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 xxxx

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
0010-4655/© 2015 Elsevier B.V. All rights reserved.

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:

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

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.

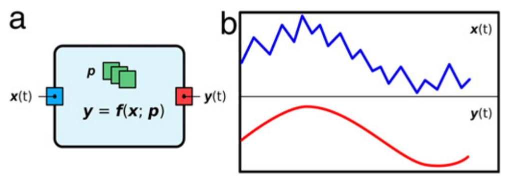

Fig. 1. (a) A typical circuit with input x, some process function f and output y. (b) An example input (x) and output (y) signal from a typical circuit.

2. Implementation

2.1. Design concepts

In PyVAFM, a complex differential equation such as Eq. (1) is represented by a network $\mathcal{M} = \{C_1, C_2, \dots, C_n\}$ (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