Computer Physics Communications 196 (2015) 429–438

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Flexible and modular virtual scanning probe microscope☆

John Tracey [a,*], Filippo Federici Canova [b], Olli Keisanen [a], David Z. Gao [c], Peter Spijker [a], Bernhard Reischl [d], Adam S. Foster [a]

[a] COMP, Department of Applied Physics, Aalto University, Otakaari 1, FI-00076 Aalto, Finland

[b] Aalto Science Institute, Aalto School of Science, PO Box 15500, FI-00076, Aalto, Finland

[c] Department of Physics and Astronomy, University College London, Gower Street, London WC1E 6BT, United Kingdom

[d] Nanochemistry Research Institute, Curtin Institute for Computation, Department of Chemistry, Curtin University, GPO Box U1987, WA 6845, Perth, Australia

ARTICLE INFO

Article history:
Received 24 December 2014
Received in revised form
2 April 2015
Accepted 13 May 2015
Available online 22 May 2015

Keywords:
NC-AFM
SPM
Simulation
Python

ABSTRACT

Non-contact Atomic Force Microscopy (NC-AFM) is an experimental technique capable of imaging almost any surface with atomic resolution, in a wide variety of environments. Linking measured images to real understanding of system properties is often difficult, and many studies combine experiments with detailed modelling, in particular using virtual simulators to directly mimic experimental operation. In this work we present the PyVAFM, a flexible and modular based virtual atomic force microscope capable of simulating any operational mode or set-up. Furthermore, the PyVAFM is fully expandable to allow novel and unique set-ups to be simulated, finally the PyVAFM ships with fully developed documentation and tutorial to increase usability.

Program summary

Program title: Python Virtual Atomic Force Microscope (PyVAFM)

Catalogue identifier: AEWX_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEWX_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 852449

No. of bytes in distributed program, including test data, etc.: 28531404

Distribution format:.ZIP

Programming language: Python input scripts and a C core.

Computer: Desktop.

Operating system: UNIX.

RAM: 500 Megabytes

Classification: 16.4.

External routines: GCC, Python 2.7, scipy and numpy

Nature of problem: Simulation of any atomic force microscope operational mode including experimental delays/artefacts.

Solution method: A modular simulation was developed where a user can connect several components together in order to simulate any operational mode. Each of these components is also developed to be

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: john.tracey@aalto.fi (J. Tracey).
URL: https://github.com/SINGROUP (J. Tracey).

http://dx.doi.org/10.1016/j.cpc.2015.05.013

mathematically similar to their real life counter parts hence incorporating any experimental delays or artefacts.

Restrictions: For tip-sample interactions beyond simple analytical forms, the interaction field should be provided by the user via separate simulations e.g first principles or classical calculations.

Unusual features: Modularity

Additional comments: The tutorials include several example tip-sample interaction approaches and fields, and authors can provide others upon request.

Running time: 2 h. The example given in the installation section of the user manual only takes about 30 s.

1. Introduction

Non-contact Atomic Force Microscopy (NC-AFM) [1,2] is an experimental technique capable of imaging surfaces with atomic resolution. The instrument is, in principle, able to probe any kind of material: it allows detailed analysis of previously unexplored insulating surfaces [3], and thereby adsorbed molecular layers [4,5]. NC-AFM has the ability to operate in atmospheric and liquid environments, allowing characterisation of a wide variety of interactions in very different mediums [6–8]. As a result of the technological success and wide applicability of NC-AFM, the instrument has reached a high level of complexity, and it is not always clear how measured signals relate to physical quantities. The basic principle of NC-AFM is to bring an atomically sharp tip attached to an oscillating cantilever within a few angstroms of a sample and detect changes in the cantilever's oscillation frequency ($\Delta\omega$) caused by the tip–sample interactions F_{TS}. In order to fully interpret the measurements, one needs to understand how the atomic/molecular species in the system determine F_{TS} and its relationship with the measured $\Delta\omega$. This is often not trivial.

Theoretical methods have often been used to reverse-engineer the experiment, in order to provide reliable interpretation of the measurements. Using atomistic models to describe the tip and the surface, the interactions resulting from the atomic configuration of the system can be calculated [9–11]. These are converted to $\Delta\omega$ and compared directly to experimental measurements using a model for the dynamics of the tip. A quite general but accurate description of the NC-AFM tip dynamics is given by:

$$\ddot{z} + \frac{\omega_0}{Q}\dot{z} + \omega_0^2(z - z_0) = R(t)\cos(\omega(t)t) + F_{\mathrm{TS}}(x, y, z) \qquad (1)$$

where x, y, z give the tip position, the cantilever resonant frequency and Q-factor are ω_0 and Q respectively, and z_0 represents the equilibrium position of the cantilever tip along z. The cantilever is driven into a constant-amplitude oscillation by feed-backs, generating an excitation signal of amplitude $R(t)$ and frequency $\omega(t)$ with a complex time-dependency. For this reason, Eq. (1) cannot be solved analytically. With further approximations, a variety of expressions to calculate F_{TS} from $\Delta\omega$ were formulated [12–14]. Unfortunately all these models do not account for the finite response time of the feed-back electronics in the instrument, making it impossible to estimate an artefact in the interactions extracted from $\Delta\omega$ or simulate bimodal cantilevers.

Previous attempts have been made to approach Eq. (1) numerically with *virtual machines*, that incorporated a description of the electronics found in the instrument [15–21]. The advantage to using these virtual machines is that they are designed to capture all aspects of a typical AFM experiment, including instrument induced artefacts. Without a complete simulation of the experimental setup, it is often impossible to directly compare theoretical predictions with experimental images in more complex techniques [22–25], particularly where analytical solutions are not available to describe the modes of operation. Generally, the available implementations only describe one particular experimental setup, usually operating in frequency modulation, and were developed specifically for an expert scientific user. As new imaging methods and operation modes are developed, a flexible virtual machine that can be easily modified and employed by a wide variety of users is increasingly necessary. In this work we present PyVAFM, a highly flexible and modular implementation of a virtual atomic force microscope with fully developed documentation and tutorials, and discuss its design principles, features and possible artefacts.

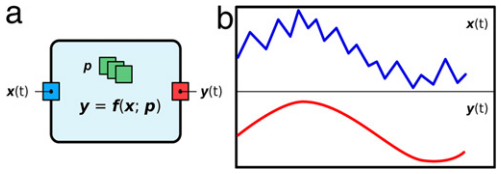

Fig. 1. (a) A typical circuit with input x, some process function f and output y. (b) An example input (x) and output (y) signal from a typical circuit.

2. Implementation

2.1. Design concepts

In PyVAFM, a complex differential equation such as Eq. (1) is represented by a network $\mathcal{M} = \{C_1, C_2, \ldots, C_n\}$ (or virtual machine) of computational units C_i called *circuits* (see Fig. 1). Each circuit C_i contains a set of input $\mathbf{x}$ channels, output $\mathbf{y}$ channels, and internal parameters $\mathbf{p}$. Circuits are characterised by their internal mechanisms, given by an operator $\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{p})$, which uses input channels and internal parameters to compute the output. Even though most of the circuits implement simple operators, such as arithmetic and logical operations, filters, and controllers, complex operators can be realised by connections, transferring the values of output channels to all connected inputs. PyVAFM solves the equation described by the network by integrating its components in discrete time-steps Δt. During each simulation step, all the circuits read their input channels $\mathbf{x}(t)$, integrate their operators and compute output values $\mathbf{y}(t + \Delta t)$ for the following time step.

For performance reasons, PyVAFM was implemented with both C and Python languages, exploiting their interoperability. The execution of the virtual machine involves iterations (for each time-step, loop over all C_i), which are notoriously inefficient in Python. Therefore, the numerical framework for the calculation is implemented in a C library as discussed in Section 2.2. The end-user only deals with an intuitive, object oriented Python interface (see Section 2.3), that makes handling the C library easier. Effort has been made to ensure that the code is well documented, and

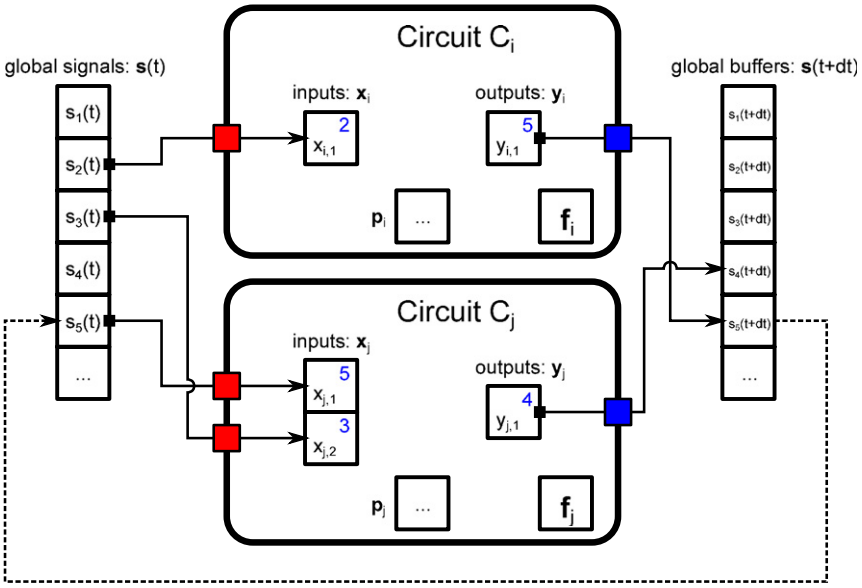

Fig. 2. Schematics of the circuit's memory representation.

this is achieved in the form of an extensive online user manual.[1] On this site one can find documentation on every circuit including a description of the process, what input and output channels they have and an example of how to use the circuit. Furthermore there are several tutorials that demonstrate typical AFM experiments and related input scripts.

2.2. C library

Circuits C_i are represented by data structures in the C core library, as illustrated in Fig. 2. Their internal parameters and I/O channels are stored in arrays, defined as follows:

$$\mathbf{p}_i = \left\{p_{i,1}, p_{i,2}, \ldots\right\}, \quad p_{i,j} \in \mathbb{R} \tag{2}$$

$$\mathbf{x}_i = \left\{x_{i,1}, x_{i,2}, \ldots\right\}, \quad x_{i,j} \in \mathbb{N} \tag{3}$$

$$\mathbf{y}_i = \left\{y_{i,1}, y_{i,2}, \ldots\right\}, \quad y_{i,j} \in \mathbb{N}. \tag{4}$$

In practice, $\mathbf{p}_i$ consists of different pointers for integer, floating point and arbitrary type of data required by the circuit, such as strings and file pointers for output. It should be noted that I/O channels $x_{i,1}$ and $y_{i,1}$ are integers, and do not correspond to the numerical values exchanged between connected circuits. The real *signals* are stored in a global array $\mathbf{s}(t) = \{s_1(t), s_2(t), \ldots, s_k(t)\}$ with $s_i(t) \in \mathbb{R}$: the value of the jth (input) channel of the ith circuit is $s_{x_{i,j}}$ (similar for outputs). Whenever a circuit is added to $\mathcal{M}$, each of its channels initialises one unique signal, extending the global array. Connections are performed simply by assigning the index of the source $y_{i,j}$ (output channel), to the destination $x_{k,l}$ (input channel) so that both channels will read the same signal. Upon disconnection, the destination $x_{k,l}$ will be reset to its initialisation value, thus pointing to its unique signal.

The C_i data structure holds a reference to its operator $\mathbf{f_i}$, called *update function*, as a pointer to functions defined for every circuit type. This way, the pointer $\mathbf{f_i}$ of every low-pass filter in $\mathcal{M}$ references the same function, declared only once. In order to maintain differences between circuits of the same type, update functions take as arguments the data structure of the particular C_i, use its data to locate channel signals and parameters, and update its mechanism by one time-step. This device has been chosen to emulate the object oriented behaviour lacking in C. The computed results can be stored internally in C_i as $\mathbf{p_i}$, or in a global *buffer* array $\mathbf{s}(t + \Delta t) = \{s_1(t + \Delta t), s_2(t + \Delta t), \ldots, s_k(t + \Delta t)\}$, through output channels. Therefore, the output computed by a particular circuit is not visible to the others until the successive time-step: this behaviour, and its possible issues will be discussed more in depth in Section 5. Once all C_i in $\mathcal{M}$ have been updated, the buffers $\mathbf{s}(t+\Delta t)$ are copied to the signals $\mathbf{s}(t)$ and the integration of a new time-step can proceed.

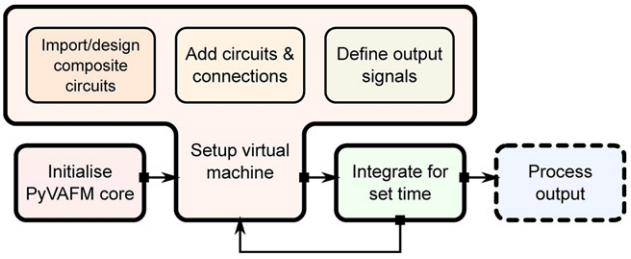

Fig. 3. Block diagram structure of the python input file.

It is possible for C_i to hold a list of references to a set of sub-circuits C^{sub}. In this case the circuit becomes a *container*, and its $\mathbf{f}_i$ loops over all C_j^{sub}, calling their respective update functions. This kind of circuit allows the end-user to design complex composite circuits and employ them in multiple virtual machines, with little to no changes in the input scripts. Containers have I/O meta-channels, allowing connections to be made to and from internal sub-circuits, as well external circuits and containers. The sub-circuits and their channels cannot be directly accessed from outside the container, which effectively acts as a black-box to other circuits. The virtual machine itself is the main container, and all other circuits are, directly or indirectly, contained.

2.3. Python interface and workflow

To the user, all the abstraction and complexity of the C library is hidden behind a Python interface, which is used to configure and run the virtual machine.

Fig. 3 illustrates the basic structure of an input file. First, the PyVAFM core is initialised, creating the main circuit container to be used as virtual machine. The user then proceeds to add basic

[1] http://singroup.github.io/PyVAFM/.

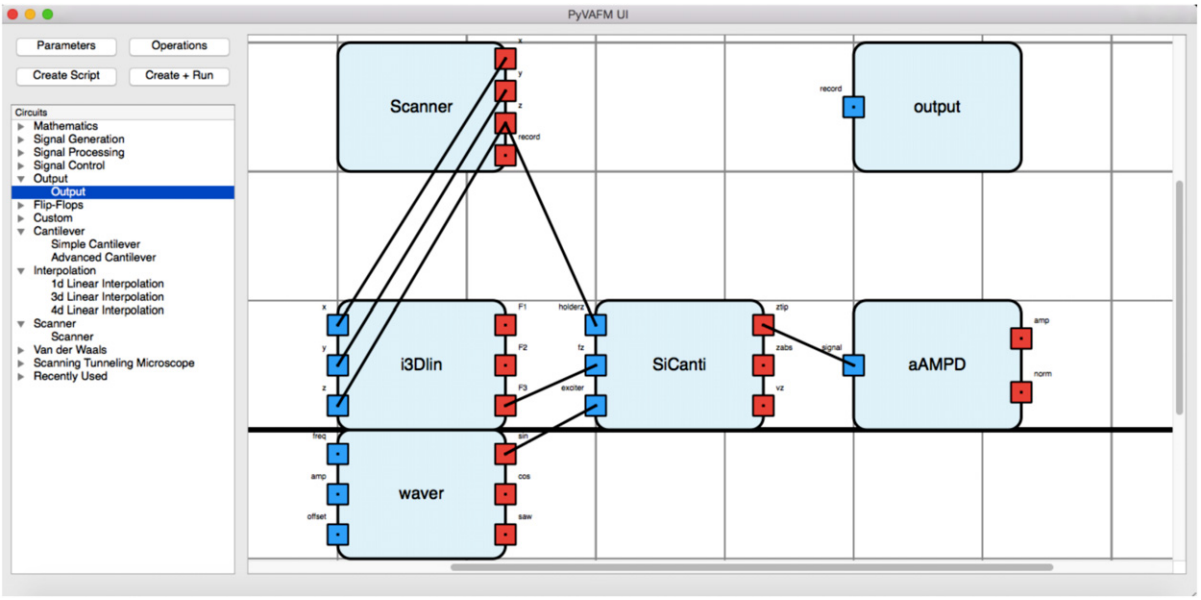

Fig. 4. Screenshot of the graphical user interface.

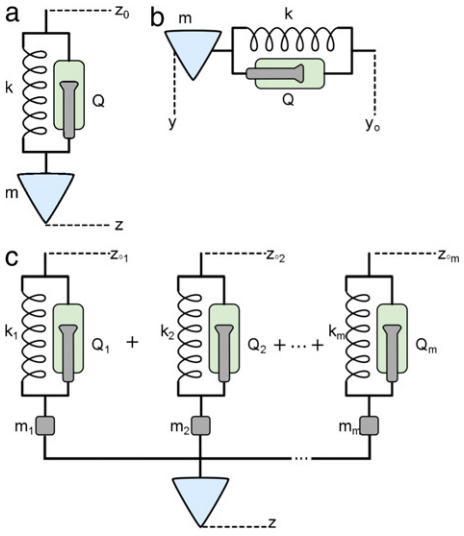

Fig. 5. Cantilever modes: (a) single flexural vertical oscillation mode, (b) single flexural lateral oscillation mode, (c) Multi-frequency oscillation mode.

circuits to the setup and connect their channels; composite circuits can also be imported from other python source files, or designed on-the-fly. With the output circuits, channel values are periodically printed to external files for data analysis. To further ease input file preparation, PyVAFM ships with a GUI capable of creating input scripts. The goal of the GUI (Graphical User Interface) is to allow users with little to no programming experience to use the PyVAFM. In the GUI it is possible to create input scripts by adding boxes (circuits) and connecting them by drawing lines between them. An example of this can be found in Fig. 4.

Once configured, the simulation can proceed by integrating the machine for a desired time. The scanner circuit offers further automation, advancing the simulation until an instruction, such as displacing the cantilever or 3D imaging, is completed. In addition to masking complexity, the Python interface shadows the structure of $\mathcal{M}$, its subcircuits and their channels, using classes, i.e. each C_i and $x_{i,j}$ has more complex object representation in the interface, referencing the memory allocation of the library. Python is used to check for inconsistencies in the user input, e.g. verifying that source or destination names of circuits and channels are valid and can be connected as instructed. Such string and hierarchy based operations are much easier to implement in Python. The interface also allows interactive execution of the virtual machine, similar to real experiments.

The PyVAFM circuit library includes all the basic components found in NC-AFM instruments. Nevertheless our modular implementation allows it to be expanded in two ways: by fully implementing new circuits, using both C and Python, or with PyCircuits. The latter only requires implementation of a Python class, following a few easy steps, but the performance is not as good in comparison to a full C implementation.

3. SPM models

3.1. Cantilever

In the simplest case the cantilever contains one flexural oscillation mode and is modelled as a damped harmonic oscillator that is subject to an external driving signal and F_{TS} (Fig. 5). The equation of motion for this system is the same as in Eq. (1). This model is implemented into the cantilever circuit and is solved using a modified version of the Velocity Verlet algorithm [26]. Velocity Verlet requires that the force input channel must be updated mid-time step. This poses an issue for the PyVAFM since it is only possible to update input and output channels at the end of a time step. This problem can be solved by reordering the Velocity Verlet algorithm so the forces are updated first, the amended algorithm is shown below:

1. Calculate the acceleration determined by the forces from the input channels (Driving Force + Tip-Sample Force + Cantilever Spring).
2. Calculate the half step velocity $v_{1/2}$, at $(t + \frac{1}{2}\Delta t)$.
3. Calculate position at $(t+\Delta t)z \leftarrow z+v_{1/2}\Delta t(1-\gamma\Delta t)+\frac{1}{2}a\Delta t^2$.
4. Calculate the velocity for the final half step using $v_{1/2}$ at $(t+\Delta t)$,

where γ is the dampening coefficient. By applying this algorithm to the discretised version of Eq. (1), it is possible to calculate the cantilevers position at every time step. This algorithm is well suited not only for simulating silicon cantilevers but also tuning fork cantilevers, due to the basic similarity of the dynamics. Due to the

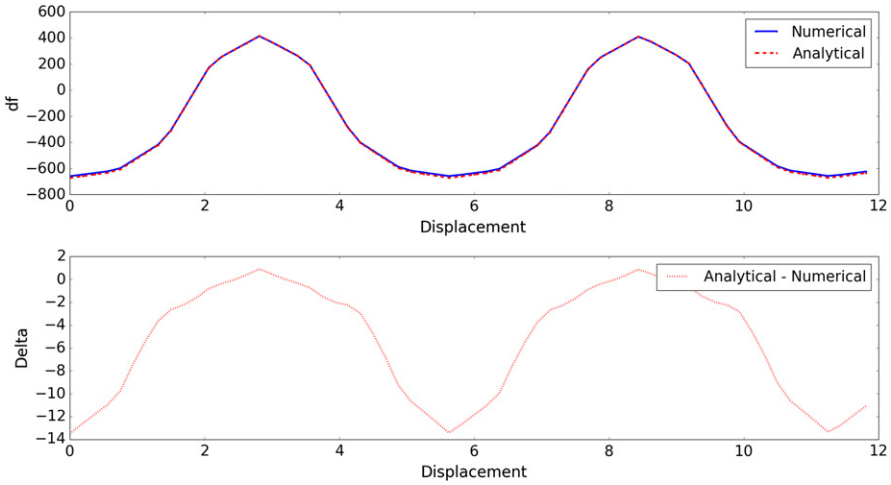

Fig. 6. Numerical and Analytical frequency shift comparison.

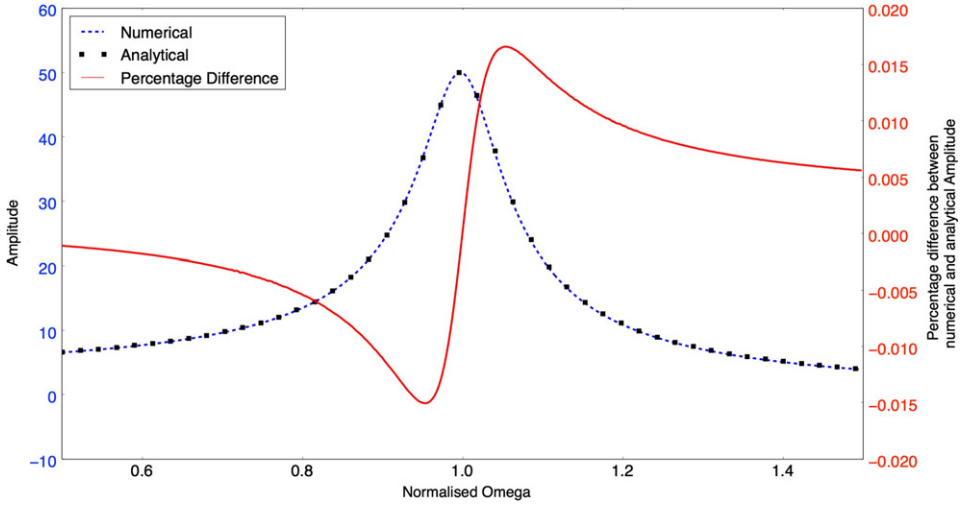

Fig. 7. Numerical and Analytical resonance response comparison.

modular nature of the PyVAFM it is possible to add further cantilever models to account for new and unique set-ups. For multi-frequency set-ups (Fig. 5(c)), Eq. (1) is extended as shown below:

$$\ddot{z_m} + \frac{\omega_m}{Q}\dot{z_m} + \omega_m^2(z_m - z_0) = R(t)\cos(\omega(t)t) + F_{\mathrm{TS}}(x, y, z_{\mathrm{tip}}) \quad (5)$$

where the notation remains the same except the subscript of m designates the individual eigenfrequency.

$$z_{\mathrm{tip}} = \sum_m z_m. \quad (6)$$

The resulting tip position is found by summing the solutions to Eq. (5) as can been seen in Eq. (6). Additionally it can be possible to simulate torsional modes by operating the cantilever in using the lateral oscillation modes as shown in Fig. 5(b).

In order to understand the influence of electronics in a simple measurement, the dynamics of the cantilever was compared to a well established analytical solution [27]. Fig. 6 shows the df curve for a single scan line above a NaCl surface. In both the analytical and numerical cases the AFM is operated in constant height mode at 4 Å above the surface, with a cantilever oscillating at an amplitude of 1 Å. The cantilevers Q factor is 10,000, spring constant is 2675.63 N/m and the resonance frequency is 150 kHz. The cantilever is then scanned over the surface at 1 Å/s with no background van der Waals interactions included. As one can see the maximum deviation from the analytical formulation is 13.50 Hz, equating to an accuracy of 97.75%. This difference is small compared to the range of Δf throughout the simulation. Regardless, this difference can be explained by the fact the electronics of the instrument are being taken into account in the numerical simulation. Fig. 7 demonstrates the resonance response of the cantilever. In this test, the amplitude of the driven cantilever at various driving frequencies was compared to the resonance response of an analytical harmonic oscillator. The numerical simulation agrees very well with the analytical solution resulting in 99.9834% accuracy. As an example to show the capabilities of the PyVAFM we have simulated constant height FM-AFM images for two distinct systems: NaCl in vacuum and calcite in water. In Fig. 8(a) we show the simulated image for NaCl which was constructed from a force field previously created by us [28] (for simulation parameters see the figure caption). When comparing this image with an experimental obtained image (Fig. 1d of [29]) good agreement in the contrast can be seen, with similar size and location of bright and dark spots. For calcite in water we used the same force field as before [30] and the simulated image, see Fig. 8b, has good resemblance to experimental images (Fig. 3a of [31]), including the observation of the protruding oxygen zig-zag pattern so common for calcite (indicated in Fig. 8(b)).

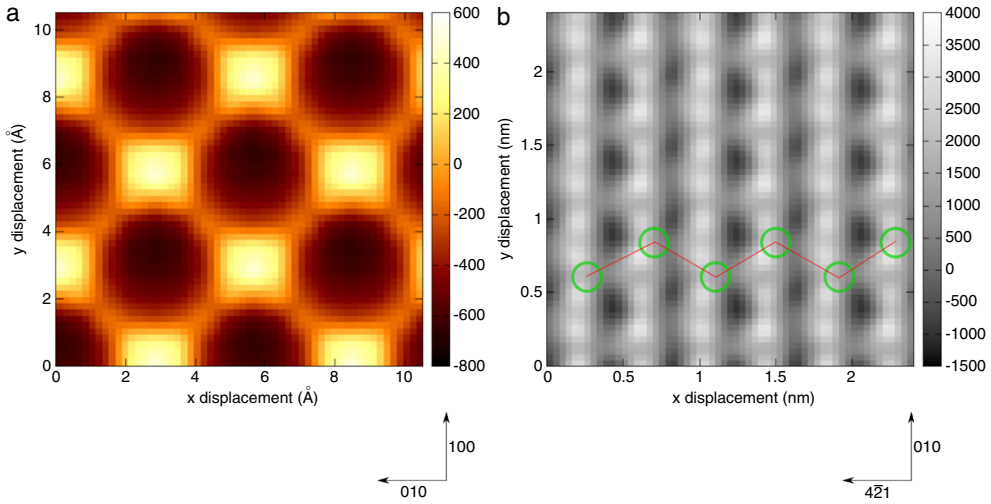

Fig. 8. Constant height simulated FM-AFM images for NaCl in vacuum (a) and calcite in water (b). In (b) the signature zig-zag pattern of the protruding calcite oxygen is indicated. For (a) the simulation parameters are $f0 = 150$ kHz, amplitude $= 0.1$ nm, $Q = 10{,}000$, height $= 0.4$ nm, and $k = 2675.63$ N/m; and for (b) $f0 = 350$ kHz, amplitude $= 0.12$ nm, $Q = 5.9$, height $= 0.48$ nm, and $k = 50.7$ N/m.

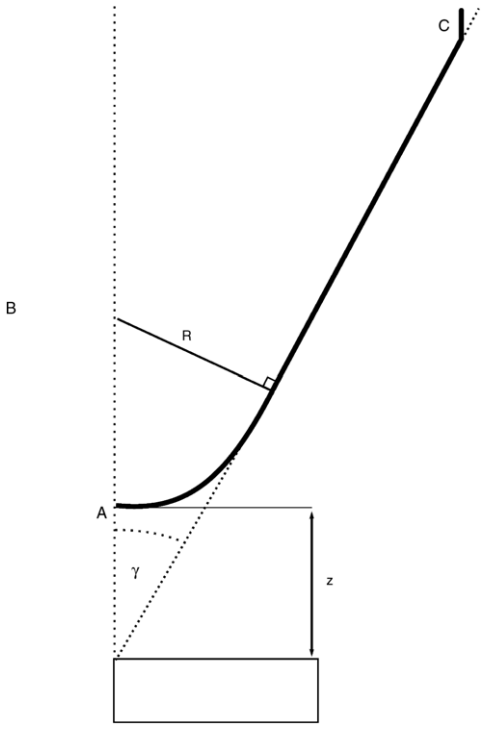

Fig. 9. van der Waals tip approximation, where γ is the angle of the cone, R is the tip radius, z is the tip–sample distance and A, B, C describe the spherical cap section of the cone.

3.2. Tip–sample interaction models

Tip–sample interactions, required in every NC-AFM simulation, are often divided into different contributions. The long-range van der Waals (vdW) interaction is not site-dependent and can be accurately approximated with classical models. PyVAFM includes a circuit modelling vdW as in Ref. [32]:

$$F_z(z) = \frac{AR^2(1-\sin\gamma)(R\sin\gamma - z\sin\gamma - R - z)}{6z^2(R+z-R\sin\gamma)^2} + \frac{-A\tan\gamma[z\sin\gamma + R\sin\gamma + R\cos(2\gamma)]}{6\cos\gamma(z+R-R\sin\gamma)^2}. \quad (7)$$

A schematic of these parameters can be found in Fig. 9. This force (and any analytical background interactions [10]) can be easily added directly within PyVAFM. For short-range, site-dependent forces, these are usually calculated with atomistic simulations (at the classical or quantum level) and tabulated for several positions of the tip above the surface, in a volumetric force field. Complex interactions such as absorbents on the surface or tip alteration by the medium can be included directly within this force field [30]. Hence, a further advantage of preparing the force field prior is the added flexibility in choosing operational mode and experimental parameters within the AFM simulation. A tri-linear interpolation circuit is used to obtain data in between these tabulations to allow the tip to feel a force at any point in the field. For a user with no experience or interest in MD/DFT simulations, they can use the simpler models below or take advantage of the many already simulated force fields, many of which are freely available from us. Several other interaction implementations are already present in PyAFM, providing models specific to certain operating modes or interactions:

- Images can be generated assuming a dipole tip model [33] using local electrostatic potential files as input.
- A simple mechanical model for imaging with a functionalised CO-tip can be used along with a simple classical model for tip–surface interactions [34,35].
- For simulations of Kelvin Probe Force Microscopy (KPFM) [11,36–38], bias dependent force fields are necessary and a four dimensional linear interpolation circuit is available.
- The PyVAFM is capable of simulating scanning tunnelling microscopy (STM) at constant height or current at the basic Tersoff–Hamann [39] level using partial electron density data as input.

New models developed in the future can be easily implemented as new circuits.

3.3. Filter theory

Electronic filters are one of the essential components in NC-AFM as well as generally in other instruments, yet only simple models have been used in previous virtual machines. Although PyVAFM includes a wide variety of ideal model filters, as an example, we will present here the implementation method of a second order low-pass filter.

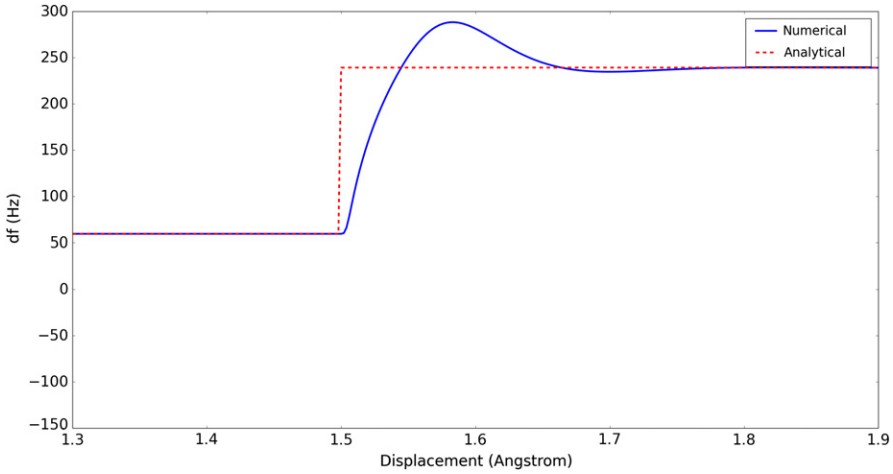

Fig. 10. Frequency shift over a step edge.

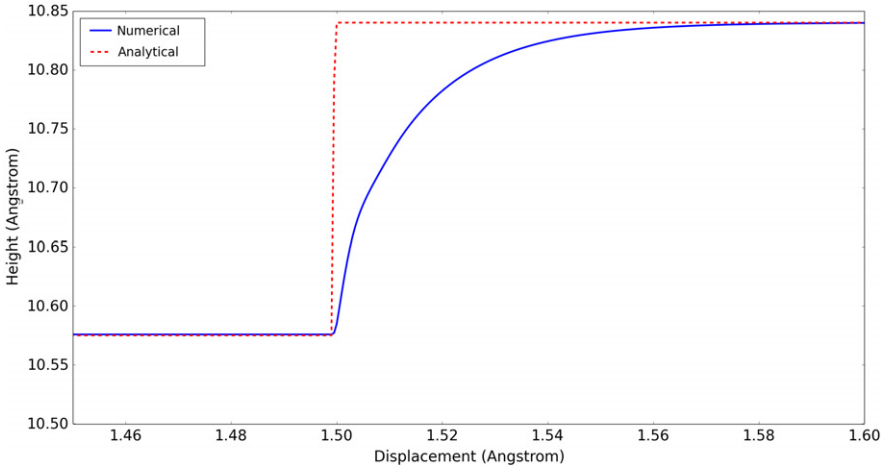

Fig. 11. Topography measurement over a step edge.

We start from the transfer function of the filter, which is often expressed in the frequency-domain:

$$y(s) = \frac{x(s)}{s^2 + \frac{\omega_c}{Q}s + \omega_c^2} \tag{8}$$

where $x(s)$ and $y(s)$ are the input and output signals of the filter at a given frequency s. The filter has a cutoff frequency ω_c, and a Q-factor Q. The frequency-domain representation, despite being easily found in electronics textbooks, is not particularly useful to our time-discretised virtual machine. Therefore, the transfer function is converted to time-domain using Laplace transform, replacing s with the operator d/dt:

$$\left[\frac{d^2}{dt^2} + \frac{\omega_c}{Q}\frac{d}{dt} + \omega_c^2\right] y(t) = x(t). \tag{9}$$

By replacing derivatives with finite differences the following expression can be found:

$$x(dt) = \frac{y(dt) - \frac{y(t-2dt)-2y(t-dt)}{dt^2} + \frac{\omega_c}{2Qdt}y(t-2dt)}{\frac{1}{dt^2} + \frac{\omega_c}{2Qdt} + \omega_c^2}, \tag{10}$$

where dt is the length of a time step. All filters in PyVAFM have been implemented following this procedure.

4. Results

As mentioned before, the electronics of an AFM experiment have an impact on the resulting image. In this section these artefacts are demonstrated using three example cases. A force field was created that resembles a step edge with an abrupt change in force (and force gradient) and used in each example. The experimental parameters and analytical model chosen are identical to those found in Section 3.1.

4.1. Constant height

In this example, the step edge force field is investigated using the analytical model and numerical VAFM model. In both cases the cantilever is operated in constant height mode and moved over the step edge and the resulting Δf is compared.

As one can see from Fig. 10 in both analytical and numerical cases a new frequency shift value is obtained after the step edge is encountered at 1.5 Å. The resulting delays from the PLL (Phase Locked loop), AGC (Automatic Gain Control), etc., have caused the frequency shift of the numerical signal to require additional time before locking onto the new frequency, which is absent in the analytical model. This delay can result in signal artefacts that can be reproduced by the PyVAFM.

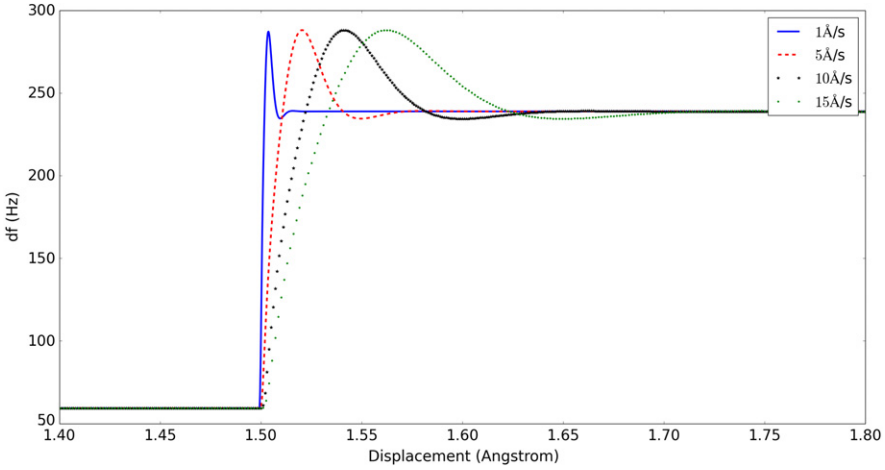

Fig. 12. Various scan speeds over a step edge.

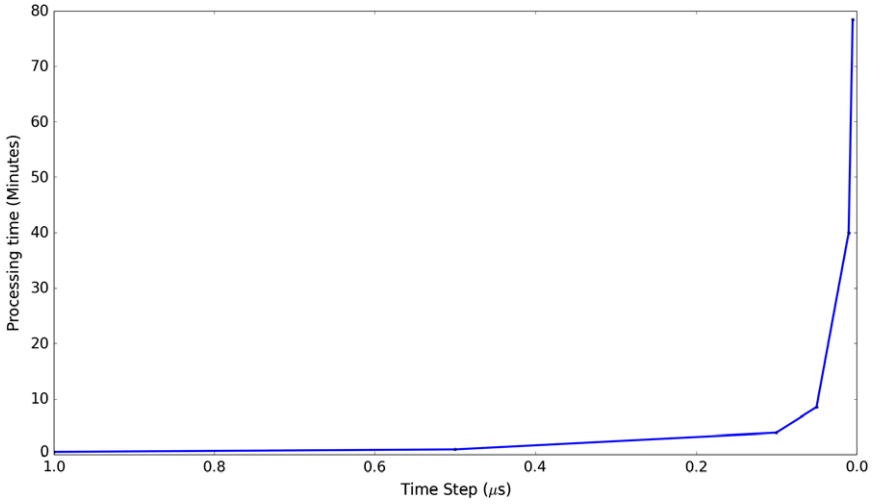

Fig. 13. Time required to execute one scan line at various time steps.

4.2. Topography

Constant frequency measurements (topography) are common place in experimental setups and the PyVAFM is capable of reproducing these experiments. A slightly different approach was taken when calculating the analytical result. For a given point the frequency is checked at various different heights and compared to our setpoint in order to simulate a constant frequency experiment.

As with the previous test, the step edge is again found at 1.5 Å. As seen in Fig. 11, the constant height measurement using the analytical model yields an instantaneous response. Although the PyVAFM model produces a very finite response that is exaggerated over the constant height case due to the use of the proportional integral circuit. This particular circuit is responsible for producing the signal that moves the tip up or down in order to maintain constant frequency. As with its real life counterpart this has some finite response time which causes a non-instantaneous response as demonstrated in Fig. 11.

4.3. Scan speed

In this section the effect of varying scan speeds (1 Å/s, 5 Å/s, 10 Å/s and 15 Å/s) is demonstrated. This test was performed using constant height mode over the same step edge force field.

It is clear that as the scan speed increases, the inaccuracy, with respect to the true *shape* of the step-edge, of the experiment also increases as shown in Fig. 12. This is due to the finite lockon time of the PLL, which, as the scan speed increases over the step edge, will yield an ever increasing delay. Since the analytical model does not take into account scan speed or the electronics of the system these delays would not be reproduced without the use of the PyVAFM.

5. Known issues

5.1. Energy dissipation

The PyVAFM does not contain any energy dissipation models as standard [21,40], so it is not possible to separate energy dissipation and normal surface interaction in a typical experiment. However it is entirely possible to add custom circuits that contain surface dissipation models in order to simulate this.

5.2. Circuit performance

In order to analyse the performance of the PyVAFM, the total run time for a scan line in a typical AFM setup was measured. The setup is identical to the one used in Section 3.1. The expected result can be seen in Fig. 13, where the time per scan line increases as the time step is decreased. In order to obtain a Verlet solution to an appropriate accuracy the time step must be altered depending on the frequency of the cantilever. For most frequencies within the range of AFM experiments the time step of 0.05 to 0.01 μs

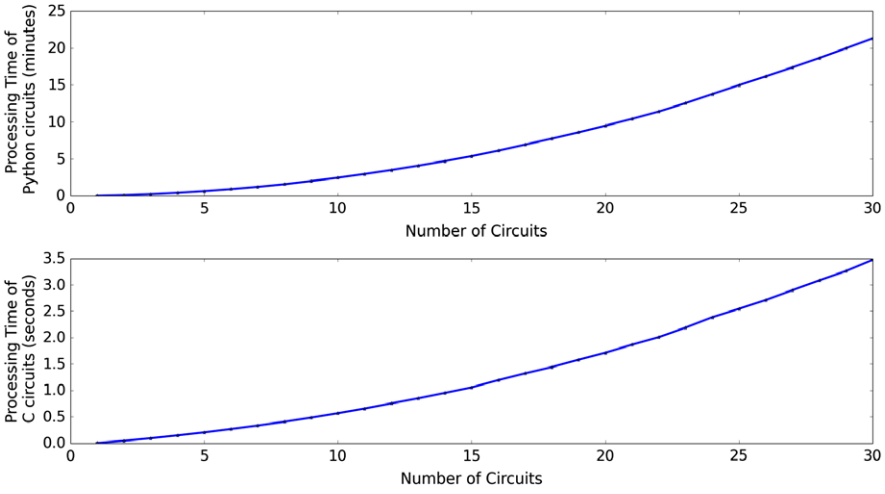

Fig. 14. Processing time for Python and C circuits.

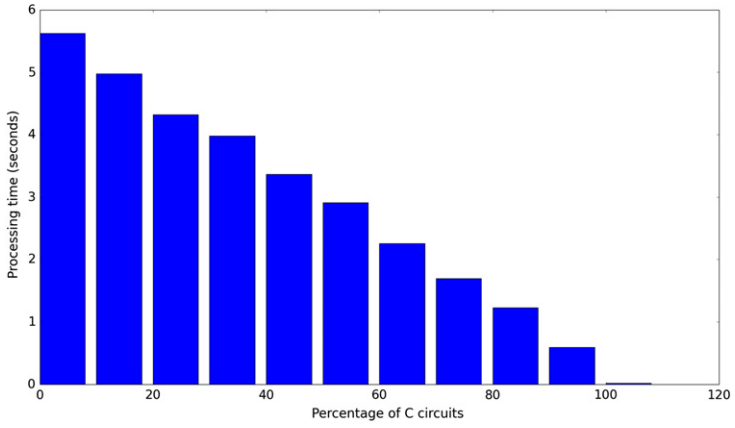

Fig. 15. Performance analysis of python and C circuit mixtures.

is appropriate. This yields a processing time of between 8.53 and 39.95 min respectably per scan line. The trend may seem surprising at first, but this behaviour is expected.

$$T = cN = c\frac{T_s}{\Delta T}. \tag{11}$$

By referring to Eq. (11) where T is total processing time, c is a constant, N is the number of time steps, T_s is the simulation time and ΔT is the time step, this behaviour can be explained. The processing time is proportional to N and N is equal to $T_s/\Delta T$. It can be seen that when T_s is held constant the processing time is proportional to $1/\Delta T$, as shown in Fig. 13.

As mentioned in Section 2.1, it is possible to implement new circuits in Python. A comparison of the performance of Python and C circuits can be found in Fig. 14. During this test, increasing numbers of addition circuits are added to the simulation. Addition circuits are designed in both Python and C. The processing time of the Python and C circuits is then compared by analysing the time required to process 5000 s of simulation time. As can be seen Python is clearly slower than C. It is also worth noting that the relative reduction in speed is far more notable in the Python circuits as more circuits are added, which ranges from 2.87 ms to 21 min compared to the C version that ranges from 3.986 ms to 3.47 s. In Fig. 15 an analysis regarding the performance change when inducing a mixture of Python and C circuits is performed. This analysis is performed over various percentages of C circuits and compared. As expected the speed increases as more C circuits are added. One interesting property is the fact that the speed increases linearly with the percentage of C circuits.

$$N_pT_p + N_cT_c = T. \tag{12}$$

Eq. (12) demonstrates this linearity where T is total processing time, N_p is the number of Python circuits, N_c is the number of C circuits, T_p and T_c is the time to process a single Python or C circuit respectably. T_p and T_c are essentially constant within our simulation so by taking this into consideration with the fact that we are increasing N_c and decreasing N_p in equal amounts a linear result would be expected.

6. Conclusion

In this work we have developed a fully modular atomic force microscope simulator, providing a flexible platform for simulating any existing or future SPM modes. At its most basic level, PyVAFM allows theoretical researchers to directly compare results produced from simulations to experimental measurements and immediately get feedback on the quality of the comparison and the accuracy of the simulated setup. Beyond this is an important training aspect, accessed by the efforts made to make PyVAFM both modification- and user-friendly, with detailed documentation. Specifically, experimental researchers can use PyVAFM themselves as a testing tool for high resolution imaging by preparing an appropriate force field, PyVAFM can re-create many of the conditions of a real experiment (including experimental artefacts) with-

out the consequences of experimental failure. This can be used for training in conventional operation and calibration, and also for testing more exotic setups as a pre-requisite to real operation.

Future development of PyVAFM will be continuous in parallel with the updates in SPM experimental techniques and according to the demands of research problems faced. The open source nature of the code means that it will be community driven, and several upgrades have been catalysed (and even implemented) from beyond the original code authors. We hope this will continue and PyVAFM will be prove to be a useful tool for the wide SPM community. In this regard, we soon hope to interface PyVAFM with the common experimental image analysis tool Gwyddion, so experimental and simulated images can be compared with the same tools.[2]

Acknowledgements

This research has been supported by the Academy of Finland through its Centres of Excellence Program project no. 915804 and EU project PAMS (contract no. 610446). Thanks to Olli Keisanen for developing the GUI interface for the PyVAFM. FFC thanks the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) for financial support through the WPI programme. D.Z.G. acknowledges the use of the HECToR and Archer High Performance Computing Facilities via our membership to the UK's HPC Materials Chemistry Consortium, which is funded by EPSRC (EP/F067496).

References

[1] S. Morita, R. Wiesendanger, E. Meyer (Eds.), Noncontact Atomic Force Microscopy, Springer, Berlin, 2002.
[2] F.J. Giessibl, Advances in atomic force microscopy, Rev. Modern Phys. 75 (2003) 949.
[3] Gerhard Meyer, Nabil M. Amer, Optical-beam-deflection atomic force microscopy: The NaCl(001) surface, Appl. Phys. Lett. 56 (21) (1990) 2100.
[4] Knud Lämmle, Thomas Trevethan, Alexander Schwarz, Matthew Watkins, Alexander Shluger, Roland Wiesendanger, Unambiguous determination of the adsorption geometry of a metal–organic complex on a bulk insulator, Nano Lett. 10 (2010) 2965.
[5] Alexander Schwarz, David Z. Gao, Knud Lämmle, Josef Grenz, Matthew B. Watkins, Alexander L. Shluger, Roland Wiesendanger, Determining adsorption geometry, bonding, and translational pathways of a metal organic complex on an oxide surface: Co-salen on NiO(001), J. Phys. Chem. C 117 (2) (2013) 1105–1112.
[6] Atsushi Ikai, STM and AFM of bio/organic molecules and structures, Surf. Sci. Rep. 26 (8) (1996) 261–332.
[7] Dimitrios Fotiadis, Simon Scheuring, Shirley A. Müller, Andreas Engel, Daniel J. Müller, Imaging and manipulation of biological structures with the afm, Micron 33 (4) (2002) 385–397.
[8] T. Fukuma, K. Onishi, N. Kobayashi, A. Matsuki, H. Asakawa, Atomic-resolution imaging in liquid by frequency modulation atomic force microscopy using small cantilevers with megahertz-order resonance frequencies, Nanotechnology 23 (2012) 135706.
[9] R. García, R. Pérez, Dynamic atomic force microscopy methods, Surf. Sci. Rep. 47 (2002) 197.
[10] W.A. Hofer, A.S. Foster, A.L. Shluger, Theories of scanning probe microscopes at the atomic scale, Rev. Modern Phys. 75 (4) (2003) 1287–1331.
[11] Clemens Barth, Adam S. Foster, Claude R. Henry, Alexander L. Shluger, Recent trends in surface characterization and chemistry with high-resolution scanning force methods, Adv. Mater. 23 (4) (2011) 477–501.
[12] F.J. Giessibl, Forces and frequency shifts in atomic-resolution dynamic-force microscopy, Phys. Rev. B 56 (1997) 16010.
[13] A.I. Livshits, A.L. Shluger, A.L. Rohl, A.S. Foster, Model of noncontact scanning force microscopy on ionic surfaces, Phys. Rev. B 59 (3) (1999) 2436–2448.
[14] J.E. Sader, S.P. Jarvis, Accurate formulas for interaction force and energy in frequency modulation force spectroscopy, Appl. Phys. Lett. 84 (2004) 1801.
[15] G. Couturier, J.P. Aime, J. Salardenne, R. Boisgard, A. Gourdon, S. Gauthier, A mechanical approach to the dissipation process in NC-AFM: experiments, model and simulation, Appl. Phys. Mater. Sci. Process. 72 (2001) S47–S50.
[16] G. Couturier, J.P. Aim 'e, J. Salardenne, R. Boisgard, A virtual non contact-atomic force microscope (NC-AFM): Simulation and comparison with analytical models, Eur. Phys. J. Appl. Phys. 15 (2001) 141–147.
[17] J. Polesel-Maris, S. Gauthier, A virtual dynamic atomic force microscope for image calculations, J. Appl. Phys. 97 (2005) 044902.
[18] L. Nony, A. Baratoff, D. Schär, O. Pfeiffer, A. Wetzel, E. Meyer, Noncontact atomic force microscopy simulator with phase-locked-loop controlled frequency detection and excitation, Phys. Rev. B 74 (2006) 235439.
[19] M. Watkins, T. Trevethan, A. Shluger, L. Kantorovich, Dynamical processes at oxide surfaces studied with the virtual atomic force microscope, Phys. Rev. B 76 (24) (2007) 245421.
[20] Thomas Trevethan, Lev Kantorovich, Jérôme Polesel-Maris, Sébastien Gauthier, Alexander Shluger, Multiscale model of the manipulation of single atoms on insulating surfaces using an atomic force microscope tip, Phys. Rev. B 76 (2007) 085414.
[21] T. Trevethan, L. Kantorovich, J. Polesel-Maris, S. Gauthier, Is atomic-scale dissipation in NC-AFM real? Investigation using virtual atomic force microscopy, Nanotechnology 18 (8) (2007) 084017.
[22] F. Federici Canova, A.S. Foster, M.K. Rasmussen, K. Meinander, F. Besenbacher, J.V. Lauritsen, Non-contact atomic force microscopy study of hydroxyl groups on the spinel $MgAl_2O_4$(100) surface, Nanotechnology 23 (32) (2012) 325703.
[23] Shigeki Kawai, Filippo Federici Canova, Thilo Glatzel, Teemu Hynninen, Ernst Meyer, Adam S. Foster, Measuring electric field induced subpicometer displacement of step edge ions, Phys. Rev. Lett. 109 (2012) 146101.
[24] Morten K. Rasmussen, Adam S. Foster, Berit Hinnemann, Filippo F. Canova, Stig Helveg, Kristoffer Meinander, Natalia M. Martin, Jan Knudsen, Alina Vlad, Edvin Lundgren, Andreas Stierle, Flemming Besenbacher, Jeppe V. Lauritsen, Stable cation inversion at the $MgAl_2O_4$(100) surface, Phys. Rev. Lett. 107 (2011) 036102.
[25] Ania Amrous, Franck Bocquet, Laurent Nony, Franck Para, Christian Loppacher, Simon Lamare, Frank Palmino, Fréderic Cherioux, David Z. Gao, Filippo Federici Canova, Matthew B. Watkins, Alexander L. Shluger, Molecular design and control over the morphology of self-assembled films on ionic substrates, Adv. Mater. Interfaces 1 (9) (2014) 2196–7350.
[26] William C. Swope, Hans C. Andersen, Peter H. Berens, Kent R. Wilson, A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters, J. Chem. Phys. 76 (1) (1982) 637–649.
[27] S. Akamine, R.C. Barrett, C.F. Quate, Improved atomic force microscope images using microcantilevers with sharp tips, Appl. Phys. Lett. 57 (3) (1990) 316–318.
[28] Kendal W. Clark, Shengyong Qin, X.-G. Zhang, An-Ping Li, Nanoscale periodic modulations on sodium chloride surface revealed by tuning fork atomic force microscopy, Nanotechnology 23 (2012) 185306.
[29] Shigeki Kawai, Filippo Federici Canova, Thilo Glatzel, Adam S. Foster, Ernst Meyer, Atomic-scale dissipation processes in dynamic force spectroscopy, Phys. Rev. B 84 (2011) 115415.
[30] Bernhard Reischl, Matthew Watkins, Adam S. Foster, Free energy approaches for modeling atomic force microscopy in liquids, J. Chem. Theory Comput. 9 (2013) 600.
[31] Sebastian Rode, Noriaki Oyabu, Kei Kobayashi, Hirofumi Yamada, Angelika Kühnle, True atomic-resolution imaging of (1014) Calcite in Aqueous solution by frequency modulation atomic force microscopy, Langmuir 25 (5) (2009) 2850–2853.
[32] C. Argento, R.H. French, Parametric tip model and force distance relation for hamaker constant determination from atomic force microscopy, J. Appl. Phys. 80 (6081) (1996).
[33] David Zhe Gao, Josef Grenz, Matthew Benjamin Watkins, Filippo Federici Canova, Alexander Schwarz, Roland Wiesendanger, Alexander L. Shluger, Using metallic noncontact atomic force microscope tips for imaging insulators and polar molecules: tip characterization and imaging mechanisms, ACS nano 8 (5) (2014) 5339–5351.
[34] L. Gross, F. Mohn, N. Moll, P. Liljeroth, G. Meyer, The chemical structure of a molecule resolved by atomic force microscopy, Science 325 (5944) (2009) 1110–1114.
[35] Prokop Hapala, Georgy Kichin, Christian Wagner, F. Stefan Tautz, Ruslan Temirov, Pavel Jelinek, Mechanism of high-resolution STM/AFM imaging with functionalized tips, Phys. Rev. B 90 (8) (2014) 085421.
[36] H.O. Jacobs, P. Leuchtmann, O.J. Homan, A. Stemmer, Resolution and contrast in Kelvin probe force microscopy, J. Appl. Phys. 84 (3) (1998) 1168–1173.
[37] Ulrich Zerweck, Christian Loppacher, Tobias Otto, Stefan Grafström, Lukas M. Eng, Accuracy and resolution limits of kelvin probe force microscopy, Phys. Rev. B 71 (2005) 125424.
[38] Wilhelm Melitz, Jian Shen, Andrew C. Kummel, Sangyeob Lee, Kelvin probe force microscopy and its application, Surf. Sci. Rep. 66 (1) (2010) 1–27.
[39] J. Tersoff, D.R. Hamann, Theory of the scanning tunneling microscope, Phys. Rev. B 31 (1985) 805–813.
[40] F. Federici Canova, A.S. Foster, The role of the tip in non-contact atomic force microscopy dissipation images of ionic surfaces, Nanotechnology 22 (2011) 045702.

[2] http://gwyddion.net.