Check for updates

SOFTWARE TOOL ARTICLE

REVISED PyFMLab: Open-source software for atomic force microscopy microrheology data analysis

[version 2; peer review: 2 approved]

Javier López-Alonso[1], Mar Eroles[2], Sébastien Janel[1], Massimiliano Berardi[3,4], Jean-Luc Pellequer[5], Vincent Dupres[1], Frank Lafont[1], Felix Rico[2]

[1]Universite de Lille, CNRS, INSERM, CHU Lille, Institut Pasteur de Lille, U1019-UMR9017, CILL—Center of Infection and Immunity of Lille, Lille, F-59000, France
[2]Aix-Marseille Univ., CNRS, INSERM, LAI, Turing Centre for Living Systems, Marseille, 13009, France
[3]LaserLab, Department of Physics and Astronomy, Vrije Universiteit Amsterdam, Amsterdam, 1081HV, The Netherlands
[4]Optics 11 B.V, Amsterdam, 1101BM, The Netherlands
[5]Univ. Grenoble Alpes, CEA, CNRS, IBS, Grenoble, F-38000, France

v2 **First published:** 30 Oct 2023, **3**:187
https://doi.org/10.12688/openreseurope.16550.1
Latest published: 24 Jul 2024, **3**:187
https://doi.org/10.12688/openreseurope.16550.2

Abstract

Background

Atomic force microscopy (AFM) is one of the main techniques used to characterize the mechanical properties of soft biological samples and biomaterials at the nanoscale. Despite efforts made by the AFM community to promote open-source data analysis tools, standardization continues to be a significant concern in a field that requires common analysis procedures. AFM-based mechanical measurements involve applying a controlled force to the sample and measure the resulting deformation in the so-called force-distance curves. These may include simple approach and retract or oscillatory cycles at various frequencies (microrheology). To extract quantitative parameters, such as the elastic modulus, from these measurements, AFM measurements are processed using data analysis software. Although open tools exist and allow obtaining the mechanical properties of the sample, most of them only include standard elastic models and do not allow the processing of microrheology data. In this work, we have developed an open-source software package (called PyFMLab, as of python force microscopy laboratory) capable of determining the viscoelastic properties of samples from both conventional force-distance curves and microrheology measurements.

Open Peer Review

Approval Status ✓ ✓

	1	2
version 2 (revision) 24 Jul 2024	✓ view	✓ view
version 1 30 Oct 2023	? view	? view

1. **Mi Li**, Shenyang Institute of Automation Chinese Academy of Sciences, Beijing, China
2. **Stylianos Vasileios Kontomaris**, Metropolitan College, Athens, Greece

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

Methods

PyFMLab has been written in Python, which provides an accessible syntax and sufficient computational efficiency. The software features were divided into separate, self-contained libraries to enhance code organization and modularity and to improve readability, maintainability, testability, and reusability. To validate PyFMLab, two AFM datasets, one composed of simple force curves and another including oscillatory measurements, were collected on HeLa cells.

Results

The viscoelastic parameters obtained on the two datasets analysed using PyFMLab were validated against data processing proprietary software and against validated MATLAB routines developed before obtaining equivalent results.

Conclusions

Its open-source nature and versatility makes PyFMLab an open-source solution that paves the way for standardized viscoelastic characterization of biological samples from both force-distance curves and microrheology measurements.

Plain language summary

Just like we can test the ripeness of fruit by touching it, we can use our hands to gently touch an object and determine if it is soft or stiff. Doctors use this technique, called palpation, to explore our organs and check for signs of disease. We can think about doing something similar, but on a much smaller scale — the nanoscale — so small that you cannot even see it with your naked eye. Atomic force microscopy (AFM) allows to apply palpation at the nanoscale.

AFM is a powerful tool that allows scientists to examine incredibly small objects, like individual cells or molecules. AFM uses super sensitive “fingers” to touch and explore things that are too small to be seen under a regular microscope. During the European project Phys2BioMed, we explored how to apply AFM to diagnose diseases using nanopalpation. For example, touching samples from biopsies of patients and determining how soft or stiff they are.

Here's the catch: there is not a single, standardized method or software that can efficiently process all the data obtained from AFM. It's a bit like having a lot of different languages but no universal translator. Just like a scale or a measuring cup are standardized, scientists need to analyze AFM data accurately and consistently. This is crucial to ensure reliable comparisons between results obtained by different researchers on different instruments, something particularly important when the results are to be used for diagnostic or predictive purposes.

To help tackle this problem, we have developed PyFMLab. This software is a reliable and easy-to-use tool that translates AFM data into insights about the tiny structures being studied. By providing a standardized, open-source, modular and accessible way to analyze AFM data, PyFMLab democratizes access to this field of Biophysics, paving the way for clinical applications of AFM.

Keywords
force spectroscopy, Elasticity, Viscoelasticity, Young's modulus, Biological Samples, cell mechanics, tissue mechanics, soft matter

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

This article is included in the Horizon 2020 gateway.

This article is included in the Software gateway.

This article is included in the Marie-Sklodowska-Curie Actions (MSCA) gateway.

Corresponding authors: Javier López-Alonso (javierlopezal.jla@gmail.com), Felix Rico (felix.rico@inserm.fr)

Author roles: López-Alonso J: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Eroles M**: Data Curation, Methodology, Resources, Validation; **Janel S**: Conceptualization, Methodology, Validation; **Berardi M**: Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Pellequer JL**: Funding Acquisition, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Dupres V**: Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; **Lafont F**: Funding Acquisition, Resources, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; **Rico F**: Conceptualization, Formal Analysis, Funding Acquisition, Methodology, Project Administration, Software, Supervision, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This project has received funding by the H2020 European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie (grant agreement No 812772) to JLP, FL and FR and from the European Research Council (ERC, grant agreement No 772257) to FR.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2024 López-Alonso J *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: López-Alonso J, Eroles M, Janel S *et al.* **PyFMLab: Open-source software for atomic force microscopy microrheology data analysis [version 2; peer review: 2 approved]** Open Research Europe 2024, **3**:187
https://doi.org/10.12688/openreseurope.16550.2

First published: 30 Oct 2023, **3**:187 https://doi.org/10.12688/openreseurope.16550.1

REVISED **Amendments from Version 1**

In this version we addressed the comments from the reviewers. We updated Figure 3 to represent a more realistic force-distance curve, without being shifted to 0, and included a new Appendix showcasing different tutorials on how users can expand the libraries developed in this work.

Any further responses from the reviewers can be found at the end of the article

Introduction

In recent years, the study of cellular and extracellular matrix (ECM) mechanics has increased considerably due to the realization of its fundamental role in several physiological and pathological processes like cell division, migration, differentiation and malignancy (Engler *et al.*, 2006; Rianna *et al.*, 2020). Amongst other techniques, atomic force microscopy (AFM) has proven to be a powerful tool to characterize the mechanical properties of living cells and ECM (Lekka *et al.*, 2023a; Lekka *et al.*, 2023b) and it is considered as a standard tool in the field. Despite its popularity, standardization of sample preparation, measurement and data analysis protocols are still lacking.

To overcome some of these limitations, several efforts have been initiated towards AFM probe calibration procedures (Sader *et al.*, 2016; Schillers *et al.*, 2017), and open source or free AFM data analysis tools like Gwyddion (Necas & Klapetek, 2012), TopoStats (Beton *et al.*, 2021), AtomicJ (Hermanowicz *et al.*, 2014), PyJibe (Müller *et al.*, 2019), ViscoIndent (Efremov *et al.*, 2020) and Rheos (Kaplan *et al.*, 2019).

The analysis of AFM force measurements is generally carried out using either home-made (Carl & Schillers, 2008; Chen *et al.*, 2022; Domke & Radmacher, 1998; Kontomaris *et al.*, 2023; Lekka *et al.*, 1999) or instrument-associated commercial software. Although these tools allow to extract the topographical and mechanical properties of the sample, most of these packages are either not open source, limited to the most standard models, not compatible with other manufacturers data formats, do not allow obtaining viscoelastic parameters or do not support microrheology data.

The most common approach to determine the elasticity of soft biological samples is the acquisition of AFM force-distance curves (FDCs), to then fit an elastic contact model, usually to the approach segment of the curve, and obtain the apparent Young's modulus (E). For determining the viscoelasticity of the sample, two main approaches are possible:

- FDC (time-domain method): where a viscoelastic model is fitted to the full FDC (approach and retract) (Brückner *et al.*, 2017; Efremov *et al.*, 2020; Sanchez *et al.*, 2021) or to certain segments (approach, pause, steps (Yango *et al.*, 2016)).
- Microrheology (frequency-domain method): consisting in the force response analysis of small amplitude z oscillations at different frequencies to determine the complex shear modulus (Alcaraz *et al.*, 2003).

Most available AFM open-source and commercial software tools allow to compute E from FDCs but not viscoelastic parameters. Amongst open-source tools, only ViscoIndent natively supports rheology data, but even that is limited to time domain analysis, which requires a priori knowledge regarding the constitutive behavior of the sample under test. A turn-key solution that encompasses both time and frequency domain viscoelastic analysis, as well as microrheology related signal correction procedures is still missing. Making this type of analysis more accessible is important, as recent works have shown the time dependent response of ECM plays a crucial role in tissue organization (Elosegui-Artola *et al.*, 2023). PyFMLab has been developed with these needs in mind.

Software architecture

The software functionalities were split into different self-contained libraries to help organize the code while making it more readable, maintainable, testable and reusable. Similarly to the AFM data analysis PyJibe developed by Paul Muller (Müller *et al.*, 2019), the functionalities of PyFMLab are split on three main Python libraries organized in a hierarchical manner (Figure 1):

PyFMReader

The source code for PyFMReader is available (from zenodo López-Alonso, 2023a) (also available from GitHub)

This library includes methods to load and pre-process data from different AFM manufacturer files, making FDCs, topography data and metadata accessible to the user. At the core of this library, is the universal file format (UFF) object (manuscript in preparation). The files from different instruments are parsed and the metadata, force curve and image data are loaded into the UFF object.

PyFMReader can be used as a stand-alone library to load data acquired by Bruker® instruments running JPK SPM software v4.2 – v7.0 (.jpk-force, .jpk-force-map, .jpk-qi-data) or Nanoscope software v7.0 – v9.0 (.spm, .pfc). PyFMReader allows exporting to UFF txt data files. Interactive Jupyter notebooks are provided with examples on how to load files and access loaded data.

To allow flexibility in implementing functionalities, each file (map or single force curve) is stored within a UFF object. This implementation results in a hierarchical data organization (Figure 2) where UFF objects point to the different force curve objects and each force curve object points to distinct

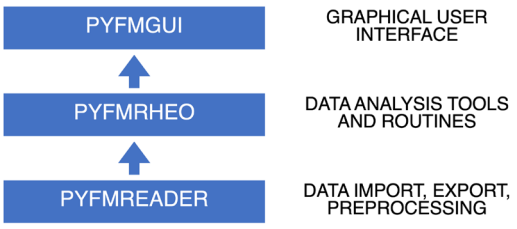

Figure 1. Diagram representing how the libraries composing PyFMLab are organized.

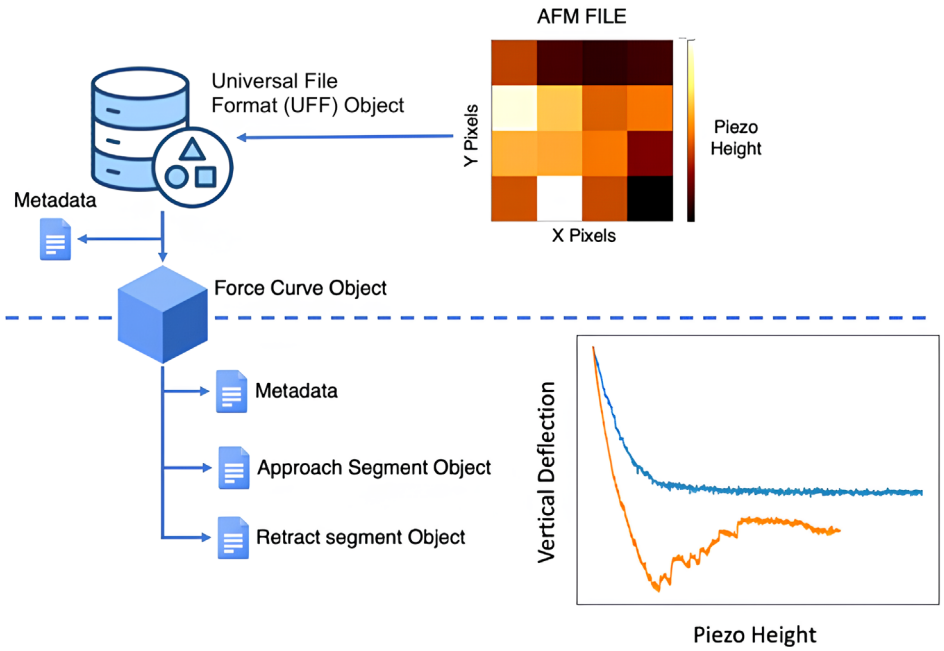

Figure 2. Hierarchical data organization used in PyFMLab. UFF stands for universal file format.

segment objects (approach, pause, modulation and retract segments). The name of the file is used as the id to relate all data objects from each file.

PyFMRheo

The source code for PyFMRheo is available (from zenodo López-Alonso, 2023a) (also available from GitHub)

As the main building block of PyFMLab, this library includes a wide variety of models and predefined routines for the nanomechanical characterization of biological samples, correction for hydrodynamic drag, calibration of the z-piezo for oscillatory measurements and non-contact calibration of AFM probes. As PyFMReader and PyFMRheo have been developed with the intention of being used together, the predefined routines available in PyFMRheo expect input FDCs and thermals as objects defined by PyFMReader. Nevertheless, this library can be used on its own, with the user being able to decide which library to use to load their data, which allows it to be used for the analysis of any data acquired on lab made or other commercial instruments. All models and tools required for processing AFM force curves and thermal tunes are fully accessible, allowing the users to compose their own custom data processing scripts. Examples on how to use each of the predefined routines are provided as interactive Jupyter notebooks. In addition, a tutorial on how to implement a contact model is provided in the dedicated GitHub site.

Elastic models. Contact models describing the force (F) versus indentation (δ) relationship allowing to obtain the Young's modulus (E) from the approach segment of FDC for several AFM tip geometries have been included (see chapter 2.2 Contact Mechanics Lacaria *et al.* in Lekka *et al.*, 2023a). By default, the fitting routine considers the point of contact and force offset as fitting parameters.

Additionally, the library includes a model considering the finite thickness of the sample (Table 2) developed by Garcia and Garcia (Garcia & Garcia, 2018), where the thickness of the sample is considered when determining the Young's modulus.

Due to the complexity of automatically determining the thickness of the sample at each pixel, in particular for force maps, the models incorporating Garcia and Garcia bottom effect correction are implemented in the PyFMRheo analysis library but are not accessible through the graphical user interface (GUI).

Default routine implementation

1. The raw data of the curve is pre-processed to obtain the tip position (δ) by subtracting the cantilever vertical deflection (d) to the piezo position (z).

2. An estimate of the point of contact (PoC) is computed using the ratio of variances (RoV) method based on Gavara, 2016. This is computed in two small windows before and after each point of the deflection signal (d):

$$RoV_i = \frac{var\left(d_{i+1} : d_{i+N}\right)}{var\left(d_{i-N} : d_{i-1}\right)},$$

   Where RoV_i is the ratio of variances at each data point i, var is the variance, d is the deflection signal and N is the window size inputted by the user.

3. The spring constant and initial estimation of the PoC are used to obtain the force *vs* indentation curve.

4. Using the parameters defined by the user, the corresponding model is fitted to the force *vs* indentation curve to obtain the refined PoC and the Young's modulus.

5. The default fitted parameters are the Young's modulus (E), the PoC and the force offset.

Viscoelastic models. Several models have been proposed to determine the viscoelastic force-indentation relationship of a sample. These models are numerical (Efremov *et al.*, 2017) or analytical (Brückner *et al.*, 2017; Garcia *et al.*, 2020; Eroles *et al.*, 2023) solutions and are based on the seminal works by Lee and Radock (Lee & Radok, 1960), Graham (Graham, 1967) and Ting (Ting, 1966; Ting, 1968). The selection and implementation of the models in PyFMLab were performed focusing on the determination of the mechanical properties of living mammalian cells and extracellular matrices. Such samples have been shown to exhibit a power law response both at low and high frequencies (Balland *et al.*, 2006; Fabry *et al.*, 2001; Jorba *et al.*, 2017; Jorba *et al.*, 2019; Rigato *et al.*, 2017). PyFMRheo includes both numerical (Table 3) and analytical (Table 4) models for a power law response material.

In the models, F is the force acting on the cantilever tip, δ is the indentation which is a function of the time, t. Following (Brückner *et al.*), the time t_1 is obtained from the auxiliary function determined by:

$$\int_{t_1(t)}^{t} E(t-\xi)\frac{\partial \delta}{\partial \xi}\,d\xi = 0,$$

where ξ is the dummy time variable required for the integration, $E(t)$ is the Young's or elastic relaxation modulus described by a power-law rheology (PLR) model:

$$E(t) = E_0\left(\frac{t}{t_0}\right)^{-\beta},$$

where E_0 is the scaling factor or instantaneous elastic modulus, t_0 an arbitrary time scale assumed to be 1s, and β is the power-law exponent or fluidity (β = 0 for material with solid-like behaviour, $\beta = 1$ for material with fluid-like behaviour).

Assuming that the indentation is proportional to the different approach and retract velocities, t_1 is found to be:

$$t_1 = t - \left(1+\frac{v_r}{v_a}\right)^{\frac{1}{1-\beta}}(t-t_m),$$

where v_a is the constant approach velocity, v_r is the constant retract velocity (Brückner *et al.*, 2017).

$\mathcal{C}$ is the geometry dependent coefficient. The definition of this parameter based on the tip geometry is included in Table 5. In addition to the models listed in Table 1–Table 5, other models have been implemented in pyFMRheo, including a simple linear model to estimate the sample stiffness, the Derjaguin-Muller-Toporov (DMT) model and the Kontomaris approximation of a spherical indenter (Kontomaris & Malamou, 2021).

Default viscoelastic routine implementation

1. The raw data of the curve is pre-processed to obtain the tip position by subtracting the cantilever vertical deflection to the piezo position.

2. An estimate of the point of contact (PoC) is computed using the ratio of variances (RoV) method based on Gavara, 2016.

3. The spring constant and initial PoC estimation are used to obtain the force vs. indentation curve.

4. The elastic model corresponding to the AFM tip geometry is fitted to the approach segment of the force vs. indentation curve to obtain an estimation of the Young's modulus and a better estimation of the PoC.

5. The corresponding viscoelastic model is then fitted to the full approach-retract cycle of the force vs indentation curve to obtain the power-law fluidity exponent (β), the instantaneous elastic modulus (E_0) and the PoC. The viscous drag force (force felt by the cantilever due to the viscosity of the medium) can be corrected, from the correction factor specified by the user and the respective approach and retract velocities.

Determination of the complex shear modulus from active microrheology measurements. Dynamic mechanical analysis (DMA), or microrheology measurements, consist of applying deformations in a cyclical manner to a material in the following manner:

1. The cantilever approaches and indents the sample.

2. The cantilever oscillates at a single frequency. This step is repeated in order to measure all the desired frequencies (see AFM measurements section below for details).

3. The cantilever is retracted from the sample.

These cyclic forces result in variable deformations. As the stress (force applied to an area of the material) or strain (material deformation) applied is oscillatory (sinusoidal), the response of the sample is also sinusoidal. If a material is purely elastic, the strain and stress will be perfectly synchronous, at all times, meaning that, when the force is applied, the material will deform, and when the force stops being applied, the sample will recover its original shape. If the material is purely viscous, an applied stress will induce a strain response that is both delayed in time and not recovered once the application of stress ceases. Generally, a material will exhibit both elastic and viscous behaviors, resulting in a strain response that will lag (phase angle 0 – 90°) behind the applied stress. From DMA, the following frequency dependent parameters can be obtained:

- Complex modulus (G^*): Overall resistance of the material to deformation including a real part (elastic, G') and an imaginary part (viscous, G'').

- Storage modulus (G'): Elasticity of the materials and its ability to store energy.

- Loss modulus (G): Viscosity of the material and its ability to dissipate energy.

- Loss tangent (η): Degree of solid- or liquid-like mechanical behavior of the material.

Table 6 shows the values for G', G'' and η for purely elastic and purely viscous materials. Mammalian cells and extracellular matrices (ECM) are viscoelastic materials, with both an elastic and viscous component. In 2003 Alcaraz *et al.* defined a methodology for the microrheological characterization of biological samples with AFM. When a sample is cyclically indented in AFM, a certain stress is applied to the sample, causing it to deform. Both the stress and the deformation of the sample can be measured through force-indentation curves, where the force can be used to monitor the stress applied, and the indentation signal can be used to monitor the corresponding deformation of the sample.

Models for two AFM tip geometries have been implemented and are shown in Table 7.

The default routine expects data which has been acquired by indenting the sample and applying single frequency oscillations at consecutive segments, as shown in Figure 5.

Default rheology routine implementation

1. The raw data of the curve is pre-processed to obtain the tip position by subtracting the cantilever vertical deflection to the piezo position.

2. The force curve is pre-processed to obtain the piezo position *vs* tip position.

3. An estimate of the point of contact (PoC) is computed from the approach segment using the ratio of variances (RoV) method based on (Gavara, 2016). This allows determining the value of the working indentation (δ_0).

4. If the user provides piezo calibration data and selects the option, the amplitude and phase of the z-piezo signal are corrected.

5. Both signals are then detrended by subtracting the rolling average of the signal with a window size equal to the number of points of each period.

6. The transfer function (H_d) is computed considering the ratio of the Fourier transforms of the small amplitude indentation ($\delta(\omega)$) as the input signal and the measured force ($F(\omega)$) as the output signal.

$$H_d(\omega) = \frac{F(\omega)}{\delta(\omega)}$$

7. If the user provides a value for the drag force at contact, the transfer function values are corrected accordingly.

8. $G^*(\omega)$ is computed using the corresponding frequency domain rheological contact model. For a paraboloid geometry: $G^*(\omega) = (1-\nu)/4\,\mathrm{sqrt}\left(R\delta_0\right)H_d(\omega)$. From the values of $G^*(\omega)$, $\eta(\omega)$ is computed.

Z-Piezo characterization. When characterizing the microrheological properties of cells or ECM, any delay between the force and indentation signals inherent to the instrument will result in an over- or underestimation of the viscoelastic properties of the sample. By acquiring oscillatory measurements in a non-deformable substrate, like glass, the inherent phase angle (ϕ_{PZT}) between the deflection and z movement signals of the instrument can be measured and used to correct the measurements acquired on the sample.

$$G^*_{corr} = G^* e^{-i\phi_{PZT}}$$

In addition to the previous correction, in this routine, the ratio between the vertical deflection and the z-piezo signals is computed. This value can then later be used to correct the z-piezo signal acquired on the sample.

$$z_{corr} = zA_{corr}$$

Default z-piezo characterization routine implementation

1. Both the z-piezo displacement (z) and the vertical deflection of the cantilever are detrended (d) by subtracting the rolling average of the signal with a window size equal to the number of points of each period.

2. The transfer function is computed considering z_{PZT} as the input signal and d_v as the output signal.

$$H_d = d/z$$

3. From the transfer function, the phase between z and d signals at each applied frequency are obtained.

4. Finally, the signal amplitude quotient between z and d signals is computed.

$$A_{corr} = \frac{A_d}{A_z}$$

Determination of viscous drag coefficient. To minimize the noise in the measurements and to provide a suitable environment when working with living cells, most AFM microrheological measurements are performed in liquid. Therefore, when the cantilever oscillates, the surrounding fluid applies force on the cantilever by adding resistance to the motion (hydrodynamic drag). In the same manner as the piezo lag, the hydrodynamic drag results in an overestimation of the viscous properties of the measured sample. To account for this effect, Alcaraz and collaborators (Alcaraz *et al.*, 2002) proposed a correction applicable to AFM measurements on soft samples in liquid at low Reynold numbers (Re < 1). They approached this problem by modelling the AFM cantilever as a sphere close to a rigid wall. The characterization of the hydrodynamic on a spherical body close to a rigid wall has been addressed by multiple authors (Brenner, 1961;

Cox & Brenner, 1967). Based on these works, Alcaraz and collaborators estimate the drag force $b(h)$ when approaching the sample to be:

$$b(h) = \frac{6\pi\eta a_{eff^2}}{(h + h_{eff})}$$

Where η is the viscosity of the fluid, a_{eff} is the effective radius of the cantilever, h_{eff} is the effective height of the tip and h is the distance between the tip and the sample.

Using this scaled spherical model, the drag force at contact ($b(0)$) can be estimated by measuring the drag force $b(h)$ at different cantilever-sample separations and extrapolating the data to h = 0.

To compute $b(h)$ the default routine expects data acquired with a specific protocol:

1. The AFM tip approaches and indents the sample (cell, tissue, gel, *etc.*).
2. The cantilever retracts a certain distance defined by the user.
3. The cantilever is oscillated at a single frequency defined by the user.
4. Steps 2 and 3 are repeated until the maximum desired distance from the sample has been reached.

An example of data collected using this protocol is shown on Figure 6.

Default viscous drag routine implementation

1. If the user provides piezo calibration data and selects the option to correct the amplitude of the z-piezo, the signal is corrected.
2. The transfer function (H_d) is computed considering the indentation (δ) as the input signal and the measured force (F) as the output signal.

$$H_d = \frac{F(\omega)}{\delta(\omega)}$$

3. If the user provides piezo calibration data, the transfer function values are corrected using the piezo amplitude and lag.
4. $b(h)$ is then computed using the following expression:

$$b(h) = \frac{H_d}{(2\pi\omega)}$$

Non-contact cantilever calibration. Many indirect spring constant calibration methods have been developed, with the Sader method being the most widely used, thanks to its ease of use and applicability to different cantilever geometries (rectangular, V-shaped) (Sader *et al.*, 1999; Sader *et al.*, 2005; Sader *et al.*, 2012). This method relies on measuring the quality factor (Q) and resonant frequency (ω_R) from the thermal spectrum of the cantilever to estimate the spring constant (k) by relying on hydrodynamic theory. When the cantilever oscillates, the surrounding fluid (air, water, *etc.*) applies force on the cantilever in two ways: by adding resistance to the motion (hydrodynamic drag) and by adding mass to the cantilever. By relating the measured ω_R and Q factor of the cantilever oscillating in a fluid to the values of the cantilever oscillating in vacuum, an analytical model to determine the spring constant of the cantilever for the first eigenmode (k_1) can be derived, by taking into consideration the geometry of the cantilever together with the density and viscosity of the fluid (Sader *et al.*, 2005). The spring constant of a rectangular cantilever can be determined applying the following formula:

$$k = 0.1906\rho b^2 L\Gamma_i(\omega_R)\omega_R{}^2 Q$$

Where ρ is the density of the medium, b is the thickness of the cantilever, L is the length of the cantilever, Γ_i are the imaginary components of the hydrodynamic function, ω_R is the resonance frequency of the cantilever and Q is the quality factor of the cantilever. Although this expression is limited to rectangular cantilevers, the Sader method was later extended to cantilevers of arbitrary shape (Sader *et al.*, 2012)

With the goal of allowing to calibrate the spring constant of cantilevers with a reference standard, Sader *et al.* developed a virtual instrument (that allows the user to obtain the spring constant of a cantilever regardless of its shape by measuring ω_R and Q. This method is referred to as the global calibration initiative (GCI) (Sader *et al.*, 2012; Sader *et al.*, 2016).

In 2006, Higgins *et al.* proposed a non-contact method to obtain the optical lever sensitivity (or deflection sensitivity) from the thermal spectrum of the cantilever, assuming that the spring constant remains unchanged in air and liquid. This approach requires a previous calibration of the spring constant in air with the Sader or other method (Higgins *et al.*, 2006). As this method yields a deflection sensitivity different to the one obtained by contact-based methods, the result is then multiplied by a correction factor to obtain the correct value.

Although the Sader and GCI methods enhance the precision of nanomechanical measurements when compared to conventional contact-based approaches (Sader *et al.*, 2012; Sader *et al.*, 2016), they both assume cantilevers with a high Q value, which is not always the case for most cantilevers in liquid. For this reason, in 2020, Sumbul and collaborators (Sumbul *et al.*, 2020) assessed the accuracy and precision of these methods. In this study, they observed that using the single harmonic oscillator (SHO) model to fit the thermal spectra together with the GCI method was less prone to systematic uncertainties and provided higher accuracy in the determination of k and the deflection sensitivity (Sumbul *et al.*, 2020). The implementation of the calibration routine in this package has been based on these works.

Default spring constant calibration routine implementation

1. The SHO model is fit to the first eigenmode peak of the thermal data to obtain the white noise of the signal

(A_{white}, the amplitude of the peak (B), the resonance frequency (f_R and Q factor of the cantilever.

$$S_{SHO}(f) = A_{white}^2 + \frac{B^2 f_R^4}{Q^2}\left[\left(f^2 - f_R^2\right)^2 + \frac{f^2 f_R^2}{Q^2}\right]^{-1}$$

2. From these values the spring constant can be computed using the general Sader method. This step is only valid for rectangular and V-shaped cantilevers.

3. If the user provides thermal data acquired in air, their username and password for the GCI web application, the software will get the GCI spring constant through the application programming interface of the GCI web application.

4. The deflection sensitivity (or inverse of the optical lever sensitivity, invOLS) is computed using the method proposed by Higgins *et al.* and Sumbul *et al.* The user can specify the factors to correct for the static versus dynamic deflection (sometimes referred as χ) and the fundamental oscillation mode (sometimes referred as β) of the cantilever (Chighizola *et al.*, 2023). If no correction factor is specified, the default values ($\sqrt{\beta}/\chi$) of 0.90 (rectangular) and 0.87 (V-shaped) are used (Pirzer & Hugel, 2009; Stark *et al.*, 2001; Sumbul *et al.*, 2020).

PyFMGUI

The source code for PyFMGUI is available (from zenodo López-Alonso, 2023a) (also available from GitHub)

To make the software more accessible to the end user, a GUI has been developed. This GUI allows the user to load, visualize and process data using the methods available in the PyAFMReader and PyAFMRheo libraries. Multiprocessing has been implemented to speed up loading and processing of large files.

The multiple-document interface (MDI) has been chosen as the model to develop the GUI. This model allows the user to have multiple analysis windows open within the application. The cross-platform Qt5 framework was chosen to program the GUI. As PyFMLab is programmed in Python, the library PyQT5 was used to access the bindings for Qt5. Thanks to its speed, PyQTGraph was used to produce the plots of the application.

As a free, cross-platform, and versatile high-level programming language, Python has gained widespread adoption within the scientific community. This popularity has, in turn, spurred the development of a rich ecosystem of Python scientific packages, enabling tasks such as data visualization, machine learning, natural language processing, and complex data analysis, among others. Table 8 provides a list of the widely recognized and well-supported Python packages used to develop PyFMLab.

For manipulating arrays and matrices, numerical computing, and linear algebra Numpy (Harris *et al.*, 2020) was the library selected. The Pandas (McKinney, 2010) library was used to organize the results into a tabular form and export them as coma separated values (.csv) files. The Scipy (Virtanen *et al.*, 2020) library was used in PyFMLab for signal detrending and the implementation of the gaussian filter, one-dimensional discrete Fast Fourier Transform, Beta function, Gamma function and Besel function. Finally, for Non-Linear Least-squares minimization using the Levenberg-Marquardt algorithm, the Lmfit (Newville *et al.*, 2023) library was used.

Software validation

Cell culture

HeLa cells (ATCC, Manassas, VA, USA) were cultured in Minimum Essential Medium (MEM) (ThermoFischer, Waltham, MA, USA) containing 10% fetal bovine serum, 10% L-Glutamine and 1% Penicillin/Streptomycin. Cells were then seeded in 35 mm diameter glass-bottom Petri dishes (WillCo, Amsterdam, NL) and incubated overnight at 37ºC and 5% CO_2.

AFM measurements

For measuring the mechanical properties of cells, maps of force curves, with a size of 30 µm x 30 µm and a resolution of 4 × 4 pixels, were acquired on the nucleus area of 20 cells using a NanoWizard III (Bruker, Santa Barbara, CA, USA). The low pixel resolution per cell was chosen to maximize the number of cells per sample without compromising the cell viability. The maps were acquired using a SAA-SPH-5UM probe (hemispherical tip, 5 µm radius, 0.192N/m spring constant) (Bruker, Camarillo, CA, USA) using the following parameters: 1 nN force setpoint, 4 µm ramp size and 20 µm/s ramp speed. During all measurements, indentations ~500 nm were targeted to avoid the bottom effect of the substrate.

We acquired two datasets. The first dataset (A, simple force curves) involved approaching and retracting the tip at a constant velocity to collect data on 20 HeLa cells for analysis using elastic and viscoelastic models. The second dataset (B, oscillatory measurements) involved approaching the tip at a constant velocity, oscillating the tip in contact with the sample at the frequencies mentioned below and retracting at constant speed and was used to collect data on 20 HeLa cells for analysis using complex shear modulus models. In this latter case, sinusoidal oscillations of 15-nm amplitude at 0.6 Hz, 1 Hz, 10 Hz, 60 Hz, 120 Hz and 200 Hz were applied during the z-piezo characterization and microrheological measurements on cells. To compute the viscous drag coefficient at contact, the cantilever was oscillated at 500 Hz with 15-nm amplitude at tip-sample distances between 500 nm and 3 µm from the contact surface.

AFM data processing

Data was processed using three software packages:

- PyFMLab: The software package developed in this work.

- MATLAB (RRID: SCR_001622) Routines: Custom routines developed in MATLAB and applied before for elastic (Rico *et al.*, 2005) and viscoelastic (Gerum *et al.*, 2022) FDCs data processing.

- JPK DP (v.7.1.23): Commercial software developed and provided by Bruker, used to visualize and process data acquired with the NanoWizard AFM platform.

Determination of the Young's modulus of living HeLa cells. The Young's modulus (E) of 20 HeLa cells was measured, as shown in Figure 3, by fitting the paraboloidal Hertz's contact model (Table 1) to the approach segment of each force-distance curve of sthe dataset A. All three software packages were used for this analysis and the results were used for comparison. Other software packages openly available to extract the Young's modulus, such as AtomicJ (Hermanowicz *et al.*, 2014) or pyJibe (Müller *et al.,* 2019), led to similar results (see Results section).

Determination of the viscoelasticity of living HeLa cells. The same dataset A (simple force curves) was used to compute the scaling factor E_0 and the power-law fluidity β exponent by fitting the analytical model developed by Brückner *et al.*, 2017 for a paraboloidal tip (Table 4), as shown in Figure 4.

Only the MATLAB routines and PyFMLab were used for this analysis. The standard error for E_0 values obtained from the two packages were calculated using bootstrapping, by computing a two-sided bootstrap confidence interval of the E_0 median (Scipy).

Microrheological measurements on HeLa cells. For this analysis, the dataset B (oscillatory measurements) acquired on 20 HeLa cells was used to compute the power-law parameters (scaling factor G_0 and fluidity exponent β). To correct for the effect of the viscous drag and Z-piezo artefacts, the phase lag and amplitude quotient of the Z-piezo, together with the drag force at contact, were determined using PyFMLab (Figure 8).

The geometric mean of the $G^*(f)$ for each cell was then fitted to a double power law:

$$G'(f) = A\cos(\pi\alpha/2)f^{\alpha} + B\cos(\pi\beta/2)f^{\beta}$$
$$G''(f) = A\sin(\pi\alpha/2)f^{\alpha} + B\sin(\pi\beta/2)f^{\beta}$$

By least squares minimization using the Lmfit (Newville *et al.*, 2023) Python 3 library, leaving the four parameters A, B, α and, β free (Figure 9). Microrheology measurements were only processed using PyFMLab.

Scanning electron microscopy

AFM probes were mounted on stub with a carbon double-side tape then observed with a Zeiss Merlin Compact FESEM (Zeiss, France) working at 1 kV.

Results

Comparison between apparent Young's modulus values. To validate the elastic model fit of PyFMLab, a dataset comprising measurements of 20 HeLa cells was analyzed using three different software packages (Figure 10). From the analysis the following results were obtained: JPK DP E = 676 Pa ± 93 Pa (median ± SE), MATLAB routines E = 776 Pa ± 110 Pa and PyFMLab E = 641 Pa ± 32 Pa. Where SE refers to the standard error, computed through bootstrapping (10000 resamples). For comparison, the AtomicJ package provided a value of 577 Pa. All values are in agreement with each other and vary in a range of 4–17%, with the MATLAB routines and PyFMLab showing the greatest difference (135 Pa). PyFMLab allows more robust determination of the contact point thanks to the RoV

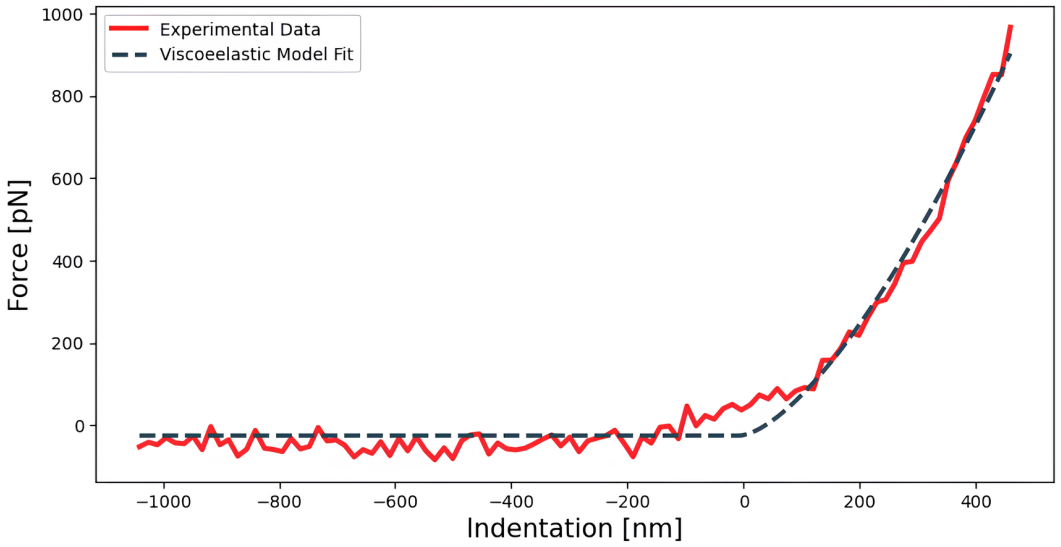

Figure 3. Approach segment of a force *vs* indentation curve acquired on a HeLa cell fitted to a Hertz paraboloidal elastic model, with an apparent Young's modulus equal to 739 Pa.

Table 1. Implemented contact elastic relationships between force (*F*) and indentation (*δ*). *R* corresponds to the tip radius, *v* to the Poisson's ratio, *E* to the Young's modulus, *θ* to the semi-opening angle of the cone or the pyramid face, $F_{Hertz}(z-d_0)$ is the force obtained from applying the Hertz model and $F_{Adhesion}(d_0)$ is the adhesion force according to DMT theory (Derjaguin *et al.*, 1975).

Tip geometry	Formula	Reference
Paraboloidal	$\mathrm{F}(\delta)=\frac{4\sqrt{R}}{3\left(1-v^2\right)}E\delta^{\frac{3}{2}}$	(Hertz, 1882)
Conical	$\mathrm{F}(\delta)=\frac{2\tan\theta}{\pi\left(1-v^2\right)}E\delta^2$	(Love, 1939; Sneddon, 1965)
4-sided regular pyramid	$\mathrm{F}(\delta)=\frac{\tan\theta}{\sqrt{2}\left(1-v^2\right)}E\delta^2$	(Barber & Billings, 1990; Bilodeau, 1992)
Paraboloidal	$F=\begin{cases}F_{Hertz}(z-d_0)+F_{Adhesion}(d_0), z\le d_0\\ F_{Adhesion}(z), z>d_0\end{cases}$	(Derjaguin *et al.*, 1975)

Table 2. Implemented bottom effect corrected elastic relationships between force (*F*) and indentation (*δ*). *R* corresponds to the tip radius, *v* to the Poisson's ratio, *E* to the Young's modulus, *h* to the sample thickness and *θ* to the semi-opening angle of the cone or the pyramid face.

Tip geometry	Formula	Reference
Paraboloidal	$F(\delta)=F_0(\delta)\left[\frac{1}{h^0}+\frac{1.133\sqrt{\delta R}}{h}+\frac{1.497\delta R}{h^2}+\frac{1.469\delta R\sqrt{\delta R}}{h^3}+\frac{0.755(\delta^2R^2)}{h^4}\right]$ $F_0(\delta)=\frac{4\sqrt{R}}{3\left(1-v^2\right)}E\delta^{\frac{3}{2}}$	(Garcia & Garcia, 2018)
Conical	$F(\delta)=F_0(\delta)\left[\frac{1}{h^0}+\frac{0.721\delta\tan\theta}{h}+\frac{0.650\delta^2(\tan\theta)^2}{h^2}+\frac{0.491\delta^3(\tan\theta)^3}{h^3}+\frac{0.225\delta^4(\tan\theta)^4}{h^4}\right]$ $F_0(\delta)=\frac{2\tan\theta}{\pi\left(1-v^2\right)}E\delta^2$	
4-sided regular pyramid	$F(\delta)=F_0(\delta)\left[\frac{1}{h^0}+\frac{0.721\delta\tan\theta}{h}+\frac{0.650\delta^2(\tan\theta)^2}{h^2}+\frac{0.491\delta^3(\tan\theta)^3}{h^3}+\frac{0.225\delta^4(\tan\theta)^4}{h^4}\right]$ $F_0(\delta)=\frac{2\tan\theta}{\sqrt{2}\left(1-v^2\right)}E\delta^2$	
Paraboloidal	$F(\delta)=\frac{2ER}{1-v^2}\left(\frac{2}{3}c_1R^{-1/2}h^{3/2}+\frac{1}{2}c_2R^{-1}h^2+\frac{1}{3}c_3R^{-2}h^3+\frac{1}{4}c_4R^{-3}h^4+\cdots\frac{1}{N}c_NR^{1-N}h^N\right)+c_0$ $c_1=1.0100000,\ c_2=-0.0730300,\ c_3=-0.1357000$ $c_4=0.0359800,\ c_5=-0.0040240,\ c_6=0.0001653$	(Kontomaris & Malamou, 2021)

Table 3. Implemented numerical viscoelastic relationships between force (*F*) and indentation (*δ*). *R* corresponds to the tip radius, *v* to the Poisson's ratio, *E* to the elastic relaxation modulus, t_1 to the integration auxiliary function and *θ* to the semi-opening angle of the cone or the pyramid face.

Tip geometry	Force curve segment	Equation	Reference
Paraboloidal	Approach	$F\left(t,\delta(t)\right)=\frac{4\sqrt{R}}{3(1-\nu^2)}\int_0^t E(t-\xi)\frac{\partial\delta^{\frac{3}{2}}}{\partial\xi}d\xi$	(Efremov *et al.*, 2017)
	Retract	$F\left(t,\delta(t)\right)=\frac{4\sqrt{R}}{3(1-\nu^2)}\int_0^{t_1(t)} E(t-\xi)\frac{\partial\delta^{\frac{3}{2}}}{\partial\xi}d\xi$	
Conical	Approach	$F\left(t,\delta(t)\right)=\frac{2\tan\theta}{\pi(1-\nu^2)}\int_0^t E(t-\xi)\frac{\partial\delta^2}{\partial\xi}d\xi$	
	Retract	$F\left(t,\delta(t)\right)=\frac{2\tan\theta}{\pi(1-\nu^2)}\int_0^{t_1(t)} E(t-\xi)\frac{\partial\delta^2}{\partial\xi}d\xi$	
4-sided regular pyramid	Approach	$F\left(t,\delta(t)\right)=\frac{\tan\theta}{\sqrt{2}(1-\nu^2)}\int_0^t E(t-\xi)\frac{\partial\delta^2}{\partial\xi}d\xi$	
	Retract	$F\left(t,\delta(t)\right)=\frac{\tan\theta}{\sqrt{2}(1-\nu^2)}\int_0^{t_1(t)} E(t-\xi)\frac{\partial\delta^2}{\partial\xi}d\xi$	

Table 4. Implemented viscoelastic analytical relationships between force (*F*) and time (*t*) assuming power law rheology ($E(t)=E_0(t/t_0)^{-\beta}$). *Γ* corresponds to the gamma function, *R* to the tip radius, *θ* to the semi-opening angle of the cone or the pyramid face, *v* to the Poisson's ratio, E_0 to the scaling factor or instantaneous elastic modulus, *β* to the power law exponent or fluidity, t_0 to the scaling time, t_1 to the integration auxiliary function, v_a to the approach velocity and $\mathcal{C}$ to the tip geometry coefficients are defined in Table 5.

Tip geometry	Force curve segment	Formula	Reference
Paraboloidal	Approach	$\mathrm{F}(t)=\frac{v_a^{\frac{3}{2}}}{\mathcal{C}}E_0\frac{t_0^{\beta}3\sqrt{\pi}\Gamma(1-\beta)}{4\Gamma\left(\frac{5}{2}-\beta\right)}t^{\frac{3}{2}-\beta}$	Based on (Brückner *et al.*, 2017)
	Retract	$\mathrm{F}(t)=\frac{1}{\mathcal{C}}E_0 v_a^{\frac{3}{2}}\left(\frac{t}{t_0}\right)^{-\beta}t_1^{\frac{3}{2}}2F1\left(\frac{3}{2},\beta;\frac{5}{2};\frac{t_1}{t}\right)$	
Conical / 4-sided regular pyramid	Approach	$\mathrm{F}(t)=2\frac{v_a^2}{\mathcal{C}}E_0\frac{t_0^{\beta}\Gamma[2]\Gamma[1-\beta]}{\Gamma(3-\beta)}t^{2-\beta}$	
	Retract	$\mathrm{F}(t)=2\frac{v_r^2E_0t_0^{\beta}}{\mathcal{C}(2-3\beta+\beta^2)}\left[t^{2-\beta}+(t-t_1)^{1-\beta}\left(t_1+(1+\beta)t\right)\right]$	

method (Gavara, 2016). This ensures less dispersion of the E_0 results,whereas the fit errors were wider for the previous methods.

Comparisons between power-law fluidity exponent values obtained. The power-law fluidity exponent was measured on HeLa cells through both a time-domain and a frequency-domain method.

For the time-domain method, the dataset A was used to fit the viscoelastic model developed by Brückner *et al.* (Brückner *et al.*, 2017) to the full approach-retract cycle (as shown on

Table 5. Geometry coefficients used in the implemented viscoelastic analytical models (Table 4). *R* corresponds to the tip radius and θ to the semi-opening angle of the cone or the pyramid face.

Tip geometry	Coefficient
Paraboloid	$\mathcal{C} = 3\left(1-\nu^2\right)/\left(4\sqrt{R}\right)$
Cone	$\mathcal{C} = \pi\left(1-\nu^2\right)/\left(2\tan\theta\right)$
4-sided regular pyramid	$\mathcal{C} = 1.342\left(1-\nu^2\right)/\tan\theta$

Table 6. Values of the storage modulus (*G′*), loss modulus (*G″*) and loss tangent (η) for purely elastic and viscous materials. *E* is the Young's modulus, ν is the Poisson's ratio and ω is the angular frequency.

Purely elastic material	Purely viscous material
$G'(\omega) = \frac{E}{2(1+\nu)}$	$G' = 0$
$G''(\omega) = 0$	$G''(\omega) = \mu$
$\eta(\omega) = 0$	$\eta(\omega) \to \infty$

Table 7. Implemented frequency domain rheological models for the complex shear modulus (*G) as a function of frequency (ω).** Where $F(\omega)$ and $\delta(\omega)$ are the Fourier transform of the force and indentation, respectively, δ_0 corresponds to the working indentation at which small amplitude oscillations are applied, *R* corresponds to the tip radius and θ to the semi-opening angle of the cone or the pyramid face.

Geometry	Function	Reference
Paraboloidal	$G^*(\omega) = \frac{1-\nu}{4\left(R\delta_0\right)^{\frac{1}{2}}}\frac{F(\omega)}{\delta(\omega)}$	(Mahaffy *et al.*, 2000; Rico *et al.*, 2005)
Pyramidal	$G^*(\omega) = \frac{1-\nu}{3\delta_0\tan\theta}\frac{F(\omega)}{\delta(\omega)}$	(Alcaraz *et al.*, 2003)

Table 8. Open-source Python libraries used by PyFMLab.

Library name	Description	References
Numpy	Numerical computation and N-dimensional arrays.	(Harris *et al.*, 2020)
Pandas	Data manipulation and DataFrame data structures.	(McKinney, 2010)
SciPy	Algorithms for scientific computing.	(Virtanen *et al.*, 2020)
Lmfit	Non-Linear least-squares minimization and curve-fitting.	(Newville *et al.*, 2023)
PyQT 5	Graphical user interface framework.	https://github.com/PyQt5
PyQTGraph	User interface graphics.	https://github.com/pyqtgraph
PyInstaller	Freezing PyFMLab code for distribution.	https://github.com/pyinstaller

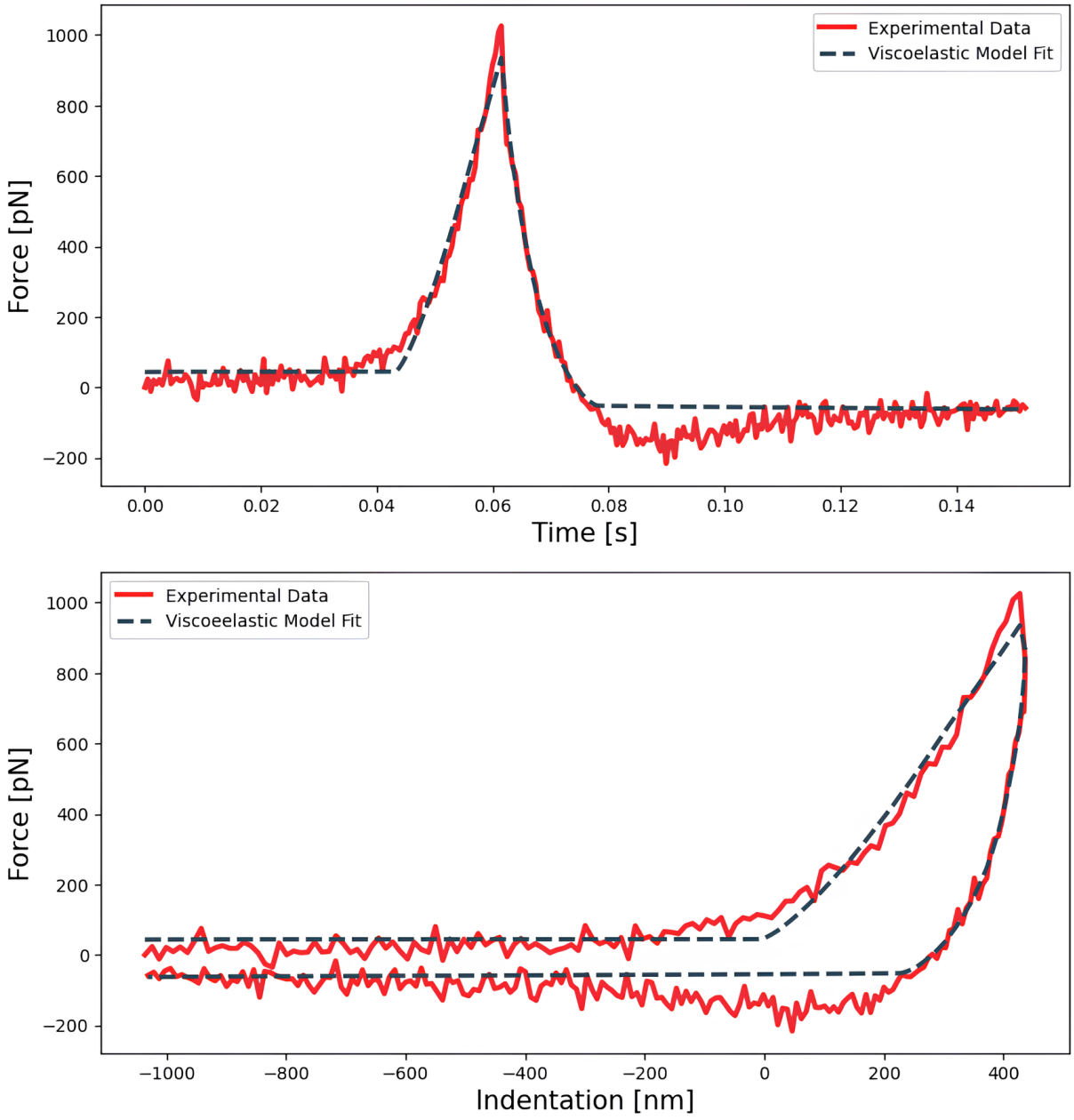

Figure 4. Force indentation curve acquired on a HeLa cell fitted to an analytic paraboloidal viscoelastic model, with an instantaneous elastic modulus (E_0) and fluidity exponent of 326 Pa and 0.18 respectively. The fit is done on the force *vs* time curve (top) and can then be displayed on the force *vs* indentation curve (bottom), allowing a direct comparison with the elastic fit shown on Figure 3.

Figure 4). The model was fitted using both the MATLAB routines and PyFMLab (Figure 11). As the measurements were performed in liquid, the data was corrected assuming a viscous drag factor of 0.003 pN nm^{-1} s. The following values were obtained from the analysis: Viscoelastic Fit MATLAB α = 0.21 ± 0.018 (median ± SE), Viscoelastic Fit MATLAB Corrected α = 0.13 ± 0.018, Viscoelastic Fit PyFMLab α = 0.18 ± 0.014, Viscoelastic Fit PyFMLab Corrected α = 0.12 ± 0.013. Where SE refers to the standard error, computed through bootstrapping (10000 resamples). As expected, the values corrected for the viscous drag were lower for fluidity exponent and higher for E_0 than the non-corrected ones. Nevertheless, the values obtained from both routines are in agreement and in the range of α values that other authors have reported for cells (Brückner *et al.*, 2017; Efremov *et al.*, 2017; Flormann *et al.*, 2021).

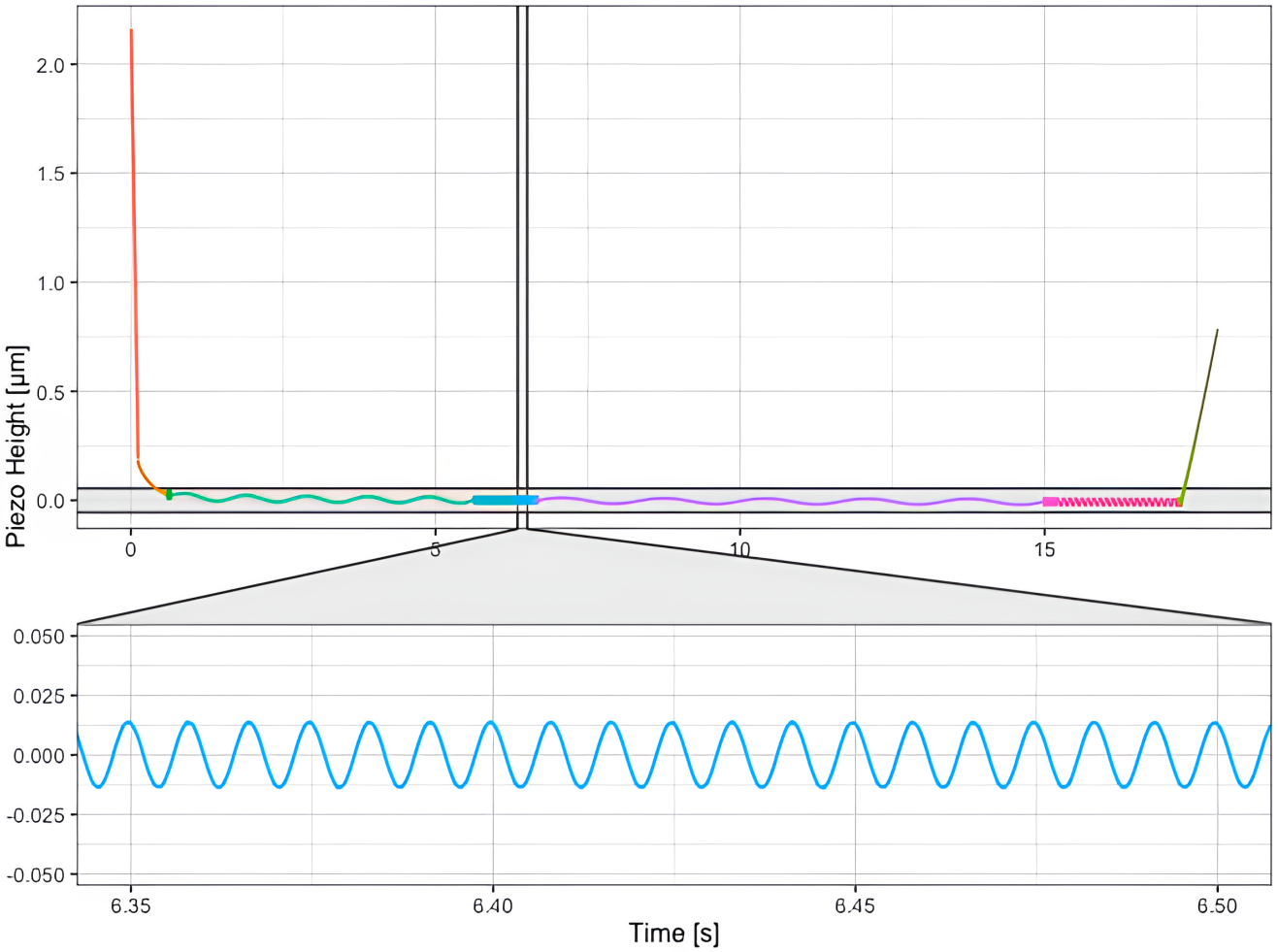

Figure 5. Microrheology data acquired on a HeLa cell. The cantilever is brought to contact with the sample up to reach the working indentation (red portion of the curve) and then oscillated at different frequencies (depicted in different colors, green, blue, violet, rose portions of the curve) before to be retracted from the sample (green portion of the curve). The inset shows the sinusoidal oscillations of the cantilever at a single frequency.

For the frequency-domain method, the dataset B, acquired on 20 other HeLa cells, was used. During the analysis, the measurements were corrected for the z-piezo phase lag, amplitude quotient and hydrodynamic drag ($b(0)$ = 0.005 pN s nm^{-1}). A α of 0.17 ± 0.013 (median ± SE) was measured. These values are within the range of values that other authors have reported for HeLa cells (Flormann *et al.*, 2021; Rother *et al.*, 2014) and are similar to the values obtained from the viscoelastic fits. This trend was previously observed by other authors (Efremov *et al.*, 2017).

Discussion

In this work, an open-source software package has been developed allowing us to measure the elastic and viscoelastic properties of soft samples from FDCs and microrheological measurements. The modularity of PyFMLab provides a user the possibility to use the package as a turn-key solution or to implement their custom features.

To validate the implemented algorithms, experimental data acquired on living HeLa cells was processed using three different software packages: commercial JPK DP, custom MATLAB routines and PyFMLab. The median apparent Young's modulus (E) of 20 cells was obtained using all three software packages. All results obtained are in agreement with each other and differ in a range of 4 – 17%. We attribute the observed differences to the use of different minimization algorithms which may affect the convergence of non-linear fits (for example, trust-region-reflective algorithm for Matlab routines, while Levenberg-Marquardt for pyFMRheo). Other possible sources are the estimation of initial parameters, the use of upper and lower bounds or the estimation of the point of contact (as a fitting parameter or independent of the fit). To assess the goodness of the fit, a list of parameters commonly used in regression analysis is exported together with the results in the .csv file. This include R-squared (R^2), Chi-square (χ^2), Reduced Chi-square (χ^2/ν), Mean Absolute Error (MAE),

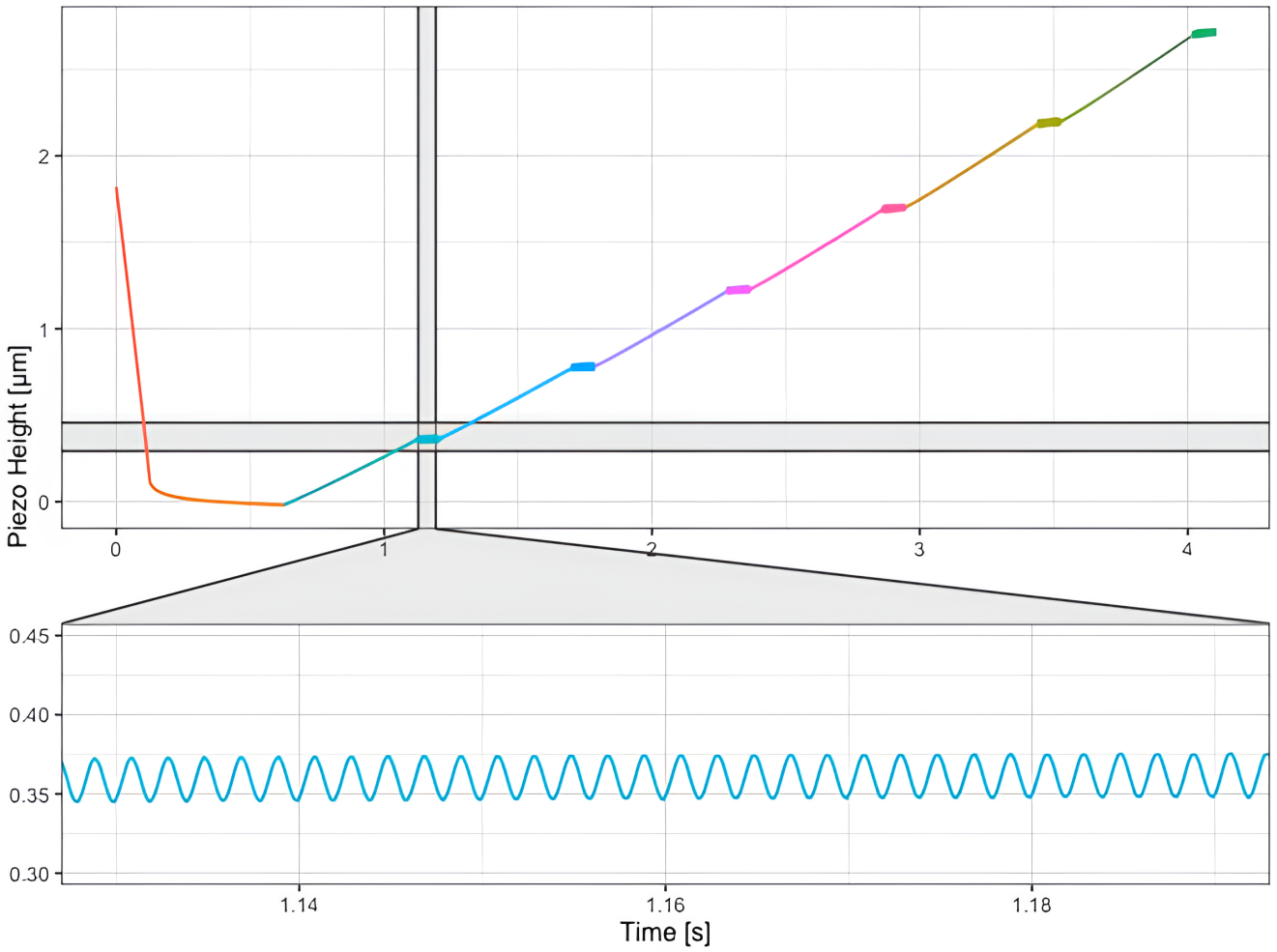

Figure 6. Data acquired on HeLa cells, using a SAA-SPH-5UM AFM (Figure 7) paraboloidal probe (Bruker, Santa Barbara, CA, USA), to determine the viscous drag. The cantilever is brought into contact with the sample (h = 0 µm, orange portion of the curve) and retracted 6 times (segments depicted in 6 different colors) until the maximum tip-sample distance of 2.75 µm is reached. After every retract, the cantilever is oscillated at the same frequency. This modulation step is shown in the inset.

Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The .csv file also includes the settings required to carry out the used fit, fostering reproducibility.

The viscoelastic properties of HeLa cells were characterized by both fitting a viscoelastic model to the approach-retract cycle (time-domain method) and by performing microrheological measurements (frequency-domain method). The viscoelastic model developed by Brücker *et al.* (Brückner *et al.*, 2017) was fitted to the approach-retract cycle of the FDCs to determine E_0 and the fluidity exponent using both the MATLAB routines and PyFMLab on the same dataset. Microrheology data was processed using PyFMLab. Values obtained for E_0 and the power-law fluidity exponent were in agreement both between software packages and methodologies. This opens the way to characterizing the rheological behavior of samples directly from datasets consisting of regular approach-retract force curves, for example, for high resolution mechanical maps. Thus, speeding data acquisition and analysis when compared to DMA experiments.

Although version 1.0 of PyFMLab includes the most widely used models and functionalities needed for mechanically characterizing biological samples, further improvements could be performed in the determination of the point of contact in the microrheology routine and other models to describe G^* could be implemented. A separation dependent viscous drag correction may be also added to the FDC based approach.

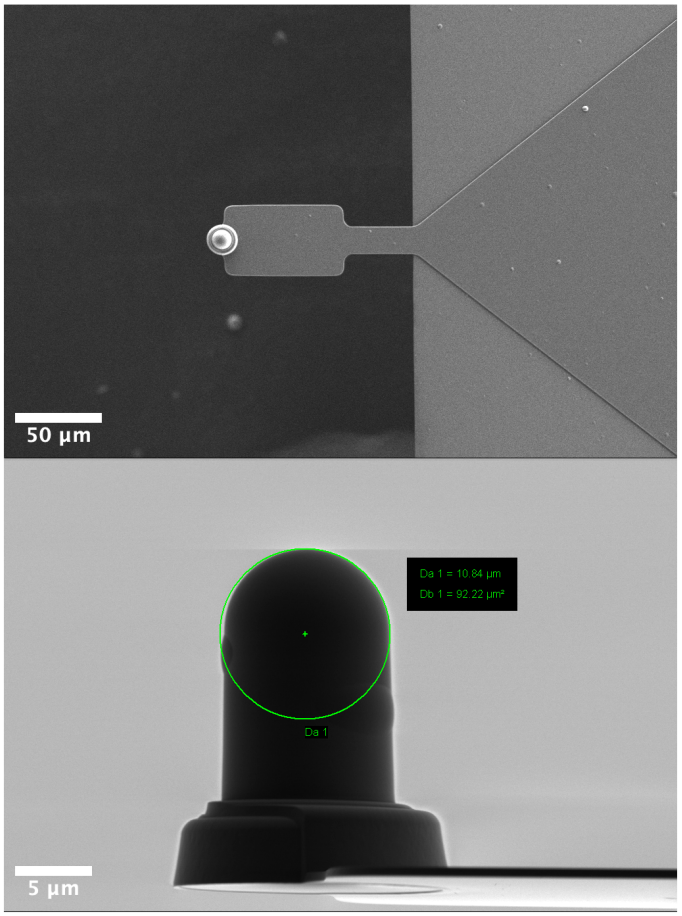

Figure 7. Scanning electron microscopy images of the SAA-SPH-5UM AFM probe (Bruker, Santa Barbara, CA, USA). Top view (top) Side view (bottom).

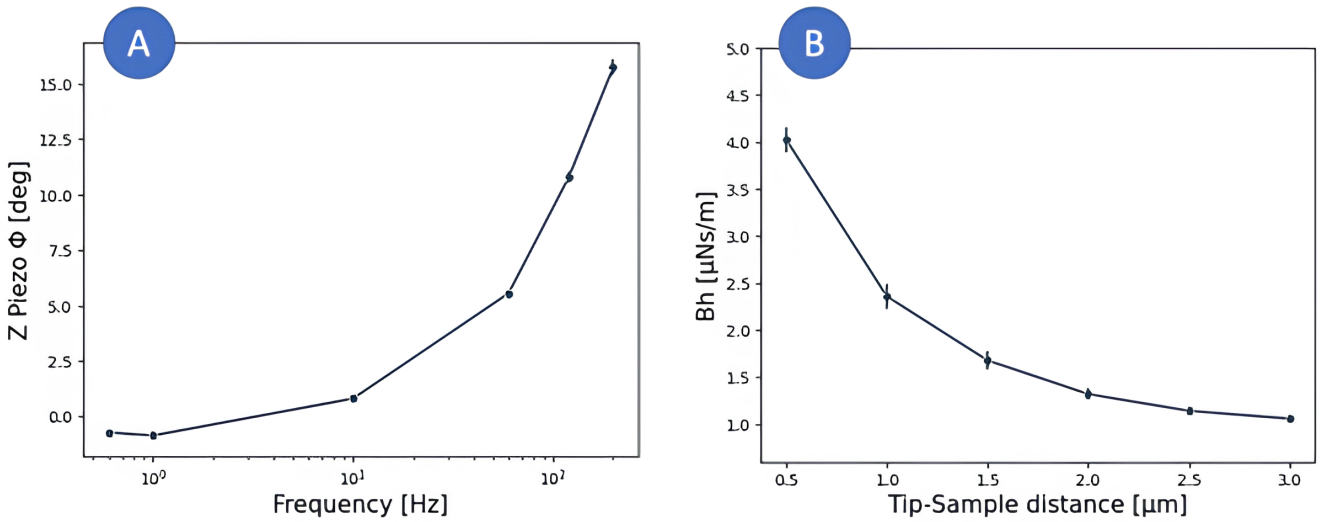

Figure 8. A) Z-piezo phase lag measured on a NanoWizard III for each frequency. **B**) Drag force measured at different cantilever-sample separations while applying 500 Hz oscillations. The data shown on each plot corresponds to the mean ± SE of three 4x4 force curves maps.

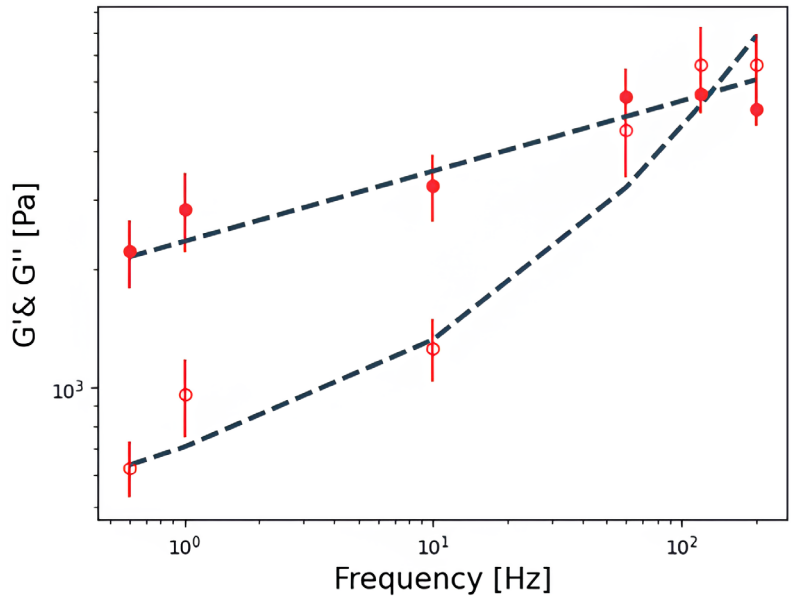

Figure 9. Experimental *G'*(*f*) (filled symbols) and *G''*(*f*) (hollow symbols) data of a cell fitted to a double power law. Each point shows the geometric mean ± SE of three 4x4 force curves map. The dashed line is a fit to the double power law with fitted parameters A, B, α and, β equal to 2663 Pa, 38 Pa, 0.17 and 0.99, respectively.

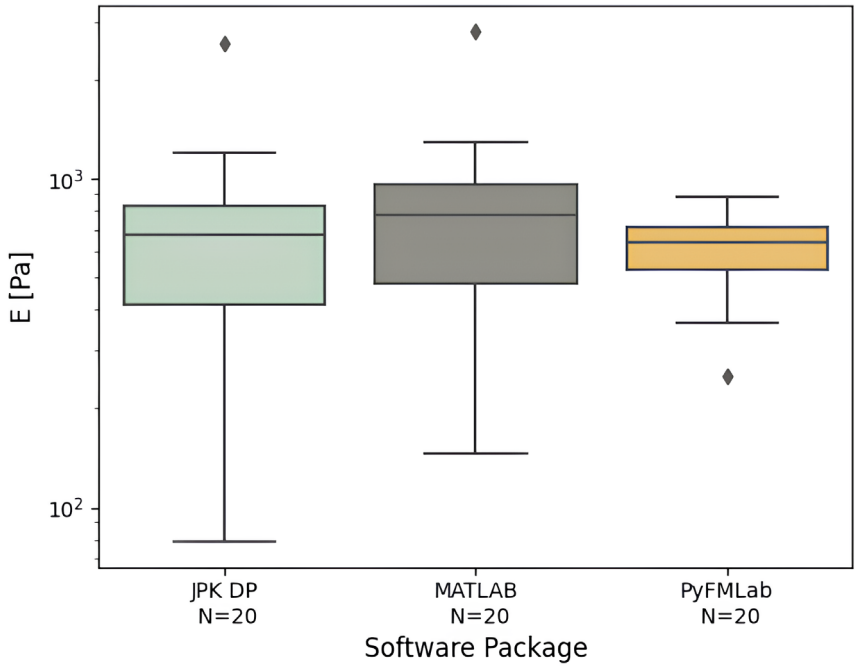

Figure 10. Median apparent Young's modulus (*E*) of individual HeLa cells (N=20 cells, with 16 measurements per cell) measured by different software packages. Centre line in the box plots represents the median value, top and bottom limits of the box represent the first and third quartiles (25% and 75%) of the data. Whiskers represent minimum and maximum values; diamonds represent the outliers (points outside 1.5 times the interquartile range).

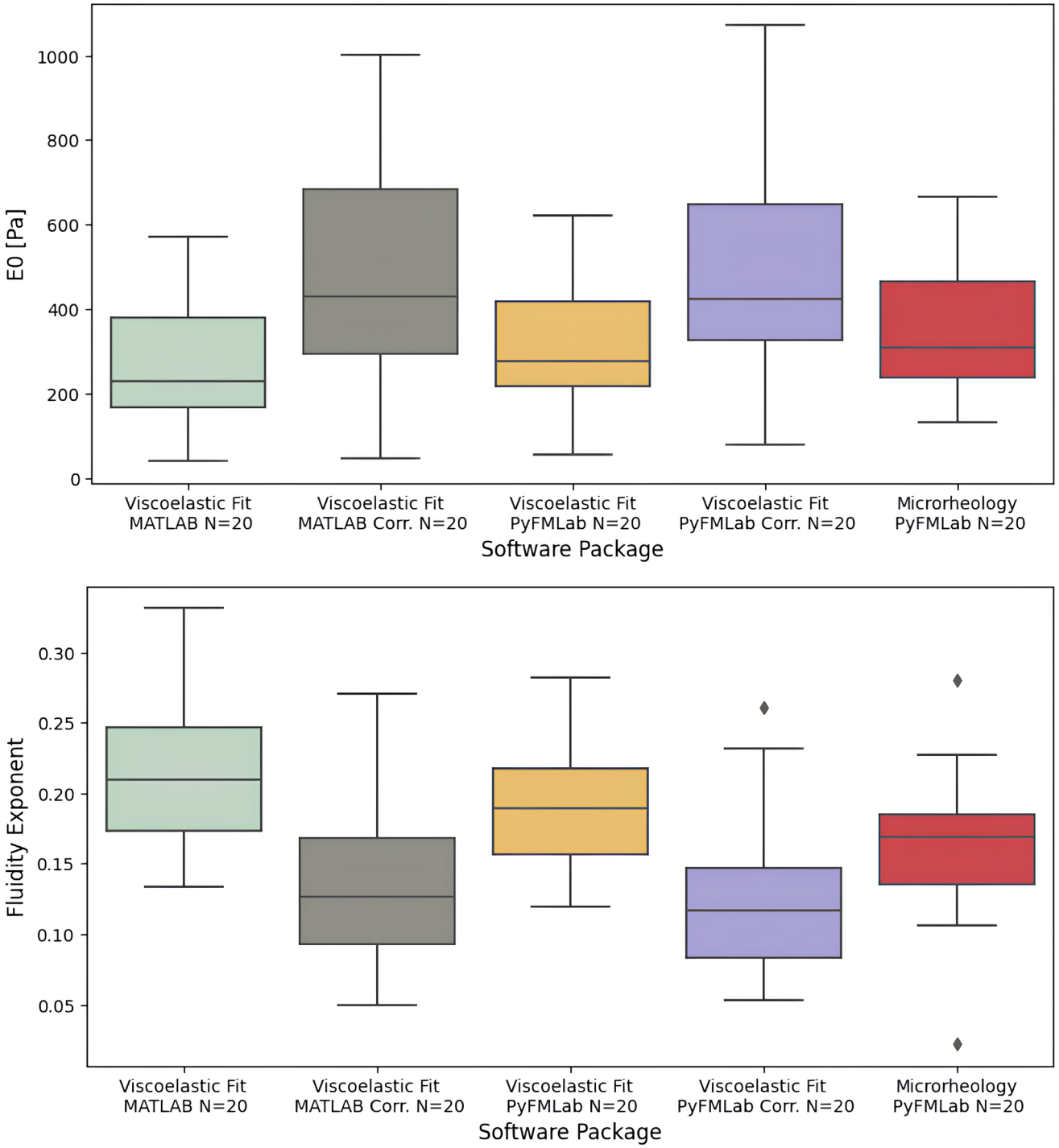

Figure 11. Instantaneous elastic modulus (E_0) (Top) and Power-law fluidity exponent (Bottom) of individual HeLa cells (N=20 cells, 16 measurements per cell) measured by different software packages. The values obtained from fitting the viscoelastic model represent the median E_0 and α with and without correcting for the viscous drag assuming a viscous drag of 0.003 pN nm^{-1} s. The values reported obtained from the microrheology measurements represent the fluidity exponent obtained by fitting the double power law model to the geometric mean for each cell. G (f) was corrected for the z-piezo phase lag, signal amplitude and hydrodynamic drag ($b(0)$ = 0.005 pN s nm^{-1}). Centre line in the box plots represents the median value, top and bottom limits of the box represent Q1 and Q3 of the data. Whiskers represent minimum and maximum value; diamonds represent the outliers (points outside 1.5 times the interquartile range).

Furthermore, implementation of thickness determination from AFM mechanical maps will allow accurate correction of the bottom effect across the cell surface.

In conclusion, PyFMLab allows determination of viscoelasticity of biological samples from both force-distance curves and microrheology measurements. We expect that PyFMLab will become a standard in the field, given its versatility, open-source nature, modularity and robustness.

Ethics and consent

Ethical approval and consent were not required.

Data availability

Underlying data

Zenodo: AFM Dataset on HeLa cells

https://doi.org/10.5281/zenodo.8342121 (López-Alonso, 2023b).

This project contains the following underlying data:

- Force Settings (HeLaDataset_20221029/ForceSettings)
- Approach-Retract curves acquired on glass (HeLaDataset_20221029/Glass) Oscillatory measurements acquired on HeLa cells (HeLaDataset_20221029/Microrheo)
- Oscillatory measurements acquired on glass for performing the piezo characterization (HeLaDataset_20221029/PiezoChar)
- Approach-Retract curves acquired on HeLa cells (HeLaDataset_20221029/SFC)
- Therma data acquired for non-contact calibration (HeLaDataset_20221029/Thermal)
- Oscillatory measurements acquired on the medium over HeLa cells to compute the viscous drag coefficient (HeLaDataset_20221029/VDrag)
- Results obtained from JPKDP v.7.1.23 (JPKDP_Results)
- Results obtained from MATLAB routines (MATLAB_Results)
- Results obtained from PyFMLab (PyFMLab_Results)

Data are available under the terms of the Creative Commons Attribution 4.0 International license (CC-BY 4.0).

Software availability

PyFMReader

Archived source code at the time of publication: https://doi.org/10.5281/zenodo.8342163 (López-Alonso, 2023a)

Source code available from: https://github.com/jlopezalo/PyFMReader

License: MIT

PyFMRheo

Archived source code at the time of publication: https://doi.org/10.5281/zenodo.8342163 (López-Alonso, 2023a)

Source code available from: https://github.com/jlopezalo/PyFMRheo

License: MIT

PyFMGUI

Archived source code at the time of publication: https://doi.org/10.5281/zenodo.8342163 (López-Alonso, 2023a)

Source code available from: https://github.com/jlopezalo/PyFMGUI

License: MIT

Acknowledgements

We would like to thank Dr. N. Barois from the BICeL imaging facility (PLBS UAR 2014 - US 41) for the acquisition of the SEM probe images., Dr. P. Muller for useful discussions and Dr. C. Valotteau for critical reading of the manuscript.

Appendix 1 – Tutorials

In this Appendix we provide three examples on how to customize and expand the capabilities of the PyFMRheo library. In the first example we will show how the Hertz model can be expanded to allow for the computation of stiffness (the slope of the force-indentation curve). The second example will show how the corrections module can be expanded to include the approximation of the spherical Sneddon solution to pyFMRheo (Kontomaris & Malamou, 2021). Finally, the third example will show how to implement a model and routine from scratch.

The library is structured in three main blocks:

- Models: Contains all the implemented models, geometric coefficients and corrections.

- Utils: Contains the generic functions used to preprocess force curves and DMA/microrheology measurements.
- Routines: Contains predefined routines that allow to process single force curves loaded using the PyFMReader.

Tutorial 1: Implementation of stiffness calculation

In this tutorial we will showcase how to adapt new models from already existing ones in the library. We will go over the steps needed to implement the stiffness computation based on the Hertz model implementation.

Implementation strategy

The function fitted to the Hertz model is defined as:

$$F = Geometriccoefficient * E_0 * Indentation^{GeometricExponent} + OffsetForce$$

Therefore, by substituting the geometric coefficient, correction coefficient and geometric exponent by 1 the equation will take the form of:

$$F = E_0 * Indentation + OffsetForce$$

Thus, E0 in this case defines the sample's stiffness (N/m) instead of the Young's modulus. This is now implemented in the repository files.

Modified files

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/models/correction_factors.py

Steps

1. Within the HertzModel class the model method is the function used in the minimization process:

```
def model(self, indentation, delta0, E0, f0, slope=None, sample_height=None):
  # Define output array
  force = np.zeros(indentation.shape)
  # Find the index where indentation is 0
  idx = (np.abs(indentation - delta0)).argmin()
  # Get the value of the contact point
  delta0 = indentation[idx]
  # Get indenter shape coefficient and exponent
  coeff, n = get_coeff(self.ind_geom, self.tip_parameter, self.poisson_ratio)
  # Get bottom effect correction coefficients
  if self.bec_model and sample_height:
      bec_coeffs = self.get_bec_coeffs(sample_height, indentation)
  else:
      bec_coeffs = np.ones(indentation.shape)
  # Compute the force using hertz model
  for i in range(len(force)):
      if indentation[i] < delta0:
          if self.fit_hline_flag:
              # Fit a line on the non contact part
              force[i] = (indentation[i] - delta0) * slope + f0
          else:
              # Assign f0 as force value
              force[i] = f0
      else:
          # Fit Hertz model on the contact part
          # F = F0 * BEC_Correction
          force[i] = coeff * bec_coeffs[i] * E0 * np.power((indentation[i] - delta0), n) + f0
  return force
```

2. The function get_coeff is the function used to obtain the geometric coefficient and exponent based on the indenter shape:

```
def get_coeff(ind_geom,  tip_parameter, poisson_ratio):
    if ind_geom == 'paraboloid':
        coeff = 4/3 * np.sqrt(tip_parameter) * 1/(1-poisson_ratio**2)
        n = 3 / 2
    elif ind_geom == 'pyramid':
        coeff = 1/np.sqrt(2) * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'blunted_pyramid':
        coeff = 1/np.sqrt(2) * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'cone':
        coeff = 2/np.pi * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'flat_punch':
        coeff = 2 * tip_parameter * 1/(1-poisson_ratio**2)
        n = 1
    else:
        raise Exception('Non valid indenter geometry')
    return coeff, n
```

3. For implementing stiffness, we must add another condition to the get_coeff function to return 1 both for the geometric coefficient (coeff) and exponent (n):

```
def get_coeff(ind_geom,  tip_parameter, poisson_ratio):
    if ind_geom == 'paraboloid':
        coeff = 4/3 * np.sqrt(tip_parameter) * 1/(1-poisson_ratio**2)
        n = 3 / 2
    elif ind_geom == 'pyramid':
        coeff = 1/np.sqrt(2) * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'blunted_pyramid':
        coeff = 1/np.sqrt(2) * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'cone':
        coeff = 2/np.pi * np.tan(np.radians(tip_parameter)) * 1/(1-poisson_ratio**2)
        n = 2
    elif ind_geom == 'flat_punch':
        coeff = 2 * tip_parameter * 1/(1-poisson_ratio**2)
        n = 1
    elif ind_geom == 'stiffness':
        coeff = 1
        n = 1
    else:
        raise Exception('Non valid indenter geometry')
    return coeff, n
```

Tutorial 2: Implementation of the Kontomaris approximation

This tutorial showcases the steps necessary to include correction factors to the Hertz model based on the approximation of a spherical indenter in Kontomaris & Malamou, 2021. This correction will be then called and applied to the simple Hertz model if this approximation is used.

Implementation strategy

The function fitted to the Hertz model is defined as:

$$F = Geometriccoefficient * E * Indentation^{GeometricExponent} + OffsetForce$$

Therefore, we have to provide the series of correction coefficients that will be used.

Modified files

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/models/correction_factors.py

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/models/hertz.py

Steps

1. The first step is to implement the function to compute the Kontomaris correction in the pyfmrheo/models/correction_factors.py module:

```
# Kontomaris 2021 EPJ: approximation to spherical indenter
# Source: https://doi.org/10.1088/1361-6404/abccfb
# Equation 17

# c1 = 1.0100000
# c2 = -0.0730300
# c3 = -0.1357000
# c4 = 0.0359800
# c5 = -0.0040240
# c6 = 0.0001653

kontomaris_model_factors = [
        lambda _, R : 2/3 * 1.0100000 / np.sqrt(R),
# Order 1
        lambda indentation, R : 1/2 * -0.0730300 * np.sqrt(indentation) / R ,
# Order 2
        lambda indentation, R : 1/3 * -0.1357000 * indentation * np.sqrt(indentation) / R ** 2 ,
# Order 3
        lambda indentation, R : 1/4 * 0.0359800 * indentation ** 2 * np.sqrt(indentation) / R ** 3,
# Order 4
        lambda indentation, R : 1/5 * -0.0040240 * indentation ** 3 * np.sqrt(indentation) / R ** 4,
# Order 5
        lambda indentation, R : 1/6 * 0.0001653 * indentation ** 4 * np.sqrt(indentation) / R ** 5
# Order 6
]

def sphere_approx_kontomaris(_, indentation, ind_shape, R, order=6):
    coefficients_out = []
    model_factors = kontomaris_model_factors
    if ind_shape != "paraboloid":
        raise Exception(f"The Kontomaris approximation is only suitable for the paraboloid geometry.")
    for i in range(len(indentation)):
        # The correction factor is computed as:
        # coef = O0 + O1 + On...
        # Order 0 coefficient common in all models
        try:
            coeff = 0.0
            for j in range(order):
                O = model_factors[j]
                coeff += 3/2 * np.sqrt(R) * O(indentation[i], R)
            coefficients_out.append(coeff)
        except Exception as _:
            # Handle exceptions to avoid correction factor array and indentation array
            # having different shapes.
            traceback.print_exc()
            coefficients_out.append(1.0)

    return coefficients_out
```

The correction_factors.py module contains functions that compute a series of values for each point of the force curve that depend on the correction coefficients for each model, on the indentation and other model specific parameters (e.g., Sample height / thickness).

2. Within the HertzModel class the model method is the function used in the minimization process:

```
def model(self, indentation, delta0, E0, f0, slope=None, sample_height=None):
  # Define output array
  force = np.zeros(indentation.shape)
  # Find the index where indentation is 0
  idx = (np.abs(indentation - delta0)).argmin()
  # Get the value of the contact point
  delta0 = indentation[idx]
  # Get indenter shape coefficient and exponent
  coeff, n = get_coeff(self.ind_geom, self.tip_parameter, self.poisson_ratio)
  # Get bottom effect correction coefficients
  if self.bec_model and sample_height:
      bec_coeffs = self.get_correction_coeffs(sample_height, indentation)
  else:
      bec_coeffs = np.ones(indentation.shape)
  # Compute the force using hertz model
  for i in range(len(force)):
      if indentation[i] < delta0:
          if self.fit_hline_flag:
              # Fit a line on the non contact part
              force[i] = (indentation[i] - delta0) * slope + f0
          else:
              # Assign f0 as force value
              force[i] = f0
```

```
    else:
        # Fit Hertz model on the contact part
        # F = F0 * BEC_Correction
        force[i] = coeff * bec_coeffs[i] * E0 * np.power((indentation[i] - delta0), n) + f0
return force
```

3. Within the model function, the get_correction_coeffs method is used to compute the correction coefficient if a correction model and sample height is provided. This function was implemented following this approach because all the corrections previously implemented were bottom effect corrections, where the height of the sample is needed to compute the correction coefficient. As the Kontomaris correction does not require the height of the sample, the model class must be modified to not require the sample height when the Kontomaris correction is applied. This can easily be achieved by including another conditional clause:

```
# Get bottom effect correction coefficients
if self.correction_model and self.correction_model == 'kontomaris':
    correction_coeffs = self.get_correction_coeffs(None, indentation)
elif self.correction_model and sample_height:
    correction_coeffs = self.get_correction_coeffs(sample_height, indentation)
else:
    correction_coeffs = np.ones(indentation.shape)
```

4. Finally, we have to modify the get_correction_coeffs method to return the Kontomaris correction values for each point in the curve:

```
def get_correction_coeffs(self, sample_height, indentation):
    correction_params = [sample_height, (indentation - self.delta0), self.ind_geom, self.tip_parameter]
    if self.correction_model == 'dimitriadis_paraboloid_bonded':
        return bec_dimitriadis_paraboloid_bonded(*correction_params)
    elif self.correction_model == 'dimitriadis_paraboloid_not_bonded':
        return bec_dimitriadis_paraboloid_not_bonded(*correction_params)
    elif self.correction_model == 'gavara_cone':
        return bec_gavara_cone(*correction_params)
    elif self.correction_model == 'managuli_cone':
        return bec_managuli_cone(*correction_params)
    elif self.correction_model == 'garcia_garcia':
        return bec_garcia_garcia(*correction_params)
    elif self.correction_model == 'kontomaris':
        return sphere_approx_kontomaris(*correction_params)
    else:
        # TO DO: Implement custom exception
        raise Exception('Correction model not implemented')
```

Tutorial 3: Implementation of the DMT fit routine

This tutorial showcases how to implement a new model and routine into the library. As an example, the DMT model has been implemented.

Implementation strategy

The Derjaguin-Muller-Toporov (DMT) model is an extension of the Hertz model that considers the adhesion forces between the samples and the cantilever tip. This model is most suitable for modeling the contact between relatively hard materials with low adhesion.

The DMT model is described as:

$$F = \begin{cases} F_{Hertz}(z - d_0) + F_{Adhesion}(d_0), z \le d_0 \\ F_{Adhesion}(z), z > d_0 \end{cases}$$

Where F_{Hertz} $(z–d_0)$ is the force obtained by applying the Hertz model and $F_{Adhesion}$ (d_0) is the adhesion force according to DMT theory. However, for its implementation this model can be further simplified by assuming a constant adhesion force that can be directly measured for the measured force – indentation curve. If adhesion forces are present a jump in the force can be observed on the retract curve. Therefore $F_{Adhesion}$ can be assumed to be equal to the value of this jump.

$$F = \begin{cases} F_{Hertz}(z - d_0) + F_{Adhesion}, z \le d_0 \\ 0.0, z > d_0 \end{cases}$$

Modified files

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/models/dmt.py

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/models/__init__.py

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/routines/DMTFit.py

https://github.com/jlopezalo/PyFMRheo/blob/main/src/pyfmrheo/routines/__init__.py

Steps

1. The first step is to create a new file in the models folder with the name of our model. In this case models\dmt.py
2. As the DMT model is based on the Hertz model we can copy paste the HertzModel class from the file models\Hertz.py and modify it.
3. All models should include 9 main functions:
    a. __init__: Initializes all the model properties with the default values and the values provided by the user when initializing the model class.
    b. build_params: Creates a Parameters class from the Lmfit library that will be used during the minimization process.
    c. model: Function that defines the model. In this case the function returns a force vector computed according to the simplified DMT model.
    d. fit: Function that performs the minimization using the experimental data provided by the user.
    e. eval: Evaluates the model using the experimental data provided by the user and the parameters obtained during the minimization process.
    f. get_residuals: Computes the residuals.
    g. get_chisq: Computes the χ^2 parameter used to assess the goodness-of-fit.
    h. get_red_chisq: Computes the reduced χ^2 by dividing the χ^2 by the total amount of free parameters during the fit.
    i. fit_report: Prints a report showing the parameters used and result of the fit.

Following this structure the code for the DMTModel class implemented is the following:

```
import numpy as np
from lmfit import Model, Parameters

from .geom_coeffs import get_coeff

class DMTModel:
    # Model abstract class
    def __init__(self, ind_geom, tip_param) -> None:
        # Tip geomtry params
        self.ind_geom = ind_geom         # No units
        self.tip_parameter = tip_param   # If radius units is meters, If half angle units is degrees
        # Model params ######################
        self.n_params = None
        # Adheshion force
        self.adhesion_force = 0.0
        # Contact point
        self.delta0 = 0.0
        # Apparent Young's Modulus
        self.E0 = 1000
        self.E0_init = 1000
        self.E0_min = 0
        self.E0_max = np.inf
        # Poisson ratio
        self.poisson_ratio = 0.5
        # Sample height
        self.sample_height = None
        # Goodness of fit metrics
        self.MAE = None
        self.SE = None
        self.MSE = None
        self.RMSE = None
        self.Rsquared = None
        self.chisq = None
        self.redchi = None
```

```
def build_params(self):
    params = Parameters()
    params.add('E0', value=self.E0_init, min=self.E0_min, max=self.E0_max)
    return params

def model(self, indentation, E0, delta0):
    # Define output array
    force = np.zeros(indentation.shape)
    # Find the index where indentation is 0
    idx = (np.abs(indentation - delta0)).argmin()
    # Get the value of the contact point
    delta0 = indentation[idx]
    # Get indenter shape coefficient and exponent
    coeff, n = get_coeff(self.ind_geom, self.tip_parameter, self.poisson_ratio)
    # Compute the force using hertz model
    for i in range(len(force)):
        if indentation[i] < delta0:
            # Assign 0.0 N as force value
            force[i] = 0.0
        else:
            # Fit Hertz model on the contact part
            # F = FHertz + FAdhesion
            force[i] = coeff * E0 * np.power((indentation[i] - delta0), n) + self.adhesion_force
    return force

def fit(self, indentation, force):
    coeff, n = get_coeff(self.ind_geom, self.tip_parameter, self.poisson_ratio)
    self.E0_init = np.max(force) / coeff / np.max(indentation) ** n
    # Param order:
    # E0, delta0
    DMTmodel =\
        lambda indentation, E0: self.model(indentation, E0, self.delta0)

    DMTmodelfit = Model(DMTmodel)

    # Define free params
    params = self.build_params()

    # Do fit
    self.n_params = len(DMTmodelfit.param_names)
    result_hertz = DMTmodelfit.fit(force, params, indentation=indentation)

    # Assign fit results to model params
    self.E0 = result_hertz.best_values['E0']

    modelPredictions = self.eval(indentation)

    absError = modelPredictions - force

    self.MAE = np.mean(absError) # mean absolute error
    self.SE = np.square(absError) # squared errors
    self.MSE = np.mean(self.SE) # mean squared errors
    self.RMSE = np.sqrt(self.MSE) # Root Mean Squared Error, RMSE
```

4. Once the model class has been implemented the following line should be added to the models__init__.py file:

```
from .dmt import *
```

This line ensures that the contents of the models\dmt.py file are imported when the PyFMRheo library is imported.

5. At this point the DMTModel class is fully implemented and can be accessed. However, to facilitate the use of the model classes, it is a good idea to include an example routine in the routines folder. These routines offer a predefined function for processing a PyFMReader FDC (Force Distance Curve) object and a dictionary of parameters (param_dict). To implement the DMT routine first we create the file routines\DMTFit.py and then in this file we define a function (doDMTFit) that takes as arguments a FDC object and the parameter dictionary. Then within this curve we define the steps necessary for fitting the DMT model.

There is not a strict structure on how to implement a routine in PyFMRheo to provide as much flexibility as possible. However generally the routine should include the following steps:

- Data preprocessing: First the raw data from the force distance curve should be prepared for analysis. This may include the determination of an initial estimation for the contact point, signal detrending, baseline correction, tilt correction, etc.
- Model class initialization: Once the data has been preprocessed, the model class should be initialized, and all the properties needed for the minimization / evaluation should be given a default value.
- Model minimization: This step is achieved by calling the fit method from the model class.

The doDMTFit function has been implemented as follows:

```
import numpy as np

from ..utils.force_curves import correct_tilt, correct_offset, find_nearest
from ..models.dmt import DMTModel

def doDMTFit(fdc, param_dict):
    # Get segment data
    if param_dict['curve_seg'] == 'extend':
        raise Exception("DMT can only be fit to the retract segment of the curve!")
    else:
        segment_data = fdc.retract_segments[-1][1]
        segment_data.zheight = segment_data.zheight[::-1]
        segment_data.vdeflection = segment_data.vdeflection[::-1]
    # Perform tilt correction
    if param_dict['offset_type'] == 'percentage':
        deltaz = segment_data.zheight.max() - segment_data.zheight.min()
        maxoffset = segment_data.zheight.min() + deltaz * param_dict['max_offset']
        minoffset = segment_data.zheight.min() + deltaz * param_dict['min_offset']
    else:
        maxoffset = param_dict['max_offset']
        minoffset = param_dict['min_offset']

    if param_dict['correct_tilt']:
        segment_data.vdeflection =\
            correct_tilt(
                segment_data.zheight, segment_data.vdeflection, maxoffset, minoffset
            )
    else:
        segment_data.vdeflection =\
            correct_offset(
                segment_data.zheight, segment_data.vdeflection, maxoffset, minoffset
            )
    # Downsample signal
    if param_dict['downsample_flag']:
        downfactor= len(segment_data.zheight) // param_dict['pts_downsample']
        idxDown = list(range(0, len(segment_data.zheight), downfactor))
        segment_data.zheight = segment_data.zheight[idxDown]
        segment_data.vdeflection = segment_data.vdeflection[idxDown]
    # Get initial estimate of PoC
    segment_data.force =  np.array(segment_data.vdeflection * param_dict['k'])
    force = segment_data.force - force[0]
    if param_dict['adhesionForce'] is None:
        adhesionForce = np.min(force)
    else:
        adhesionForce = param_dict['adhesionForce']
    tipPosition = np.array(segment_data.zheight - segment_data.vdeflection)
    comp_PoC_idx = find_nearest(force, adhesionForce)
    comp_PoC = tipPosition[comp_PoC_idx]
    # Prepare data for the fit
    indentation = tipPosition - comp_PoC
    contact_mask = force >= adhesionForce
    ncont_ind = indentation[~contact_mask]
    cont_ind = indentation[contact_mask]
    ncont_force = force[~contact_mask]
    cont_force = force[contact_mask]
    if param_dict['fit_range_type'] == 'indentation':
        mask = (cont_ind >= param_dict['min_ind']) & (cont_ind <= param_dict['max_ind'])
        cont_ind, cont_force = cont_ind[mask], cont_force[mask]
    elif param_dict['fit_range_type'] == 'force':
        mask = (cont_force >= param_dict['min_force']) & (cont_force <= param_dict['max_force'])
        cont_ind, cont_force = cont_ind[mask], cont_force[mask]
    indentation = np.r_[ncont_ind, cont_ind]
    force = np.r_[ncont_force, cont_force]
    # Perform fit
    dmt_model = DMTModel(param_dict['contact_model'], param_dict['tip_param'])
    dmt_model.delta0 = comp_PoC[0]
    dmt_model.adhesion_force = adhesionForce
    if not param_dict['auto_init_E0']:
        dmt_model.E0_init = param_dict['E0']
    dmt_model.fit(indentation, force)
    # Return fitted model object
    return dmt_model
```

6. Finally, for the same purpose as in step 4, we include the following line in the routines__init__.py file:

```
from .DMTFit import *
```

References

Alcaraz J, Buscemi L, Grabulosa M, *et al.*: **Microrheology of human lung epithelial cells measured by Atomic Force Microscopy.** *Biophys J.* 2003; **84**(3): 2071–2079.
PubMed Abstract | Publisher Full Text | Free Full Text

Alcaraz J, Buscemi L, Puig-de-Morales M, *et al.*: **Correction of microrheological measurements of soft samples with Atomic Force Microscopy for the hydrodynamic drag on the cantilever.** *Langmuir.* 2002; **18**(3): 716–721.
Publisher Full Text

Balland M, Desprat N, Icard D, *et al.*: **Power laws in microrheology experiments on living cells: comparative analysis and modeling.** *Phys Rev E Stat Nonlin Soft Matter Phys.* 2006; **74**(2 Pt 1): 021911.
PubMed Abstract | Publisher Full Text

Barber JR, Billings DA: **An approximate solution for the contact area and elastic compliance of a smooth punch of arbitrary shape.** *Int J Mech Sci.* 1990; **32**(12): 991–997.
Publisher Full Text

Beton JG, Moorehead R, Helfmann L, *et al.*: **TopoStats - a program for automated tracing of biomolecules from AFM images.** *Methods.* 2021; **193**: 68–79.
PubMed Abstract | Publisher Full Text | Free Full Text

Bilodeau GG: **Regular pyramid punch problem.** *J Appl Mech.* 1992; **59**(3): 519–523.
Publisher Full Text

Brenner H: **The slow motion of a sphere through a viscous fluid towards a plane surface.** *Chem Eng Sci.* 1961; **16**(3–4): 242–251.
Publisher Full Text

Brückner BR, Nöding H, Janshoff A: **Viscoelastic properties of confluent MDCK II cells obtained from force cycle experiments.** *Biophys J.* 2017; **112**(4): 724–735.
PubMed Abstract | Publisher Full Text | Free Full Text

Carl P, Schillers H: **Elasticity measurement of living cells with an Atomic Force Microscope: data acquisition and processing.** *Pflugers Arch.* 2008; **457**(2): 551–559.
PubMed Abstract | Publisher Full Text

Chen SW, Teulon JM, Kaur H, *et al.*: **Nano-structural stiffness measure for soft biomaterials of heterogeneous elasticity.** *Nanoscale Horiz.* 2022; **8**(1): 75–82.
PubMed Abstract | Publisher Full Text

Chighizola M, Rodriguez-Ramos J, Rico F, *et al.*: **3.1.3 AFM calibration issues**. In: *3.1.3 AFM Calibration Issues*. De Gruyter, 2023; 105–128.
Publisher Full Text

Cox RG, Brenner H: **The slow motion of a sphere through a viscous fluid towards a plane surface—II small gap widths, including inertial effects.** *Chem Eng Sci.* 1967; **22**(12): 1753–1777.
Publisher Full Text

Derjaguin BV, Muller VM, Toporov YP: **Effect of contact deformations on the adhesion of particles.** *J Colloid Interface Sci.* 1975; **53**(2): 314–326.
Publisher Full Text

Domke J, Radmacher M: **Measuring the elastic properties of thin polymer films with the Atomic Force Microscope.** *Langmuir.* 1998; **14**(12): 3320–3325.
Publisher Full Text

Efremov YM, Kotova SL, Timashev PS: **Viscoelasticity in simple indentation-cycle experiments: a computational study.** *Sci Rep.* 2020; **10**(1): 13302.
PubMed Abstract | Publisher Full Text | Free Full Text

Efremov YM, Wang WH, Hardy SD, *et al.*: **Measuring nanoscale viscoelastic parameters of cells directly from AFM force-displacement curves.** *Sci Rep.* 2017; **7**(1): 1541.
PubMed Abstract | Publisher Full Text | Free Full Text

Elosegui-Artola A, Gupta A, Najibi AJ, *et al.*: **Matrix viscoelasticity controls spatiotemporal tissue organization.** *Nat Mater.* 2023; **22**(1): 117–127.
PubMed Abstract | Publisher Full Text | Free Full Text

Engler AJ, Sen S, Sweeney HL, *et al.*: **Matrix elasticity directs stem cell lineage specification.** *Cell.* 2006; **126**(4): 677–689.
PubMed Abstract | Publisher Full Text

Eroles M, Lopez-Alonso J, Ortega A, *et al.*: **Coupled mechanical mapping and Interference Contrast Microscopy reveal viscoelastic and adhesion hallmarks of monocyte differentiation into macrophages.** *Nanoscale.* 2023; **15**(29): 12255–12269.
PubMed Abstract | Publisher Full Text

Fabry B, Maksym GN, Butler JP, *et al.*: **Scaling the microrheology of living cells.** *Phys Rev Lett.* 2001; **87**(14): 148102.
PubMed Abstract | Publisher Full Text

Flormann DAD, Anton C, Pohland MO, *et al.*: **Oscillatory microrheology, creep compliance and stress relaxation of biological cells reveal strong correlations as probed by Atomic Force Microscopy.** *Front Phys.* 2021; **9**.
Publisher Full Text

Garcia PD, Garcia R: **Determination of the viscoelastic properties of a single cell cultured on a rigid support by force microscopy.** *Nanoscale.* 2018; **10**(42): 19799–19809.
PubMed Abstract | Publisher Full Text

Garcia PD, Guerrero CR, Garcia R: **Nanorheology of living cells measured by AFM-based force-distance curves.** *Nanoscale.* 2020; **12**(16): 9133–9143.
PubMed Abstract | Publisher Full Text

Gavara N: **Combined strategies for optimal detection of the contact point in AFM force-indentation curves obtained on thin samples and adherent cells.** *Sci Rep.* 2016; **6**: 21267.
PubMed Abstract | Publisher Full Text | Free Full Text

Gerum R, Mirzahossein E, Eroles M, *et al.*: **Viscoelastic properties of suspended cells measured with shear flow deformation cytometry.** *eLife.* 2022; **11**: e78823.
PubMed Abstract | Publisher Full Text | Free Full Text

Graham GAC: **The contact problem in the linear theory of viscoelasticity when the time dependent contact area has any number of maxima and minima.** *Int J Eng Sci.* 1967; **5**(6): 495–514.
Publisher Full Text

Harris CR, Millman KJ, van der Walt SJ, *et al.*: **Array programming with NumPy.** *Nature.* 2020; **585**(7825): 357–362.
PubMed Abstract | Publisher Full Text | Free Full Text

Hermanowicz P, Sarna M, Burda K, *et al.*: **AtomicJ: an open source software for analysis of force curves.** *Rev Sci Instrum.* 2014; **85**(6): 063703.
PubMed Abstract | Publisher Full Text

Hertz H: **Ueber die Berührung fester elastischer Körper.** In: A. L. Crelle, C. W. Borchardt, and Schellbach (Eds.), *Band 92.* De Gruyter, 1882; 156–171.
Publisher Full Text

Higgins MJ, Proksch R, Sader JE, *et al.*: **Noninvasive determination of optical lever sensitivity in atomic force microscopy.** *Rev Sci Instrum.* 2006; **77**(1): 013701.
Publisher Full Text

Jorba I, Beltrán G, Falcones B, *et al.*: **Nonlinear elasticity of the lung extracellular microenvironment is regulated by macroscale tissue strain.** *Acta Biomater.* 2019; **92**: 265–276.
PubMed Abstract | Publisher Full Text | Free Full Text

Jorba I, Uriarte JJ, Campillo N, *et al.*: **Probing micromechanical properties of the extracellular matrix of soft tissues by Atomic Force Microscopy.** *J Cell Physiol.* 2017; **232**(1): 19–26.
PubMed Abstract | Publisher Full Text

Kaplan JL, Bonfanti A, Kabla AJ: **RHEOS.jl—A julia package for rheology data analysis.** *J Open Source Softw.* 2019; **4**(41): 1700.
Publisher Full Text

Kontomaris SV, Malamou A: **A novel approximate method to calculate the force applied on an elastic half space by a rigid sphere.** *Eur J Phys.* 2021; **42**(2): 025010.
Publisher Full Text

Kontomaris SV, Stylianou A, Chliveros G, *et al.*: **Determining spatial variability of elastic properties for biological samples using AFM.** *Micromachines (Basel).* 2023; **14**(1): 182.
PubMed Abstract | Publisher Full Text | Free Full Text

Lee EH, Radok JRM: **The contact problem for viscoelastic bodies.** *J Appl Mech.* 1960; **27**(3): 438–444.
Publisher Full Text

Lekka M, Laidler P, Gil D, *et al.*: **Elasticity of normal and cancerous human bladder cells studied by scanning force microscopy.** *Eur Biophys J.* 1999; **28**(4): 312–316.
PubMed Abstract | Publisher Full Text

Lekka M, Navajas D, Radmacher M, *et al.*: **Volume 1 biomedical methods.** De Gruyter, 2023a.
Publisher Full Text

Lekka M, Navajas D, Radmacher M, *et al.*: **Volume 2 biomedical applications.** De Gruyter, 2023b.
Publisher Full Text

López-Alonso J: **PyFMLab (1.0.2).** [Source code] Zenodo. 2023a.
http://www.doi.org/10.5281/zenodo.8342163

López-Alonso J: **AFM Dataset on HeLa cells**. Zenodo. [Data set], 2023b.
http://www.doi.org/10.5281/zenodo.8342121

Love AEH: **Boussinesq's problem for a rigid conE.** *Q J Math.* 1939; **os-10**(1): 161–175.
Publisher Full Text

Mahaffy RE, Shih CK, MacKintosh FC, *et al.*: **Scanning Probe-based frequency-dependent microrheology of polymer gels and biological cells.** *Phys Rev Lett.* American Physical Society (APS), 2000; **85**(4): 880–883.
PubMed Abstract | Publisher Full Text

McKinney W: **Data structures for statistical computing in python.** In: S. van der Walt and J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference.* 2010; 56–61.
Publisher Full Text

Müller P, Abuhattum S, Möllmert S, *et al.*: **nanite: Using machine learning to assess the quality of atomic force microscopy-enabled nano-indentation data.** *BMC Bioinformatics.* 2019; **20**(1): 465.
PubMed Abstract | Publisher Full Text | Free Full Text

Nečas D, Klapetek P: **Gwyddion: an open-source software for SPM data analysis.** *Open Phys.* 2012; **10**(1): 181–188.
Publisher Full Text

Newville M, Otten R, Nelson A, *et al.*: **lmfit/lmfit-py: 1.2.0.** *Zenodo.* 2023.
Publisher Full Text

Pirzer T, Hugel T: **Atomic force microscopy spring constant determination in viscous liquids.** *Rev Sci Instrum.* 2009; **80**(3): 035110.
PubMed Abstract | **Publisher Full Text**

Rianna C, Radmacher M, Kumar S: **Direct evidence that tumor cells soften when navigating confined spaces.** *Mol Biol Cell.* 2020; **31**(16): 1726–1734.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Rico F, Roca-Cusachs P, Gavara N, *et al.*: **Probing mechanical properties of living cells by atomic force microscopy with blunted pyramidal cantilever tips.** *Phys Rev E Stat Nonlin Soft Matter Phys.* 2005; **72**(2 Pt 1): 021914.
PubMed Abstract | **Publisher Full Text**

Rigato A, Miyagi A, Scheuring S, *et al.*: **High-frequency microrheology reveals cytoskeleton dynamics in living cells.** *Nat Phys.* 2017; **13**(8): 771–775.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Rother J, Nöding H, Mey I, *et al.*: **Atomic force microscopy-based microrheology reveals significant differences in the viscoelastic response between malign and benign cell lines.** *Open Biol.* 2014; **4**(5): 140046.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Sader JE, Borgani R, Gibson CT, *et al.*: **A virtual instrument to standardise the calibration of Atomic Force Microscope cantilevers.** *Rev Sci Instrum.* 2016; **87**(9): 093711.
PubMed Abstract | **Publisher Full Text**

Sader JE, Chon JWM, Mulvaney P: **Calibration of rectangular Atomic Force Microscope cantilevers.** *Rev Sci Instrum.* 1999; **70**(10): 3967–3969.
Publisher Full Text

Sader JE, Pacifico J, Green CP, *et al.*: **General scaling law for stiffness measurement of small bodies with applications to the Atomic Force Microscope.** *J Appl Phys.* 2005; **97**(12): 124903.
Publisher Full Text

Sader JE, Sanelli JA, Adamson BD, *et al.*: **Spring constant calibration of Atomic Force Microscope cantilevers of arbitrary shape.** *Rev Sci Instrum.* 2012; **83**(10): 103705.
PubMed Abstract | **Publisher Full Text**

Sanchez JG, Espinosa FM, Miguez R, *et al.*: **The viscoelasticity of adherent cells follows a single power-law with distinct local variations within a single cell and across cell lines.** *Nanoscale.* 2021; **13**(38): 16339–16348.
PubMed Abstract | **Publisher Full Text**

Schillers H, Rianna C, Schäpe J, *et al.*: **Standardized nanomechanical Atomic Force Microscopy Procedure (SNAP) for measuring soft and biological samples.** *Sci Rep.* 2017; **7**(1): 5117.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Sneddon IN: **The relation between load and penetration in the axisymmetric boussinesq problem for a punch of arbitrary profile.** *Int J Eng Sci.* 1965; **3**(1): 47–57.
Publisher Full Text

Stark RW, Drobek T, Heckl WM: **Thermomechanical noise of a free v-shaped cantilever for Atomic-Force Microscopy.** *Ultramicroscopy.* 2001; **86**(1–2): 207–215.
PubMed Abstract | **Publisher Full Text**

Sumbul F, Hassanpour N, Rodriguez-Ramos J, *et al.*: **One-Step calibration of AFM in liquid.** *Front Phys.* 2020; **8**.
Publisher Full Text

Ting TCT: **The contact stresses between a rigid indenter and a viscoelastic half-space.** *J Appl Mech.* 1966; **33**(4): 845–854.
Publisher Full Text

Ting TCT: **Contact problems in the linear theory of viscoelasticity.** *J Appl Mech.* 1968; **35**(2): 248–254.
Publisher Full Text

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

Yango A, Schäpe J, Rianna C, *et al.*: **Measuring the viscoelastic creep of soft samples by step response AFM.** *Soft Matter.* 2016; **12**(40): 8297–8306.
PubMed Abstract | **Publisher Full Text**

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 05 August 2024

https://doi.org/10.21956/openreseurope.19086.r42420

© **2024 Kontomaris S.** 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.

✔ **Stylianos Vasileios Kontomaris**
Metropolitan College, Athens, Greece

I congratulate the authors on this very useful paper for the entire AFM community. I have no further comments.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: AFM mechanical characterization, mathematical modelling

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.

Reviewer Report 24 July 2024

https://doi.org/10.21956/openreseurope.19086.r42421

© **2024 Li M.** 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.

✔ **Mi Li**
Shenyang Institute of Automation Chinese Academy of Sciences, Beijing, China

My concerns have been well addressed and I have no further comments.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: atomic force microscopy, mechanobiology

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.

Version 1

Reviewer Report 08 December 2023

https://doi.org/10.21956/openreseurope.17865.r36122

© 2023 Kontomaris S. 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.

?

Stylianos Vasileios Kontomaris
[1] Metropolitan College, Athens, Greece
[2] Metropolitan College, Athens, Greece

This is a very important work in the field of AFM data processing regarding biological samples. As correctly indicated by the authors, there is currently no standardized method or software that can efficiently process all the data obtained from AFM. For example, powerful software packages (e.g. the AtomicJ software provide accurate results but do not provide all the information mentioned here, i.e. do not allow the processing of microrheology data).This is a significant drawback in AFM research, as many researchers are compelled to build home-made algorithms, which is a time-consuming procedure. In many cases, these algorithms lack accuracy. Therefore, this work could represent a significant step forward in streamlining data processing for AFM experiments.

I have some questions/suggestions provided below:

1) In Table 1, there are the standard elastic models for paraboloid, conical and 4 sided regular pyramids. The Hertz equation for parabolic indenters can also be applied to spherical indenters for small indentation depths (e.g., if the indentation depth h is h < R/10, where R is the indenter's radius). However, in many cases, spherical indenters with a radius in the range of a few micrometers are used for data processing. In this case, for large indentation depths, the force-indentation data do not follow the first equation in Table 1. The same restriction also applies to sphero-conical indenters. Therefore, in my opinion, it should be emphasized in the article that these cases will be included in future versions of the software.

2) I have been using AtomicJ software for several years, and it is a powerful tool for acquiring Young's modulus. The authors clearly mentioned that both AtomicJ and the new software (PyFMLab) led to similar results. In my opinion, this is important. However, I believe that the percentage difference in Young's modulus values between AtomicJ and PyFMLab should be added for comparison. It is significant to determine if the percentage difference between these two is 4%, 17%, or something similar.

3) In the AFM measurements section, the authors explain that, for measuring the mechanical

properties of cells, maps of force curves were acquired on the nucleus area of 20 cells. These maps had a size of 30 μm x 30 μm and a resolution of 4 × 4 pixels. I wonder why didn't they use a standard map with 16 x 16 pixels (or even a map with 64 x 64 pixels) to provide a more 'detailed' mechanical characterization? If there is a specific reason for this choice, it should be clearly mentioned within the manuscript.

4) The 4-17% difference between different software is explained based on the accuracy of the contact point determination. It is widely known that if we 'miss' the correct contact point, we can end up with a significant error, even around 100%. However, my question is whether this is the only source of these differences. If there are also other sources, they should be mentioned and discussed.

5) In addition, the authors used HeLa cells. I wonder why they did not first validate the accuracy of their software using an approximately homogeneous and isotropic material, such as an agarose gel. The reason for this question is that I wonder if, in this case, the percentage difference between different software is minimized.

6) Another point to mention is Figure 3. In this case the Hertz model fits almost perfectly the data (it seems that the R-squared coefficient is close to 1 in this case). My question is if all the tested data 'perfectly followed' a model of contact mechanics. As we know there are cases of force indentation curves in which the data only approximately follow the relative model. Therefore, what is the average R-squared coefficient in the tested data? Additionally, do you believe that in cases with low R-squared coefficients, the differences when using different algorithms will be affected or not?

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?

Yes

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: AFM mechanical characterization, mathematical modelling

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 27 November 2023

https://doi.org/10.21956/openreseurope.17865.r36129

© **2023 Li M.** 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.

?

Mi Li
[1] Shenyang Institute of Automation Chinese Academy of Sciences, Beijing, China
[2] Shenyang Institute of Automation Chinese Academy of Sciences, Beijing, China

This paper presents an open-source software named PyFMLab for the analysis of experimental data obtained during atomic force microscopy (AFM) mechanical assay. Various mechanical properties can be obtained by the software, including elasticity, viscoelasticity, and frequency-dependent microrheological mechanics. AFM has now become an important and standard tool for characterizing the mechanical properties of living biological specimens, but so far, nanomechanical studies with the use of AFM still mainly focus on characterization of single mechanical property such as elasticity. Cellular mechanical properties are highly complex, and cells possess multiple mechanical behaviors in addition to elasticity. Therefore, the software presented here which can analyze various mechanical properties of cells will benefit the field of AFM-based mechanobiology. I have several comments for the paper in its current form:

1. The force curves in Fig. 3 and Fig. 4 are ideal force curves since the cantilever deflection keeps nearly zero before in contact with the cell, and it is easy to visually determine the contact point. In many cases, the obtained force curves during AFM mechanical experiments are not ideal, and baseline correction is often needed before the analysis of force curves. I think adding a baseline correction to the software will facilitate the analysis of experimental data.

2. Table 2 shows the most commonly used elastic models (Hertz-Sneddon) for extracting the Young's modulus of the specimen from force curves. Adding more theoretical models, such as JPK and DMT, which are also used in the field in case of the existence of adhesion between AFM tip and specimen, into the software will benefit comprehensively analyzing the force curves.

3. Details of AFM-based oscillatory microrheology experiments are lacking, such as instrumental requirement, data collection, and data analysis.

4. Besides Young's modulus, the stiffness (the unit is N/m) calculated from the force curves is also used to characterize the rigidity/softness of specimen, and thus adding a function of extracting specimen stiffness from the force curves will be meaningful.

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: atomic force microscopy, mechanobiology

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.