Check for updates

SOFTWARE TOOL ARTICLE

PySteMoDA: An open source Python package for the analysis of steered molecular dynamics simulations data

[version 1; peer review: 2 approved with reservations]

Ismahene Mesbah[1], Colin Klaus[2], Marcos Sotomayor[2], Fidan Sumbul[1], Felix Rico[1]

[1]Aix-Marseille University, INSERM, DyNaMo, Turing centre for living systems, Marseille, 13009, France
[2]Department of Biochemistry and Molecular Biology and Center for Mechanical Excitability, The University of Chicago, Chicago, Illinois, 60637, USA

v1 **First published:** 28 Mar 2026, **6**:84
https://doi.org/10.12688/openreseurope.22884.1
Latest published: 28 Mar 2026, **6**:84
https://doi.org/10.12688/openreseurope.22884.1

Abstract

Molecular dynamics simulation is a powerful computational technique used for predicting and understanding the dynamic behavior of biomolecular systems. Steered molecular dynamics (SMD) simulations enable the study of force-induced processes in biomolecules, effectively mimicking single-molecule force spectroscopy experiments probing protein unfolding and receptor-ligand unbinding. Given the stochastic nature of these mechanical events, accurately exploring the dynamic behavior of biomolecules and extracting accurate physical information requires several *in-silico* experiments. This includes performing many pulling simulations at different velocities or force loading rates. The large amount of data obtained from these simulation sets requires efficient automated data processing tools. We present PySteMoDA, a novel Python package with a user-friendly graphical interface specifically designed for constant-velocity SMD data analysis. The automated force peak detection methods reduce user bias, improve accuracy, and accelerate data analysis. The package also allows identification of residues involved in mechanical events through computation of the time-dependent mechanical work and correlation factors between residue pairs. This package not only addresses automated data processing in SMD simulations and accurate parameter extraction, but also significantly enhances accessibility and usability. Through PySteMoDA, users can efficiently analyze simulation data without the barrier of coding, facilitating a wider range of investigations and insights in the field of computational biochemistry and biophysics.

Open Peer Review

Approval Status ? ?

	1	2
version 1 28 Mar 2026	? view	? view

1. **Gustavo E. Olivos-Ramirez**, Instytut Podstawowych Problemow Techniki Polskiej Akademii Nauk (Ringgold ID: 86911), Warsaw, Poland
Adolfo Poma, Polish Academy of Sciences,, ul. Pawińskiego 5B,, Poland

2. **Rafael Delgado-Buscalioni**, Universidad Autonoma de Madrid, Campus de Cantoblanco,, Madrid 28049,, Spain

Any reports and responses or comments on the article can be found at the end of the article.

Keywords
force spectroscopy, molecular force probe, protein unfolding, receptor-ligand unbinding, biophysics, mechanobiology

This article is included in the European Research Council (ERC) gateway.

This article is included in the Horizon 2020 gateway.

Corresponding authors: Ismahene Mesbah (ismahene.mesbah@inserm.fr), Felix Rico (felix.rico@inserm.fr)

Author roles: Mesbah I: Conceptualization, Investigation, Methodology, Project Administration, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Klaus C**: Methodology, Resources, Validation, Writing – Review & Editing; **Sotomayor M**: Funding Acquisition, Methodology, Project Administration, Resources, Supervision, Validation, Writing – Review & Editing; **Sumbul F**: Conceptualization, Investigation, Methodology, Supervision, Validation, Writing – Review & Editing; **Rico F**: Conceptualization, Funding Acquisition, Investigation, Methodology, Project Administration, Resources, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No [772257]. Also, funding from France 2030, the French Government programme managed by the French National Research Agency (ANR) under grant agreement No. ANR-16-CONV-0001; the Excellence Initiative of Aix-Marseille University (A*MIDEX) and the U.S. National Institutes of Health, National Institute on Deafness and Other Communication Disorders (NIH/NIDCD R01 DC015271); and the Human Frontier Science Program (HFSP, grant No. RGP0056/2018).

Copyright: © 2026 Mesbah I *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Mesbah I, Klaus C, Sotomayor M *et al.* **PySteMoDA: An open source Python package for the analysis of steered molecular dynamics simulations data [version 1; peer review: 2 approved with reservations]** Open Research Europe 2026, **6**:84 https://doi.org/10.12688/openreseurope.22884.1

First published: 28 Mar 2026, **6**:84 https://doi.org/10.12688/openreseurope.22884.1

Introduction

Molecular dynamics (MD) simulations used as "computational microscopes" enable atomic-level prediction and understanding of biomolecular processes such as side chain fluctuations and conformational dynamics of proteins.[1–6] MD simulations of biomolecules are commonly carried out using packages such as CHARMM,[7,8] OpenMM,[9] AMBER,[10] GROMACS,[11,12] and NAMD.[13] NAMD is open source, easy to use, highly optimized for massively parallel supercomputers, and compatible with the visual molecular dynamics (VMD) software[14] that permits interactive analysis of simulation outputs. Steered molecular dynamics (SMD) simulations represent a powerful computational technique that mimics single-molecule force spectroscopy (SMFS) experiments and, thus, allows for the exploration of molecular interactions and conformational changes during relevant biological processes where mechanical force is involved.[15–18] By applying external forces to specific atoms or molecules, SMD enables the study of processes such as protein unfolding, ligand-receptor unbinding, and the mechanical probing of macromolecular structures.[19,20] In addition, SMD can be used to accelerate processes that otherwise occur over timescales beyond those accessible to standard equilibrium MD simulations.[21] The application of SMD has been instrumental in elucidating the pathways and energetics of mechanically induced conformational transitions and intermediate states that are often critical to the function of biomolecules. For example, simulations of mechanical unfolding in proteins—such as titin in muscle sarcomeres[20]—and ligand-receptor unbinding events, like (strept) avidin and biotin,[19,22,23] offer valuable insights into the mechanisms and forces underlying these processes, including those that involve highly stable interactions.[24] The ability to model these forces and to predict and understand their effects at a microscopic level is essential for the design of new drugs, the engineering of protein-based materials, and the development of nanoscale devices.

SMFS experiments probe the mechanical properties of biomolecules, while SMD simulations provide atomistic insight into their force-induced responses. For instance, SMD revealed that unfolding of the titin I27 (I91) domain begins with separation of the A and B β-strands, followed by disruption of the A' and G β-strands.[20,25] Early SMD studies were limited to pico- to nanosecond timescales, which made direct comparison with slower SMFS experiments difficult.[22,26] Despite this, their molecular predictions were later validated at millisecond timescales.[27] Advances in high-speed SMFS and long-timescale SMD now allow overlapping timescales,[23,28] and many relevant mechanical events occur within microseconds.[29–33] This timescale is accessible with modern MD engines and supercomputers.[2,34]

In SMD simulations, the pulling velocity or force is kept constant. In constant-velocity simulations a virtual spring is attached to an atom or group of atoms, and the free end of this spring is moved at a constant velocity, applying a force proportional to the spring extension that induces conformational changes in the biomolecule. In constant-force SMD simulations a force is directly applied to an atom or group of atoms in a predetermined direction. In both constant-velocity and constant-force simulations, the protein is either stretched from one end by keeping the other end fixed or from both ends in opposite directions. Other configurations for force application, including length clamp,[35] can be implemented as well.

As a typical constant-velocity SMD simulation progresses, the force applied to the molecule and atomic coordinates of each atom are recorded at defined time intervals. Different parameters are then extracted from simulation trajectories, including unfolding forces, molecular extensions, molecular stiffnesses and loading rates. For many systems, it is now possible with modern computers to carry out hundreds of simulations for more accurate statistical analysis of these dynamic ensembles.[23,28,36–38] However, manually analyzing the large amount of data generated by SMD simulations is time-consuming and prone to error. To address this challenge, we introduce PySteMoDA, a Python package designed to semi-automatically extract and analyze data from SMD simulations.

PySteMoDA is an open-source, freely accessible, semi-automated, and user-friendly package that facilitates the analysis of the output files of constant-velocity SMD simulations performed using NAMD. It features a graphical user interface (GUI) allowing users to carry out advanced analysis without needing to write code. Additionally, a Google Colab version is available, enabling users to run the pipeline directly without local installation. Once configured, the entire pipeline, including data processing, peak detection, loading rate calculation, correlation analysis, and inter-residues deformation measurement, runs automatically using the default values provided by the platform. Both the GUI- and Colab-based implementations significantly lower the barrier to entry and improve usability for researchers across disciplines interested in SMD analysis.

Material and methods

1. Implementation

PySteMoDA was developed in Python 3.8 and requires several package dependencies including Tkinter,[39] ProDy,[40] scikit-learn[41] and SciPy.[42] Our package adopts an object-oriented model, enhancing its scalability, modularity, and reusability. This design choice enables users to efficiently extend its functionality to suit their specific research needs.

A user-friendly GUI ensures that even newcomers to the field can quickly become proficient with PySteMoDA. This simplicity enables users to save time and easily perform their analysis. The package is executable on Linux, Mac, and Windows operating systems. The PySteMoDA workflow involves inputting file names, extracting data, doing analyses and exporting results from analyses (Figure 1). A Jupyter notebook have also been implemented to allow batch processing from the command line. Additionally, a Google Colab version is available at https://colab.research.google.com/drive/1Cc1IqlMnsJ-SDoE3CIiMhnLHVN5y61Dp?usp=sharing, enabling users to run the pipeline directly without local installation.

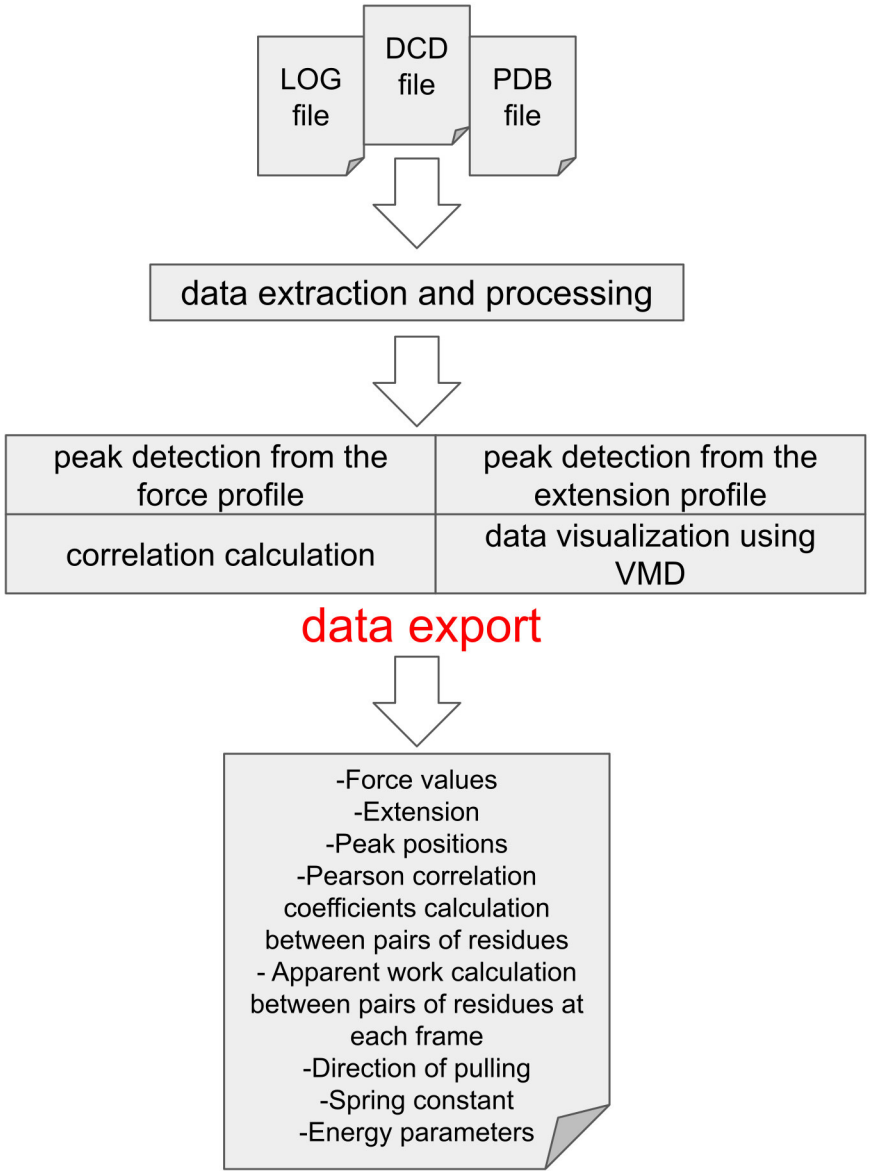


Figure 1. Schematic representation showing the PySteMoDA workflow. Analysis starts by loading the LOG and DCD files obtained from SMD simulations using NAMD in addition to the PDB file containing the atomic coordinates. Users can perform peak detection either from the force or extension curves. The Pearson coefficient of correlation and apparent work is calculated between pairs of residues at different frames, which allows users to observe the effect of changes in distance between residues on force. The simulation movie can be visualized via VMD where the DCD and PDB files are uploaded automatically by PySteMoDA. At the end of the analysis, users have the possibility to export the analysed data in different formats.

2. Simulations

Representative data used to validate and showcase PySteMoDA in this manuscript were obtained from two sets of NAMD SMD simulations. The first set corresponds to SMD simulations of a cadherin-23 (CDH23) subdomain including extracellular cadherin (EC) repeats EC19–20 with three bound calcium ions (extracted from PDB: 5TFK and hereafter referred to as CDH23 EC19–20).[43,44] Details of these simulations are described elsewhere. Briefly, simulations of the CDH23 EC19–20 system (n_{atoms} = 244,076 atoms) were performed using NAMD 2.12[13] with the CHARMM36 force field[8] and the TIP3P water model. Van der Waals forces were computed using a 12 Å cutoff with a switching function starting at 10 Å. Electrostatic forces were computed using the particle mesh Ewald algorithm with a grid spacing of 1 Å. Dynamics in time was advanced using a 2 fs timestep. The system was first minimized for 1,000 steps, then subjected to 50,000 steps of constrained backbone dynamics ($k \sim 700$ pN/nm). A Langevin thermostat and hybrid Nosé-Hoover piston method were used to maintain a temperature of 300 K and pressure of 1 atmosphere (*NpT* ensemble). In the first 1 ns a Langevin damping coefficient of 1 ps^{-1} was used for the thermostat while in the following 10 ns a Langevin damping coefficient of 0.1 ps^{-1} was used. A piston period of 200 fs and a damping timescale of 100 fs were used for pressure control. For SMD simulations, virtual springs were attached to the N- and C-terminal C_α atoms with spring constants of $k \sim 700$ pN/nm. The free ends of the virtual springs were moved at stretching speeds of 5 nm/ns in opposite directions (10 nm/ns total). SMD data output and coordinates were written to disk every 40 fs and every 1 ps, respectively.

The second set of simulations used to validate and showcase PySteMoDA involved the intercellular adhesion molecule 1 (ICAM-1, n_{atoms} = 67,215) which was simulated with NAMD 2.9[13] using the CHARMM36 force field[7,8] and the TIP3P water model. Dynamics in time were advanced using a 2 fs timestep (**See underlying data**[45]). We modelled ICAM-1 based on the cryo-EM structure of D1-D5 (PDB ID: 1Z7Z)[46] and crystal structure of D3-D5 (PDB ID: 2OZ4).[47] Disulfide bonds were specified using the corresponding patch to create the protein structure file (PSF) for simulation. ICAM-1 was stretched at 1 nm/ns along a direction defined by the vector that connected the fixed and SMD atoms after equilibration ($v = 0.0524, 0.3795, 0.9236$). The N-terminal Cα atom was fixed during the simulation. The spring constant corresponds to 104 pN/nm. The simulation was carried out over a total of 70 ns with outputs written every 10 ps for both DCD (trajectory coordinates) and SMD frequencies. In the case of the ICAM-1, the simulation was conducted in the *NVE* ensemble, where the number of particles, volume, and total energy were conserved throughout the run.

3. Data extraction and conversion

PySteMoDA requires NAMD output files, including the standard output (generally saved to a file with extension LOG) and a trajectory DCD file. These files are used to extract data, including external forces applied during the simulation to manipulate the molecule, commonly referred to as "SMD forces". To obtain the SMD forces in the direction of pulling in picoNewton (pN), PySteMoDA extracts raw SMD forces (pN) and the direction of pulling. These are the forces used to mimic either physiological or experimental conditions such as those used in atomic force microscopy (AFM) and optical tweezers (OT). External forces can be applied to atoms in NAMD simulations in different ways, including by using the SMD module, the "tool command language" TCL interface, or the collective variables module. PySteMoDA can read the NAMD LOG file assuming the standard SMD output format ("SMD" *timestep x y z* $f_x f_y f_z$) used for simulations in which one set of atoms is stretched. For simulations in which two groups of atoms are stretched, PySteMoDA assumes that one group is stretched using the SMD module and the other is stretched using the TCL interface with a predefined output format ("TCL:" *timestep x y z* f_x). Both SMD and TCL forces are extracted and can be visualized within the GUI with respect to simulation time or stretching distance. The stretching distance corresponds to the movement of the SMD atom.

The scalar force along the pulling axis is computed as the dot product between the SMD force vector and the pulling direction:

$$F(v) = F_{smd}(Fx, Fy, Fz) \cdot direction_{pulling}(x, y, z)$$

where the resulting force is in pN.

If any TCL forces (kcal/mol/Å) have been applied during the simulation and recorded in the LOG file, PySteMoDA extracts them automatically and converts them to picoNewton:

$$F(-v) = F_{tcl} \times 69.479$$

assuming 1 kcal/mol = 69.479 pN·Å.

The spring constant recorded in kcal/mol/$Å^2$ is converted to pN/nm:

$$k_{(pN/nm)} = k_{\left(kcal/mol/A^2\right)} \times 69.479 \times 10$$

The time is converted to seconds knowing the time step of the simulation, in most of the cases corresponding to 2 femtoseconds, in our simulations:

$$Time_{(seconds)} = TimeStep \times 2 \times 10^{-15}$$

Additionally, PySteMoDA extracts atom positions and time steps from the trajectory DCD file. The package calculates the extensions at each frame that the user provides in the trajectory file. These correspond to the Euclidean distances between the Cα atoms of two given residues $r1$ and $r2$ based on their three-dimensional coordinates at each time step, according to:

$$extension(t) = \sqrt{(x_{r1}(t) - x_{r2}(t))^2 + (y_{r1}(t) - y_{r2}(t))^2 + (z_{r1}(t) - z_{r2}(t))^2}$$

The extension is calculated by default at all the frames between the Cα of the first residue (*r1*) and the Cα of the last residue (*r2*), although users can select any other residue or frame coordinates.

The correlation matrix of forces and distances between pairs of residues is obtained by calculating the Pearson coefficient of correlation. To achieve this, first, PySteMoDA calculates the center of mass (COM) of each amino acid residue for each Cartesian coordinate as:

$$x_{COM} = \frac{\sum_{i=1}^{N} m_i . x_i}{\sum_{i=1}^{N} m_i}$$

$$y_{COM} = \frac{\sum_{i=1}^{N} m_i . y_i}{\sum_{i=1}^{N} m_i}$$

$$z_{COM} = \frac{\sum_{i=1}^{N} m_i . z_i}{\sum_{i=1}^{N} m_i}$$

where x_i, y_i, z_i are the cartesian coordinates of the ith atom, m_i is its atomic mass, and N is the number of atoms. PySteMoDA uses the COM coordinates (instead of Cα coordinates) because these represent the average position of all the atoms in a residue, weighted by their masses. This provides a single location that reflects the distribution of the entire molecular mass. Therefore, using the center of mass gives a more global, holistic view of the residue's position and movement while Cα coordinates represent only the backbone of the protein. Once the centers of mass of residues are obtained at each frame (k), their Euclidean distances are computed and used as the distance between pairs of residues $(r1, r2)$:

$$d_i(r1, r2) = \sqrt{\left(x1(k)_{COM} - x2(k)_{COM}\right)^2 + \left(y1(k)_{COM} - y2(k)_{COM}\right)^2 + \left(z1(k)_{COM} - z2(k)_{COM}\right)^2}$$

where *x1, y1, z1* are the spatial coordinates of *r1* and *x2, y2, z2* are the coordinate of *r2*. The Pearson correlation "work" coefficient is calculated between forces and distances for each pair of residues that are within 13 Å (default value that can be manually changed) at a given range of n frames from:

$$r_{fd} = \frac{\sum_{i=1}^{n} (d_i - \underline{d})(F_i - \underline{F})}{\sqrt{\sum_{i=1}^{n} (d_i - \underline{d}) \sum_{i=1}^{n} (F_i - \underline{F})}}$$

where d_i is the distance between the pair of residues at frame k, $\underline{d}$ is the mean distance between this pair of residues along the range of frames $[k{:}k + n]$, F_i is the force vector at frame k, and $\underline{F}$ is the mean force along the given range of frames $[k{:}k + n]$.

The **apparent work** (in Joules) is calculated along the trajectory frames via the dot product of the obtained distances vector (Δr) and the force vector (direction of pulling):

$$work(Joules) = \sum_{frame=0}^{N} F \Delta r$$

where F and Δr are the force and distance vectors. The Δr vector is calculated for each pair of residues using the center of mass:

$$\Delta r = (dx, dy, dz)$$

where:

$$dx = x_{COM_{residue_1}} - x_{COM_{residue_2}}$$

$$dy = y_{COM_{residue_1}} - y_{COM_{residue_2}}$$

$$dz = z_{COM_{residue_1}} - z_{COM_{residue_2}}$$

4. Detection and characterization of unfolding/unbinding events

Detection of unfolding/unbinding events was implemented using two different approaches:

4.1 Smoothing and automatic peak detection using the force curve

In this approach, an algorithm scans the force curve to identify local maxima. A local maximum is a point where the value is greater than its neighboring points. These local maxima represent potential specific unfolding/unbinding events in the data. For each identified local maximum, the prominence is calculated where prominence is defined by NumPy's `find_peaks` function. Prominence is a measure of how much a peak stands out from the neighboring peaks. Peaks with a prominence below a certain threshold are considered as noise and are filtered out by PySteMoDA. By default, this threshold is computed automatically as the mean value of all peak prominences. This default thresholding step helps eliminate smaller peaks that may not be of interest. Users can modify this threshold value if a different level of sensitivity is desired.

The interface offers a "natural cubic spline" smoothing filter. In this context, knots are specific points along the *x*-axis where individual cubic polynomial pieces meet and are joined together. A natural cubic spline fits the data by dividing the curve into intervals (or windows) defined by these knots. Within each interval, a cubic polynomial is used, and at each knot the spline is constructed so that the function, its first derivative, and its second derivative are all continuous. The number of knots can be defined by the user or automatically filled within the interface (with knots spaced at regular intervals of approximately 0.3 nm along the *x*-axis). The number of knots directly controls the flexibility of the spline, for example, more knots allow the spline to follow fine-scale variations in the data, while fewer knots result in a smoother, more global fit.

4.2 Semi-automatic peak detection using the extension curve

In this approach, the derivative of the extension-time curve is calculated and plotted. This mathematical operation involves determining the rate of change of extension with respect to time, and the resulting derivative graph serves to detect sharp transitions in the original curve as the increasing steps in the extension appear as sharp peaks in its derivative. The process of calculating the derivative facilitates a more precise and efficient detection of their locations.

4.3 Peak analysis

Once the peaks are detected, relevant parameters such as the loading rate (pN/s) and molecular and effective stiffness are computed automatically, within a given extension window before the peak (1 nm by default, but manually modifiable). The loading rate is determined from the slope of the force versus time signal over the 1 nm default distance before the peak and the effective stiffness as the loading rate divided by the pulling velocity. The molecular stiffness is calculated using the formula:

$$\frac{1}{k_{molecular}} = \frac{1}{k_{effective}} - \frac{1}{k_{virtual}}$$

where $k_{molecular}$ is the molecular stiffness, $k_{effective}$ is the effective stiffness, and $k_{virtual}$ is the virtual spring constant.

5. Inter-residues distance deformation

Finally, an inter-residues distance deformation is calculated. This metric is a useful way to track structural changes in the protein over time at the residue level. It, therefore, provides a measure of how much the spatial arrangement of amino acids deviates from its original configuration.

$$d_i(t) = 1 - \frac{1}{nb}\sum_j exp\left(\frac{-\left(d_{ij}(0) - d_{ij}(t)\right)^2}{{l_d}^2}\right)$$

The formula computes a deformation metric that quantifies the change in the distances between residues in a protein over time. Here, $d_{ij}(0)$ is the initial distance between residue i and j at frame 0, $d_{ij}(t)$ is the distance between them at a later time t, and l_d is a characteristic length scale (usual distance between two amino acids) used here to normalize the obtained value. The sum runs over all residue pairs i, j, and nb represents the total number of pairs being considered. If the distance between amino acids is large the term $exp\left(\frac{-\left(d_{ij}(0)-d_{ij}(t)\right)^2}{l_d{}^2}\right)$ results in smaller values (i.e., more deformation). The final result reflects how much the distances between residues have changed on average over the specified frames.

This formula was adapted to incorporate mechanical work and, thus, to analyze the work due to applied forces:

$$W_i(t) = 1 - \frac{1}{nb}\sum_j exp\left(\frac{-[(dij(t) - dij(0))\times(Force(t) - Force(0))]^2}{k_B T^2}\right)$$

where:

Force: is the applied force in pN.

$k_B T$: is the Boltzmann constant multiplied by the temperature.

The distance deformation measurements can be carried out within the google colab but can also be calculated using the command line. After opening a terminal, users can use the following command to obtain the deformation measurement:

```
python3 distance_deformation.py --pdb pdb_file.pdb --dcd dcd_file.dcd
--frame_start 0 --frame_end 5000 --frame_interval 100 --threshold
13 --distance_change_threshold 13.0 --residue_range 1-200 --output_txt
name_output_file.txt --output_png name_file_plot.png
```

The "threshold" parameter here is the maximum distance to be considered when calculating the deformation metric, the "distance change threshold" parameter is the maximum distance to report in the "output_file" once the residues separate from each other during the pulling.

Results

PySteMoDA is freely available and downloadable (https://gitlab.com/fm4b_lab/pystemoda) and can also be accessed as a Google Colab notebook (https://colab.research.google.com/drive/1Cc1IqlMnsJ-SDoE3CIiMhnLHVN5y61Dp?usp=-sharing). To illustrate PySteMoDA's user-friendly GUI we describe here various steps to carry out analyses of SMD simulations for two systems, CDH23 EC19–20 and ICAM-1. We will focus first on standard steps for the analysis of CDH23 EC19–20 SMD simulations and will finish with a deformation analysis for ICAM-1.

The first step consists of uploading the NAMD output (LOG) file of the simulation. PySteMoDA extracts relevant data from it and converts units automatically. The force versus displacement (also called distance) for CDH23 EC19–20 simulations shows one clear main peak followed by a minor peak at both ends (SMD and TCL) of the protein fragment (Figure 2). Users can also plot and analyze different quantities such as temperature, pressure, and VDW energies. Other data can be exported, including force (pN), virtual spring constant (pN/nm), velocity (nm/ns), displacement/distance (nm) and time (s). PySteMoDA allows plotting and analysing both SMD and TCL forces when available by simply checking the "*See F(−v)*" button (Figure 2). Moreover, PySteMoDA allows the users to smooth the thermal noise by applying a natural cubic spline smoothing filter[48] that requires specification of a "knots" parameter. PySteMoDA provides a default value of "knots" to guide users (**see Methods**). Smoothing of the CDH23 EC19–20 SMD data after applying the natural cubic spline filter with the suggested default number of "knots" shows clearer force peaks (Figure 3A). These automated data extraction tools are a key benefit of PySteMoDA.

In addition, PySteMoDA facilitates the analysis of SMD trajectories by enabling detection of specific unfolding/unbinding peaks from the force curve. To optimize this process, default values of relevant parameters, such as the prominence threshold, are suggested and automatically filled within the interface as soon as data are uploaded. Peak detection using the prominence parameter helps to identify sharp peaks that should correspond to unfolding/unbinding events. Peaks with high prominence are considered distinct from the background noise. The prominence value suggested by PySteMoDA corresponds to the mean prominence of all the potential peaks in the curve (**See Methods**). Users can modify the default values of knots and prominence if not satisfied with the output. The peak detection algorithm identifies

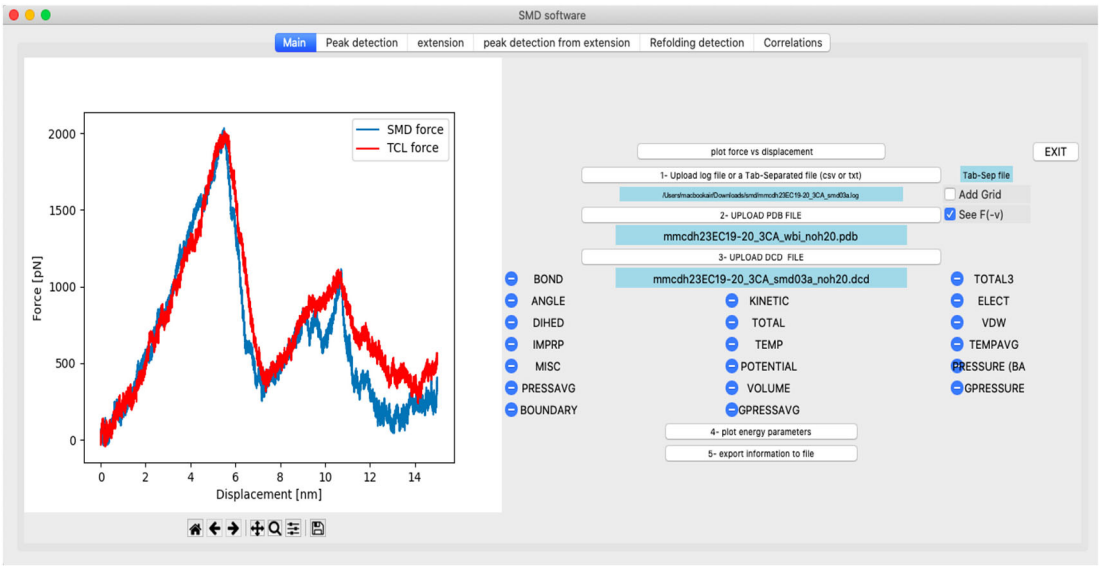


Figure 2. Snapshot of the PySteMoDA main tab. The force-displacement (also called force-distance) curve for the extracellular cadherin (EC) fragments 19 and 20 of cadherin-23 with three calcium ions (CDH23 EC19–20) is shown for a SMD simulation at 10 nm/ns. In this example, the protein was pulled from both ends, the TCL force curve is shown in red and the SMD force in blue. TCL forces can be visualized by checking the "See F(−v)" button.

8 unfolding events for the stretching of the CDH23 EC19–20 protein fragment using the default values of smoothing (knots) as well as the prominence value (Figure 3B). All the detected peaks are displayed in a table that records unfolding/unbinding forces (pN), time (s), loading rates (pN/s), stiffnesses (pN/nm) and molecular extensions (nm) (Figure 3B). Users can select the relevant peaks and export the information displayed within the table to different file formats (.csv, .dat, or .txt) (**supp. Table S1**).[45] We expect these PySteMoDA features to facilitate analyses of SMD trajectories.

Calculation of the loading rate at the main unfolding peak of the CDH23 EC19–20 molecule reveals a value of 2.5 x 10^{12} pN/s, as expected for a simulation done at a very fast stretching speed (10 nm/ns) using a stiff spring constant (Figure 4). The loading rate is obtained via a linear fit of the force-time curve and plotted at each peak within the GUI. The linear fit is calculated at a distance of 1 nm before each peak. This default distance was shown to work well on the example simulations but can be modified within the corresponding entry (Figure 4). Peak detection and corresponding analyses can also be done for TCL forces when these are applied during the SMD simulation **(supp. Figure S2).**

PySteMoDA can extract the extension curve corresponding to the Euclidean distance between the N- and C-terminal residues at each frame, which corresponds to the stretching of the molecule. When plotting the extension-time curve the unfolding events may appear as deviations from a linear increase in extension (Figure 5A). The extension curve of CDH23 EC19–20 in nm with respect to time in seconds shows two unfolding events as deviations from a linear increase in extension that resemble steps (Figure 5A). When these steps are sharp, they can be accurately detected from the derivative of the extension curve. This alternative peak detection approach from the time derivative of the extension is provided by PySteMoDA and available within the "*peak detection from extension*" tab. Within this tab, users can identify the unfolding events by selecting the number of highest peaks *N*. PySteMoDA identifies two unfolding events in the derivative of the extension profile of CDH23 EC19–20 (Figure 5B). These peaks can be selected by the user and represented in the force vs time curve (Figure 5C). Extension curves can also be calculated across a given range or ranges of amino acids (separated by commas, e.g. 1–89, 90–165 …), for example, to define the extension of each domain in a multidomain protein (**Figure S3**). We expect this extraction pipeline to accelerate analyses of similar SMD trajectories and to easily correlate force peaks with protein regions undergoing unfolding.

PySteMoDA can calculate a matrix of correlations to gain insight into how distances between residue pairs could determine changes in force during unfolding/unbinding processes. For this, the Pearson coefficient is calculated between the distances of each residue pair and the forces. This correlation matrix is represented as a heatmap within the graphical interface. The correlation heatmap for the CDH23 EC19–20 protein fragment shows positive correlation coefficients in red and negative coefficients in blue (Figure 6). The heatmap can be exported as a static image (png, jpeg and svg) or a

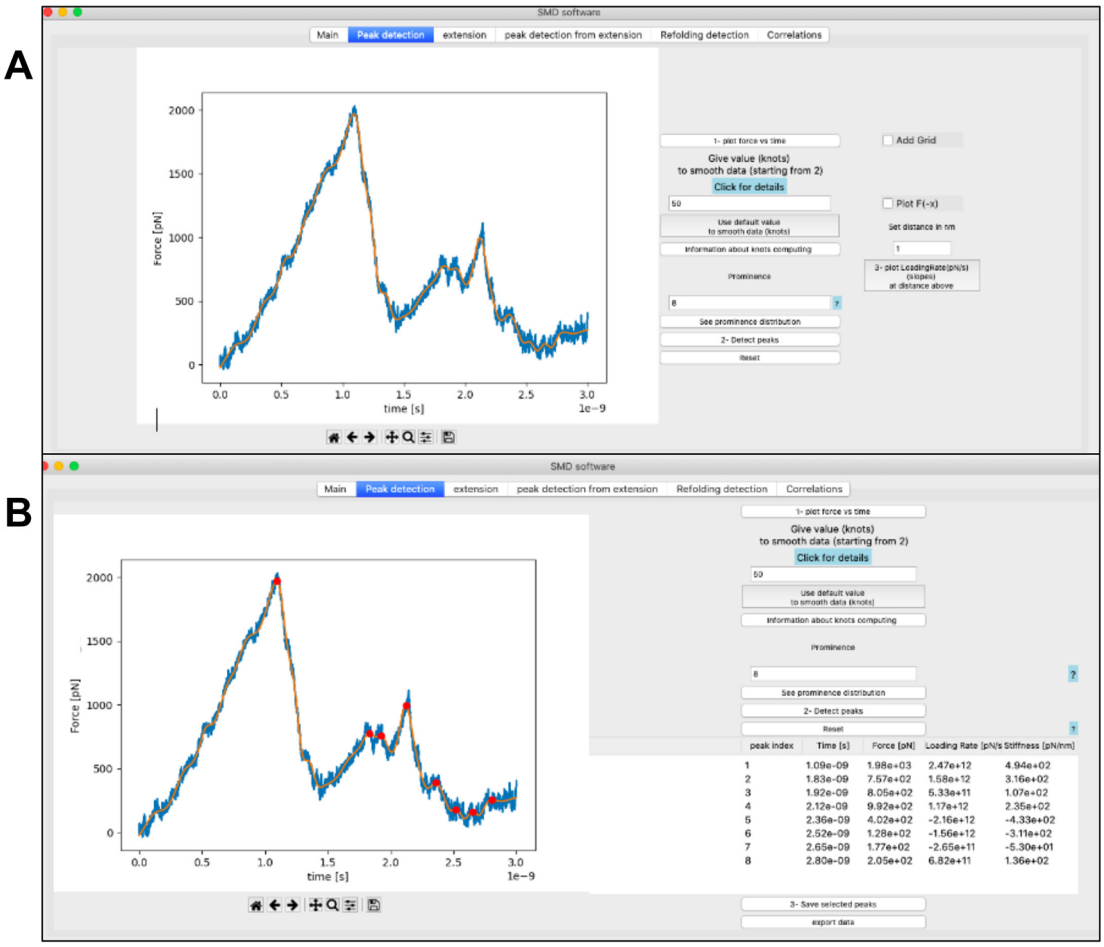


Figure 3. Data smoothing and peak detection. (A) Snapshot showing the natural cubic spline smoothing filter applied with default values to the force curve (orange, filtered; blue, raw). **(B)** Snapshot of the tab dedicated to peak detection from the force curve of the unfolding of CDH23 EC19–20 using the default parameters for smoothing (knots) and prominence. The list of the corresponding peaks is displayed within the table where the time (s), force (pN), loading rate (pN/s) and stiffness (pN/nm) associated to each peak are automatically calculated and displayed.

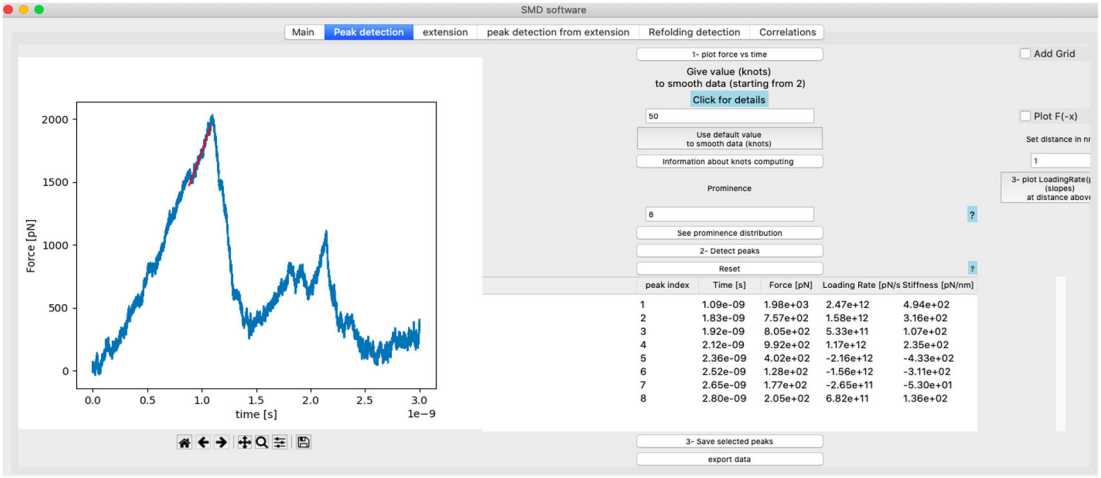


Figure 4. Loading rate calculation. Snapshot of loading rate (pN/s) plot displayed within the GUI after calculating the slope of the linear regression fit at the peak in the force versus time curve of CDH23 EC19–20. This plot appears when the user clicks the "plot loading rate" button while the loading rate (pN/s) and stiffness (pN/nm) values are calculated automatically and displayed within the main table shown within the GUI.

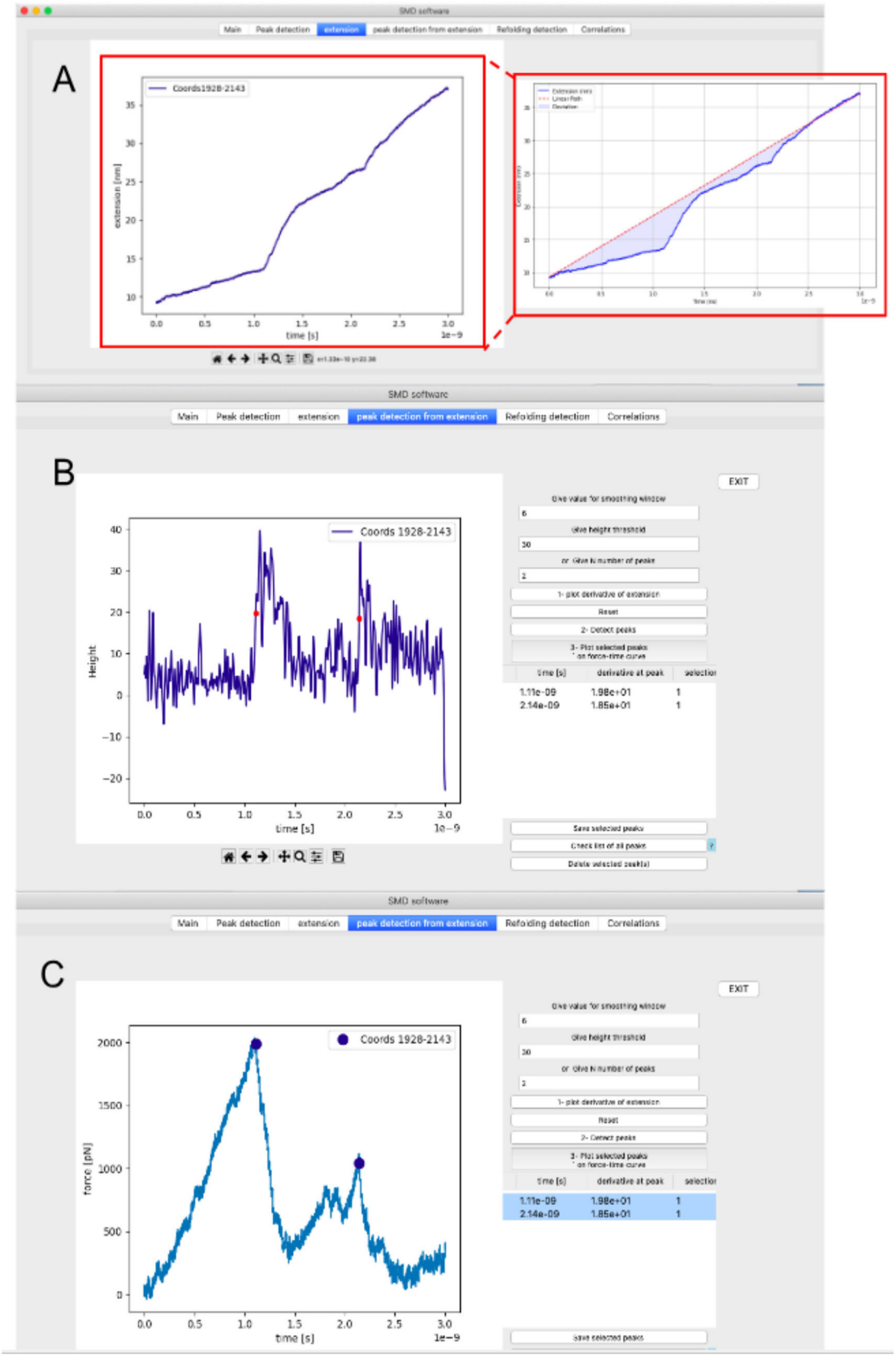


Figure 5. Snapshot of the *peak detection from extension* tab for CDH23 EC19–20. (A) Extension vs time curve. The unfolding peaks appear as increasing steps in extension. Inset shows linear change of position for the free end of the spring. **(B)** Derivative of the extension facilitates peak detection. A Gaussian filter has been applied to smooth the derivative of the data. The input parameters to detect the unfolding peaks are height and the number of highest peaks (*N*), which were set to 30 and 2 respectively. **(C)** The two peaks detected in **(B)** are represented by circles in the force versus time curve.

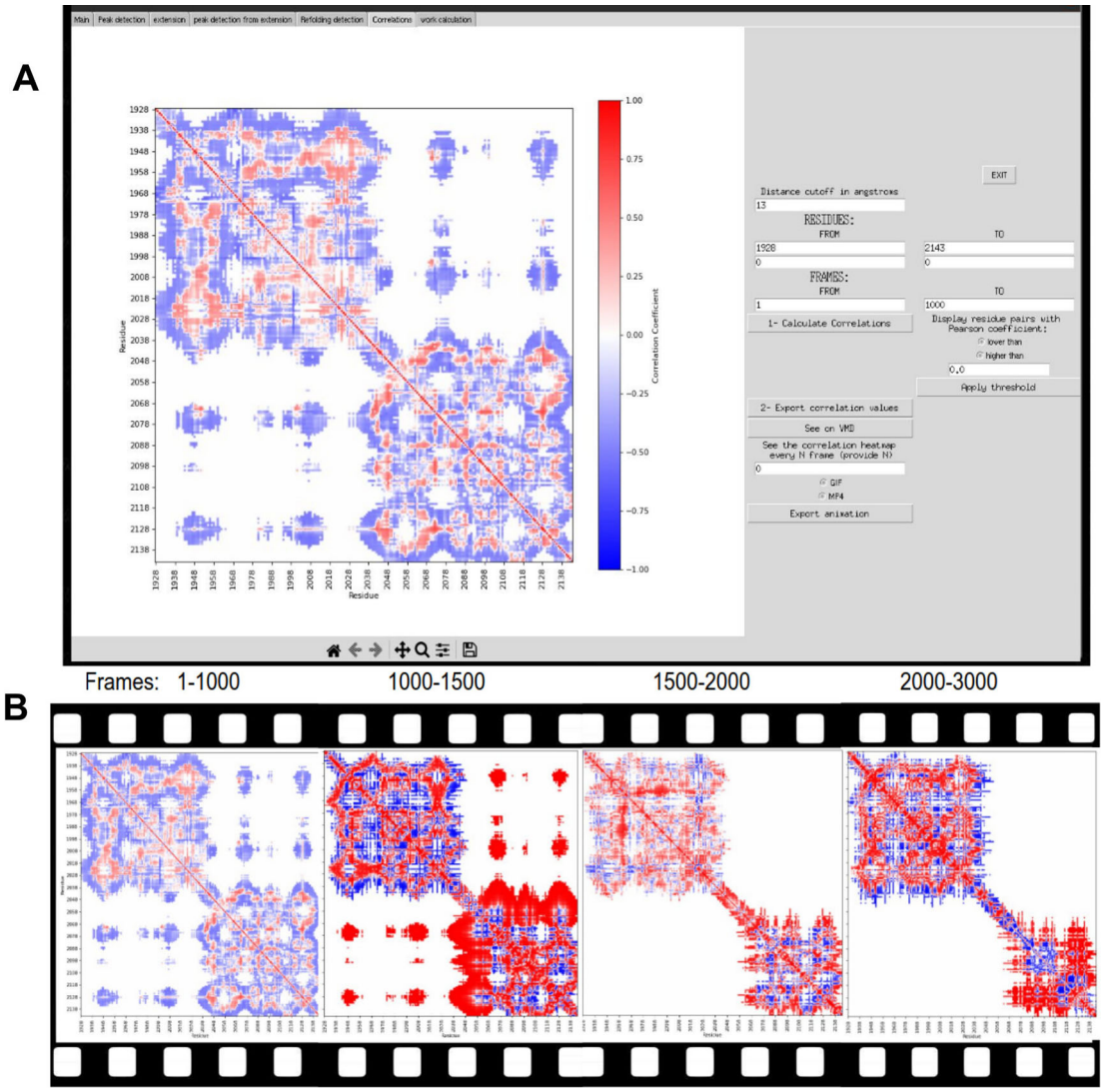


Figure 6. Pearson correlation computation. (A) Snapshot of the heatmap that shows the Pearson coefficients of correlations between force and all pairs of residues of CDH23 EC19–20 within a distance threshold of 13 Å (can be modified by the user) between the frames 1–1000. The correlation matrix within the interface is shown with the corresponding parameters. Note that the coefficient is equal to 0 when the color is white, negative values are blue, and positive values are red. **(B)** The correlation map along the unfolding trajectory. The first unfolding peak occurs after frame 1,000 (i.e., after 1 ns) where we see positive correlations (red) in EC20. This indicates that when the force is increasing the distance between pairs of residues is also increasing. In other words, the EC20 fragment is unfolding.

dynamic movie that shows the correlations every *N* trajectory step defined by the user. The file format in this case is either .gif or .mp4, both available as outputs within the GUI and the Colab notebook. The correlation coefficient values can also be exported as a matrix within a file (.txt, .csv, .dat). These coefficients of correlation are calculated along a range of time frames. By default, PySteMoDA uses the range from the first to last time frame, automatically extracted from the DCD trajectory file. This default range can be changed as needed, e.g., to select the frames around the major unfolding event. The residues taken by default in the correlation calculation are the first and last residue, their corresponding coordinates (*x, y, z*) are automatically extracted by PySteMoDA from the PDB file. The work (Joules) heatmap is also calculated along the indicated range of frames. An example of the work heatmap before and after the major unfolding peak shows that CDH23 EC20 has partially unfolded (Figure 7). The residue range can also be modified and can be provided in broken ranges, which may be useful for large protein structures with multiple domains. These analyses can be used to provide insight into which residues are most correlated with force rupture events.

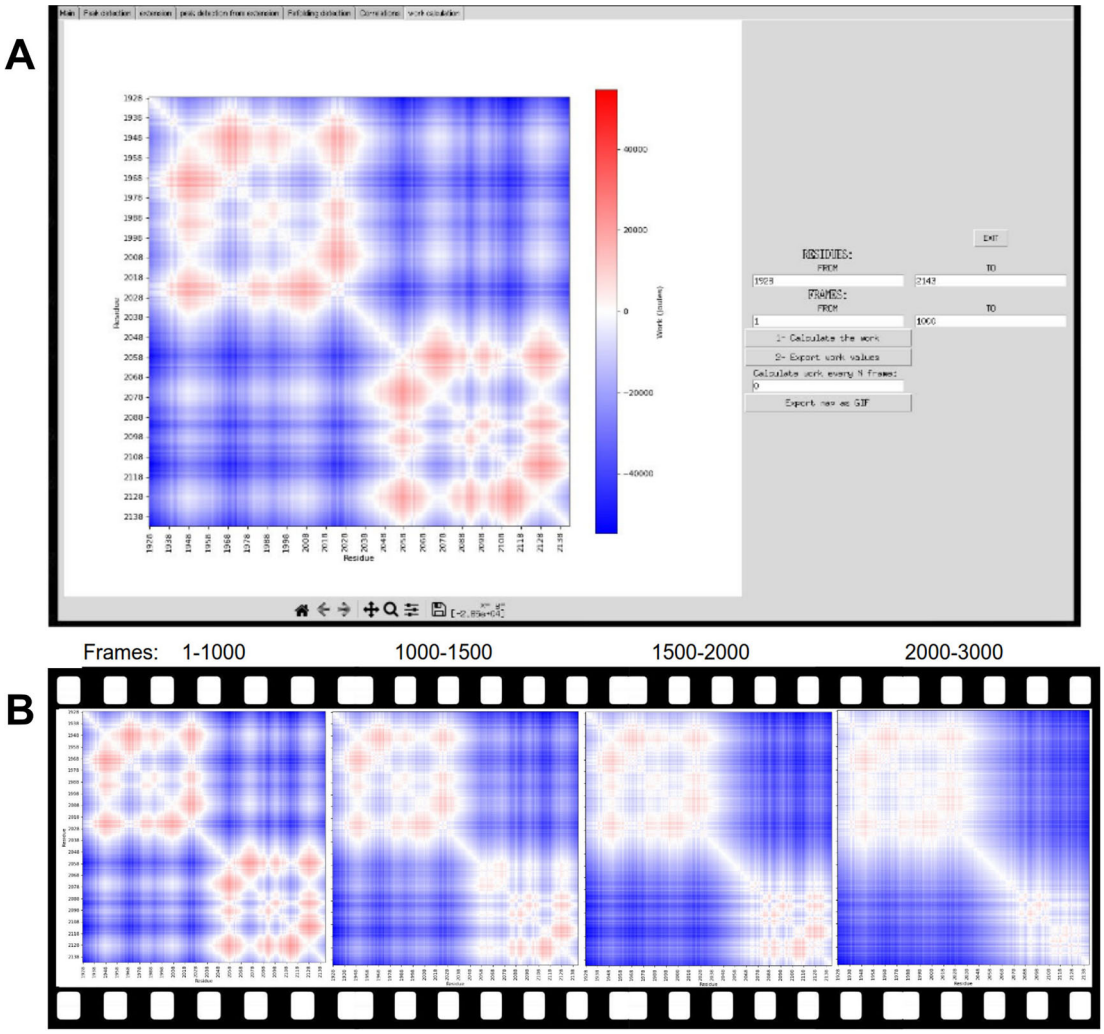


Figure 7. Heatmap of work for each pair of residues along the unfolding trajectory of CDH23 EC19–20. Negative work values in Joules appear in blue, positive values in red while the white color corresponds to zero. **(A)** snapshots of the heatmap of work within the interface with the corresponding parameters (i.e, frame ranges). **(B)** the heatmap of the work between frame 1 and 3,000 (i.e., from 0 to 3 ns). It is visible in the last map that EC20 has partially unfolded although the C-terminal end of the fragment is still resistant to force.

PySteMoDA can also be used to compute the inter-residue distance deformation, which was done here to characterize the mechanical response of the fourth domain of ICAM-1 (residues 282–364) during stretching in SMD simulations (Figure 8). The distance deformation matrix was calculated every 50 simulation frames, with frame intervals reported along the *y*-axis and residue numbers along the *x*-axis. Distinct clusters of residues undergoing unfolding events are observed as regions highlighted in light green and yellow at specific frame intervals. The temporal evolution of the extension of the fourth ICAM-1 domain (red) and the total applied force (blue) shows correlated changes over the same time scale. This enables direct association between residue-level unfolding events, domain extension, and force response.

Finally, the "see on VMD" button enables users to perform further analysis using VMD (versions 1.9 to 2.0)[14] by automatically loading the PDB and DCD files within the VMD interface installed within their local machine (**Figure S4**).

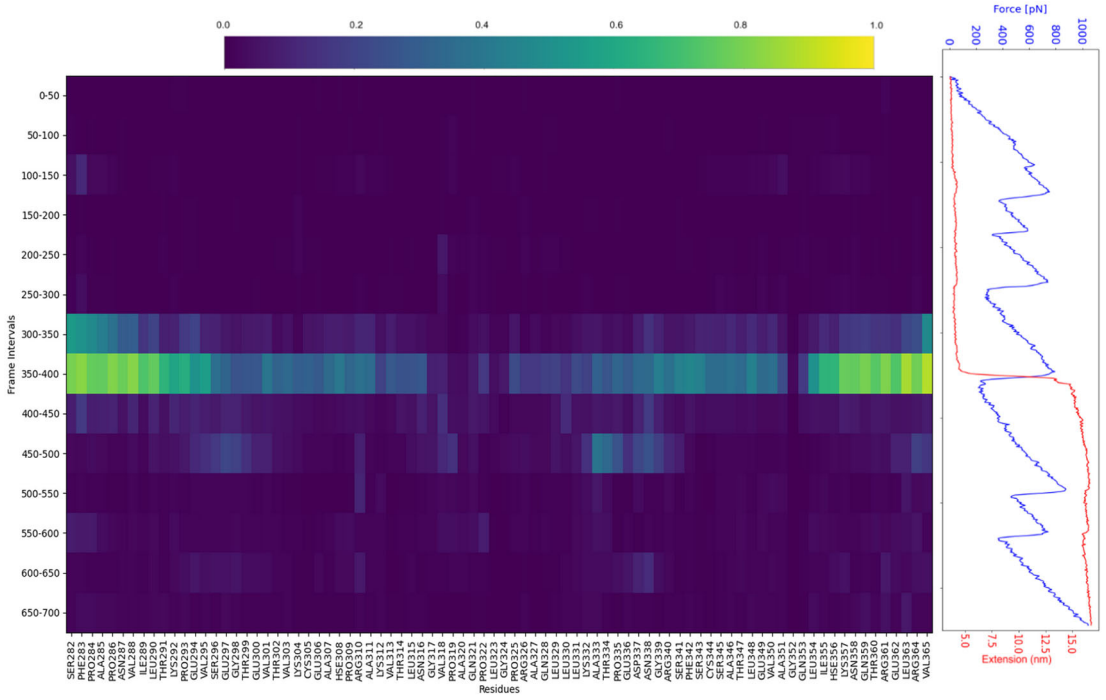


Figure 8. The inter-residue distance deformation measurement of the fourth domain of ICAM-1. The domain covers residues 282 to 364. The matrix is shown on the left. The calculation of the metric was performed every 50 frames. Frame intervals are reported in the *y*-axis of the matrix and the residue number is in the *x*-axis. Clusters of residues unfolding at each frame interval appear in light green and yellow. Extension of the fourth domain of ICAM-1 (red) and the total force (blue) as a function of time are shown on the right. All graphs have the same time scale (in frames).

Discussion

PySteMoDA serves as a valuable tool for the molecular dynamics community using SMD simulations with NAMD. Its primary objective is to facilitate and automate the processing and analysis of SMD simulation data. The software offers an intuitive and user-friendly interface designed to guide users at all proficiency levels, even those unfamiliar with Python. For those new to the software, a detailed tutorial is provided to ensure efficient utilization of its capabilities accessible at: https://gitlab.com/fm4b_lab/pystemoda

One notable feature of PySteMoDA is its capacity to integrate well with VMD by enabling automatic opening and data loading into it. This integration not only simplifies the workflow but also enhances the overall analytical process by providing a visual representation of the protein unfolding/unbinding trajectory.

Moreover, PySteMoDA is a semi-automatic package that offers default parameter settings that provides users with a preliminary idea of the order of magnitude for the default analysis parameters and helps understanding the characteristics of the data, without needing immediate adjustments. The software remains customizable which allows the users to modify the parameters to their specific requirements and preferences. A Google Colab (ipynb) notebook has been prepared to facilitate processing multiple simulation files, accessible via https://colab.research.google.com/drive/1Cc1IqlMnsJ-SDoE3CIiMhnLHVN5y61Dp?usp=sharing or available for download to run on a local machine as a Jupyter notebook. This Google Colab environment offers a range of advantages, including free access to a GPU runtime environment, enabling faster computations for data analysis. Furthermore, its compatibility with popular Python libraries, such as SciPy, Scikit-learn and ProDy, ensures a smooth setup for different tasks.

In general, smoothing force data facilitates the characterization of specific unfolding peaks in single-molecule SMD simulations. PySteMoDA offers two possibilities to smooth the data: natural cubic splines and Gaussian filters. In natural cubic splines, the knots parameter controls the flexibility of the spline and boundary transitions. More knots increase local control but risk overfitting by capturing noise. Properly set, knots enhance peak detection and reduce noise effectively. The Gaussian filter, on the other hand, is known for its effectiveness in signal processing.[49] It attenuates noise while preserving the essential features associated with unfolding/unbinding events. Both of these smoothing approaches help users in refining peak detection.

PySteMoDA also offers two different peak detection approaches for analysis of molecular stretching pathways, the specific unfolding/unbinding peaks can be detected either from the force or from the extension curve. This second approach can be used as a complement to the first one as it may be more accurate and efficient in the identification of small peaks in some cases (e.g., in multi-domain proteins, **Figure S3**). Peak detection from the extension curve is possible via the derivative where the unfolding/unbinding peaks are reflected in sharp changes that enable a more accurate peak identification. In addition, this extension peak detection approach also allows the detection of refolding events, obscured in the force trace. These appear as steps with negative slope within the extensions versus time profile **(Figure S3A**) and can be accurately detected and exported from the derivative of the extensions profile as it is already the case with unfolding peaks. These tools accelerate the analysis of SMD simulations.

The correlation and apparent work maps enable users to identify which residues play a significant role in the mechanical stability of the protein. Maps can be calculated across a sliding window along the trajectory, providing frame specific data of pairs of residues. Residues with high correlation values within the range of frames near the unfolding/unbinding event may be critical in maintaining structural integrity under stress. Additionally, correlated residues may be evolutionarily conserved. Therefore, studying these residues (e.g., through sequence alignment with orthologs) can provide insights into the evolutionary pressure that shaped the protein's structure and mechanical function.

The inter-residue distance deformation measurement during the trajectory is relevant to understand the dynamics of the stretched protein (Figure 8), as it provides insight into how the spatial relationships between residues evolve over time. For example, when the difference $\left(d_{ij}(0) - d_{ij}(t)\right)$ is larger than the characteristic length scale l_d= 13 Å, usually used to represent 3D structures in 2D contact maps, or l_d = 3.8 Å, which is the typical distance between two amino acids, it indicates a significant change in the distance between residues, leading to a large exponential decay and a smaller deformation metric. This suggests that the residues have undergone a significant conformational shift (i.e., structural rearrangements). In contrast, when the distance change is smaller than l_d, the exponential term will be closer to 1, and the deformation metric will be larger, indicating minimal change. The subtraction from 1 at the start makes the metric more intuitive. Therefore, the higher the deformation the closer the metric is to 1. This method has also been adapted to compute deformation in terms of work (in Joules), **see extended data.**[45] These calculations can be performed using the provided Google Colab notebook or via the command line, while the analysis of these spatial relationships gives insight into the conformational rearrangement of the molecule before and after unfolding events.

Conclusion

PySteMoDA finds application in computational protein dynamics where researchers are interested in protein unfolding and unbinding. Researchers can use PySteMoDA to extract and analyze forces, trajectories, and structural data from SMD simulations which enables them to gain deeper insights into the predicted behavior of biomolecules. PySteMoDA represents a valuable addition to the toolkit of computational biology and researchers involved in using SMD simulations. Its user-friendly Graphical User Interface (GUI), data handling capabilities, open-source nature and comprehensive documentation make it accessible to new and experienced users.

Data availability

Underlying data

Underlying data supporting the results of this study are available in Zenodo (https://doi.org/10.5281/zenodo.18593785) contains the following datasets[45]:

- Results from ICAM1 SMD simulations (PySteMoDA_Datasets/data_ICAM1), including NAMD output files (LOG and DCD) and the corresponding protein 3D structure (PDB).

- Results from mmCdh23EC19–20 SMD simulations (PySteMoDA_Datasets/DATA_mmCDH23_EC19–20), including NAMD output files (LOG and DCD) and the corresponding protein 3D structure (PDB).

Extended data

Extended data associated with this study are also available in Zenodo at (https://doi.org/10.5281/zenodo.18593785).[45]

No restrictions apply to the access or use of these data. All data are openly available under the terms of the Creative Commons Attribution 4.0 International license (CC BY 4.0).

Software availability

Software available from: https://pypi.org/project/PySteMoDA/0.1.0/ or https://colab.research.google.com/drive/1Cc1IqlMnsJ-SDoE3CIiMhnLHVN5y61Dp?usp=sharing[50]

Source code available from: https://gitlab.com/fm4b_lab/pystemoda

Archived software available from: https://doi.org/10.5281/zenodo.18669890

License: MIT

Acknowledgement

The authors acknowledge Dr. Jean-Michel Arbona for insightful discussions on the inter-residue deformation measurement. Some simulations were carried out at the Ohio Supercomputing Center. Centre de Calcul Intensif d'Aix-Marseille is acknowledged for granting access to its high performance computing resources.

References

1. Smith JC, Roux B: **Eppur si muove! The 2013 Nobel Prize in Chemistry.** *Struct Lond Engl 1993.* 2013 Dec 3; **21**(12): 2102–2105.
2. Dror RO, Dirks RM, Grossman JP, *et al.*: **Biomolecular simulation: a computational microscope for molecular biology.** *Annu. Rev. Biophys.* 2012; **41**: 429–452.
Publisher Full Text
3. Lee EH, Hsin J, Sotomayor M, *et al.*: **Discovery through the computational microscope.** *Struct Lond Engl 1993.* 2009 Oct 14; **17**(10): 1295–1306.
4. Daggett V, Levitt M: **Realistic simulations of native-protein dynamics in solution and beyond.** *Annu. Rev. Biophys. Biomol. Struct.* 1993; **22**: 353–380.
PubMed Abstract | Publisher Full Text
5. Adcock SA, McCammon JA: **Molecular dynamics: survey of methods for simulating the activity of proteins.** *Chem. Rev.* 2006 May; **106**(5): 1589–1615.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Karplus M, Petsko GA: **Molecular dynamics simulations in biology.** *Nature.* 1990 Oct; **347**(6294): 631–639.
Publisher Full Text
7. Vanommeslaeghe K, Hatcher E, Acharya C, *et al.*: **CHARMM General Force Field (CGenFF): A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields.** *J. Comput. Chem.* 2010 Mar; **31**(4): 671–690.
8. Hwang W, Austin SL, Blondel A, *et al.*: **CHARMM at 45: Enhancements in Accessibility, Functionality, and Speed.** *J. Phys. Chem. B.* 2024 Oct 17; **128**(41): 9976–10042.
PubMed Abstract | Publisher Full Text | Free Full Text
9. Eastman P, Galvelis R, Peláez RP, *et al.*: **OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials.** *J. Phys. Chem. B.* 2024 Jan 11; **128**(1): 109–116.
PubMed Abstract | Publisher Full Text | Free Full Text
10. Case DA, Cheatham TE III, Darden T, *et al.*: **The Amber biomolecular simulation programs.** *J. Comput. Chem.* 2005; **26**(16): 1668–1688.
PubMed Abstract | Publisher Full Text | Free Full Text
11. Van Der Spoel D, Lindahl E, Hess B, *et al.*: **GROMACS: fast, flexible, and free.** *J. Comput. Chem.* 2005 Dec; **26**(16): 1701–1718.
PubMed Abstract | Publisher Full Text
12. Abraham MJ, Murtola T, Schulz R, *et al.*: **GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers.** *SoftwareX.* 2015 Sep 1; **1–2**: 19–25.
Publisher Full Text
13. Phillips JC, Braun R, Wang W, *et al.*: **Scalable Molecular Dynamics with NAMD.** *J. Comput. Chem.* 2005 Dec; **26**(16): 1781–1802.
PubMed Abstract | Publisher Full Text | Free Full Text
14. Humphrey W, Dalke A, Schulten K: **VMD: visual molecular dynamics.** *J. Mol. Graph.* 1996 Feb; **14**(1): 33–8. 27–8.
15. Isralewitz B, Gao M, Schulten K: **Steered molecular dynamics and mechanical functions of proteins.** *Curr. Opin. Struct. Biol.* 2001 Apr 1; **11**(2): 224–230.
Publisher Full Text
16. Franz F, Daday C, Gräter F: **Advances in molecular simulations of protein mechanical properties and function.** *Curr. Opin. Struct. Biol.* 2020 Apr 1; **61**: 132–138.
Publisher Full Text
17. Sotomayor M, Schulten K: **Single-molecule experiments in vitro and in silico.** *Science.* 2007 May 25; **316**(5828): 1144–1148.
PubMed Abstract | Publisher Full Text
18. Grubmüller H: **Force probe molecular dynamics simulations.** *Methods Mol Biol Clifton NJ.* 2005; **305**: 493–515.
19. Grubmüller H, Heymann B, Tavan P: **Ligand binding: molecular mechanics calculation of the streptavidin-biotin rupture force.** *Science.* 1996 Feb 16; **271**(5251): 997–999.
PubMed Abstract | Publisher Full Text
20. Lu H, Isralewitz B, Krammer A, *et al.*: **Unfolding of Titin Immunoglobulin Domains by Steered Molecular Dynamics Simulation.** *Biophys. J.* 1998 Aug 1; **75**(2): 662–671.
PubMed Abstract | Publisher Full Text | Free Full Text
21. Park S, Schulten K: **Calculating potentials of mean force from steered molecular dynamics simulations.** *J. Chem. Phys.* 2004 Apr 1; **120**(13): 5946–5961.
Publisher Full Text
22. Izrailev S, Stepaniants S, Balsera M, *et al.*: **Molecular dynamics study of unbinding of the avidin-biotin complex.** *Biophys. J.* 1997 Apr; **72**(4): 1568–1581.
PubMed Abstract | Publisher Full Text | Free Full Text
23. Rico F, Russek A, González L, *et al.*: **Heterogeneous and rate-dependent streptavidin–biotin unbinding revealed by high-speed force spectroscopy and atomistic simulations.** *Proc. Natl. Acad. Sci.* 2019 Apr 2; **116**(14): 6594–6601.
PubMed Abstract | Publisher Full Text | Free Full Text
24. Milles LF, Gaub HE: **Extreme mechanical stability in protein complexes.** *Curr. Opin. Struct. Biol.* 2020 Feb; **60**: 124–130.
Publisher Full Text
25. Lu H, Schulten K: **The Key Event in Force-Induced Unfolding of Titin's Immunoglobulin Domains.** *Biophys. J.* 2000 Jul 1; **79**(1): 51–65.
PubMed Abstract | Publisher Full Text | Free Full Text
26. Evans E, Ritchie K: **Dynamic strength of molecular adhesion bonds.** *Biophys. J.* 1997 Apr; **72**(4): 1541–1555.
PubMed Abstract | Publisher Full Text | Free Full Text
27. Marszalek PE, Lu H, Li H, *et al.*: **Mechanical unfolding intermediates in titin modules.** *Nature.* 1999 Nov 4; **402**(6757): 100–103.
PubMed Abstract | Publisher Full Text
28. Rico F, Gonzalez L, Casuso I, *et al.*: **High-speed force spectroscopy unfolds titin at the velocity of molecular dynamics simulations.** *Science.* 2013 Nov 8; **342**(6159): 741–743.
PubMed Abstract | Publisher Full Text
29. Nisler CR, Narui Y, Scheib E, *et al.*: **Interpreting the Evolutionary Echoes of a Protein Complex Essential for Inner-Ear Mechanosensation.** *Mol. Biol. Evol.* 2023 Apr 4; **40**(4): msad057.
30. Corey DP, Hudspeth AJ: **Kinetics of the receptor current in bullfrog saccular hair cells.** *J. Neurosci.* 1983 May; **3**(5): 962–976.
PubMed Abstract | Publisher Full Text | Free Full Text
31. Upadhyaya A, Baraban M, Wong J, *et al.*: **Power-limited contraction dynamics of Vorticella convallaria: an ultrafast biological spring.** *Biophys. J.* 2008 Jan 1; **94**(1): 265–272.
PubMed Abstract | Publisher Full Text | Free Full Text
32. Jaiganesh A, Narui Y, Araya-Secchi R, *et al.*: **Beyond Cell-Cell Adhesion: Sensational Cadherins for Hearing and Balance.** *Cold*

Spring Harb. Perspect. Biol. 2018 Sep 4; **10**(9): a029280.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

33. Robles L, Ruggero MA: **Mechanics of the mammalian cochlea.** *Physiol. Rev.* 2001 Jul; **81**(3): 1305–1352.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

34. Freddolino PL, Liu F, Gruebele M, *et al.*: **Ten-microsecond molecular dynamics simulation of a fast-folding WW domain.** *Biophys. J.* 2008 May 15; **94**(10): L75–L77.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

35. Sotomayor M: **Computational exploration of single-protein mechanics by steered molecular dynamics.** *AIP Conf Proc.* 2015 Dec 31; **1703**(1): 030001.

36. Costescu BI, Sturm S, Gräter F: **Dynamic disorder can explain non-exponential kinetics of fast protein mechanical unfolding.** *J. Struct. Biol.* 2017 Jan 1; **197**(1): 43–49.
PubMed Abstract | **Publisher Full Text**

37. Orzeł U, Pasznik P, Miszta P, *et al.*: **GS-SMD server for steered molecular dynamics of peptide substrates in the active site of the γ-secretase complex.** *Nucleic Acids Res.* 2023 Jul 5; **51**(W1): W251–W262.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

38. Bernardi RC, Durner E, Schoeler C, *et al.*: **Mechanisms of Nanonewton Mechanostability in a Protein Complex Revealed by Molecular Dynamics Simulations and Single-Molecule Force Spectroscopy.** *J. Am. Chem. Soc.* 2019 Sep 18; **141**(37): 14752–14763.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

39. Bezerra Beniz D, Espíndola A: **USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS.** 2016.

40. Bakan A, Meireles LM, Bahar I: **ProDy: Protein Dynamics Inferred from Theory and Experiments.** *Bioinformatics.* 2011 Jun 1; **27**(11): 1575–1577.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

41. Pedregosa F, Varoquaux G, Gramfort A, *et al.*: **Scikit-learn: Machine Learning in Python.** *J. Mach. Learn. Res.* 2011 Nov 1; **12**: 2825–2830.

42. Virtanen P, Gommers R, Oliphant TE, *et al.*: **SciPy 1.0: fundamental algorithms for scientific computing in Python.** *Nat. Methods.* 2020 Mar; **17**(3): 261–272.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

43. Jaiganesh A, De-la-Torre P, Patel AA, *et al.*: **Zooming in on cadherin-23: Structural diversity and potential mechanisms of inherited deafness.** *Struct Lond Engl 1993.* 2018 Sep 4; **26**(9): 1210–1225.e4.
Publisher Full Text

44. Sotomayor M, Weihofen WA, Gaudet R, *et al.*: **Structural Determinants of Cadherin-23 Function in Hearing and Deafness.** *Neuron.* 2010 Apr 15; **66**(1): 85–100.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

45. Mesbah I, Klaus C, Sotomayor M, *et al.*: **PySteMoDA Datasets.** *Zenodo.* 2026 [cited 2026 Feb 11].
Reference Source

46. Xiao C, Bator-Kelly CM, Rieder E, *et al.*: **The crystal structure of coxsackievirus A21 and its interaction with ICAM-1.** *Struct Lond Engl 1993.* 2005 Jul; **13**(7): 1019–1033.
Publisher Full Text

47. Chen X, Kim TD, Carman CV, *et al.*: **Structural plasticity in Ig superfamily domain 4 of ICAM-1 mediates cell surface dimerization.** *Proc. Natl. Acad. Sci.* 2007 Sep 25; **104**(39): 15358–15363.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

48. Feng G: **Data smoothing by cubic spline filters.** *Signal Process IEEE Trans On.* 1998 Nov 1; **46**: 2790–2796.
Publisher Full Text

49. Deisenroth M, Ohlsson H: **A general perspective on Gaussian filtering and smoothing: Explaining current and deriving new algorithms.** 2011; 1807.

50. Mesbah I, Klaus C, Sotomayor M, *et al.*: **PySteMoDA: An Open Source Python Package for the Analysis of Steered Molecular Dynamics Simulations Data.** 2026 Jan 15 [cited 2026 Feb 11].
Reference Source

Open Peer Review

Current Peer Review Status: ? ?

Version 1

Reviewer Report 06 May 2026

https://doi.org/10.21956/openreseurope.24743.r72020

© **2026 Delgado-Buscalioni R.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Rafael Delgado-Buscalioni
Universidad Autonoma de Madrid, Campus de Cantoblanco,, Madrid 28049,, Spain

This tool can become quite useful, and I believe it will be used by the community.
Ideally, the tool should be alive, and open to adapt new analyses.
The presentation is clear, although some parts deserve clarification:

A)
For instance, the apparent work is calculated for each pair of residues, I understand.
Hence Delta r
should read Delta r_ij and F should read F_ij.
What is Fij = F_j - F_j ? please clarify these eqs.

B)
Please define k_molecular k_effective and k_virtual
using quantities already defined along the paper.
I understand, k_molecular is just the result. So , please clearly define k_eff and k_virtual

C)
5. Inter-residues distance deformation
Provide a citation on some paper where these equations for deformation are defined and justified.
Define nb in the equation (number of bonds??)
Similarly, for the work parameter.

D) The authors might consider possible extensions to their code:
Jarzinsky equation into the set of tools
Would this be possible/appropiate?

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Stat.Physics and simulation of soft matter, fluids and biomolecules

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 16 April 2026

https://doi.org/10.21956/openreseurope.24743.r71762

© 2026 Poma A et al. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Gustavo E. Olivos-Ramirez
Instytut Podstawowych Problemow Techniki Polskiej Akademii Nauk (Ringgold ID: 86911), Warsaw, Masovian Voivodeship, Poland

Adolfo Poma
Polish Academy of Sciences,, ul. Pawińskiego 5B,, Warsaw 02-106,, Poland

In this study Mesbah et al. present a software package for interpretation of force-distance profiles commonly used in nanomechanical characterization via molecular dynamics simulation. The tool is code in python and has some versatility in analysis but I recommend to follow these recommendation to improve the clarify of the manuscript and tool description.

Major concerns:
1) It is mentioned in paragraphs 4 and 5 of the Introduction that the developed package can extract and analyze Steered Molecular Dynamics (SMD) data in a semi-automatic manner, which does not fully agree with what is stated in the abstract. It might be better to avoid using the word “semi”, or alternatively to explain more clearly to what extent the user can control the automation of the process, for example by adjusting specific parameters.

2) In paragraph 5 of the Introduction, the authors mention that the package can analyze SMD output data from the MD package NAMD. Currently, there are several trajectory formats in MD simulations such a DCD, XTC, NC, etc.. It would therefore be useful to clarify whether the python package can also be used with other trajectory formats. This would increase the potential reach of the package.

3) The idea of providing a Google Colab version is very appealing. However, computational resources and storage space are usually limited. The authors could comment on whether SMD outputs from small or large protein systems can be analyzed in Colab, or alternatively provide an approximate size of systems (e.g. residue length, trajectory frames, number of replicas) that can reasonably be analyzed in that environment. In addition, as mentioned many
replicas are required for statistical analysis, increasing the computational resources which should be indicated.

4) In Materials and Methods, Section 2 (Simulations), it would be convenient to specify the reason behind the use of the NVE
ensemble, as in more routine studies the NVT/NPT ensemble is conventionally chosen.

5) In Section 3 (Data extraction and conversion), the need to provide a log file is characteristic of NAMD simulations. Does this restrict the software to NAMD users? If so, what other modifications are necessary to be more general across other MD packages. Certainly GROMACS's user can appreciate it.

6) In Section 5 (“Inter-residues distance deformation”), the authors could indicate which atom or group of atoms are selected for this calculation: Cα atom selection, or the whole residue.

7) Additional to the inter-residues distance deformation in section 5, the author can look at analysis of key native contacts that are presence under native conditions (zero mechanical force) and start to stretch during the nanomechanical study. They can help to infer interactions associated with the pattern deformation. Authors can look at the present review (https://doi.org/10.12693/APhysPolA.145.S9) for a broad discussion on protein contact determination criteria widely use in nanomechanical interpretation of protein complexes. Each contact pairs is associated with "distance" that change over the nanomechanical profiles.

8) In the third paragraph of the Results section, it is indicated that some parameters can be changed by
the users. Please provide examples of those parameters that can be tuned, and under which specific circumstances such adjustments would be useful in the analysis.

9) The tool has a default parameter that produces plots in seconds (X axis), and unit conversion E-9 in the far right side. I guess the unit should be given in ns or ps, and user can adjust accordingly. To see second is misleading as typical MD scales are several orders of magnitude smaller.

10) The idea of correlating residues with force is very interesting. The authors are advised to provide a more detailed discussion of the systems tested and of the interpretation of the correlation coefficients obtained for the pulling examples presented in this study.

11) In the Conclusion, it is highly recommended to add concluding remarks about the evaluated

system, whether the software has provide any new insight about it.

12) Although the software is adapted to atomistic data, it is worth mentioning that nowadays coarse-grained MD simulation has been established as advanced way to reach large conformational changes, low loading rates (2-4 order of magnitude smaller than atomistic ones) and study of large scale system. So the versatility of the current software can be also directed to such community. A comment will be appreciated.

Minor revision:
- In Materials and Methods, Section 1 (Implementation), in the second line, a comma is missing after scikit-learn 41.
- On page 6, the word Cartesian should be capitalized, and the i in “ith” should be italicized.
- The title of Figure 2 is a bit misleading. I think authors can consider: Graphical interface of the PySteMoDA software. Moreover, instead of explaining the results, it would be more convenient to provide details about the graphical interface, since this figure is intended as a software presentation.
- In the last sentence on page 9, a comma is missing in: (png, jpeg, and svg).
- Figure 5 has a very low resolution compared with the other figures. Please replace it.

References
1. Cofas-Vargas L, Moreira R, Poblete S, Chwastyk M, et al.: The GōMartini Approach: Revisiting the Concept of Contact Maps and the Modelling of Protein Complexes. *Acta Physica Polonica A*. 2024; **145** (3): S9-S20 Publisher Full Text

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Computational biophysics and multiscale modelling

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.