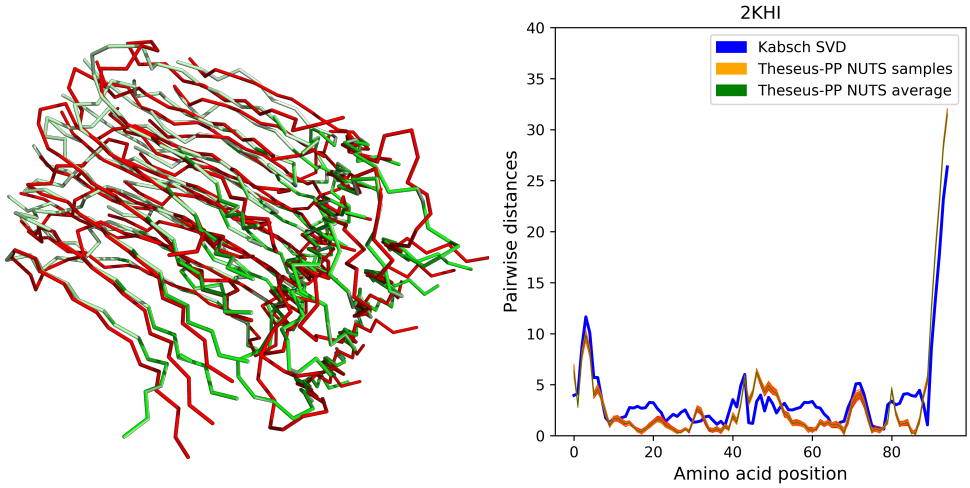
Fig. 7. Superposition of two conformations of protein 2KHI obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.



(a) Kabsch-RMSD

(b) Theseus MAP

2KHI



(c) Theseus NUTS

(d) Distances NUTS

Fig. 8. Superposition of two conformations of protein 2KHI obtained from (a) conventional RMSD superimposition and (b) THESEUS MAP and (c) THESEUS NUTS. The protein in green is rotated (X_2). Graph (c) shows the pairwise distances (in Å) between the C_α coordinates of the structure pairs. The blue and orange lines represent RMSD and THESEUS-PP superposition, respectively.