MEMBRANE ELASTIC HETEROGENEITY STUDIED AT NANOMETRICAL SCALE ON LIVING CELLS

THÈSE N^{O} 3985 (2007)

PRÉSENTÉE LE 20 DÉCEMBRE 2007
À LA FACULTÉ DES SCIENCES DE LA VIE
LABORATOIRE DE NEUROBIOLOGIE CELLULAIRE
PROGRAMME DOCTORAL EN NEUROSCIENCES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Charles RODUIT

biologiste diplômé de l'Université de Lausanne
de nationalité suisse et originaire de Leytron (VS)

acceptée sur proposition du jury:

Prof. R. Schneggenburger, président du jury
Prof. S. Catsicas, Dr S. Kasas, directeurs de thèse
Prof. G. Dietler, rapporteur
Prof. Y. Dufrêne, rapporteur
Prof. B. Samori, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2008

Werner Heisenberg [1958]:

> We have to remember that what we observe is not nature in itself but nature exposed to our method of questioning.

Contents

Summary ix

Résumé xi

Remerciements xiii

Acronyms xv

1 Introduction 1
1.1 Atomic force microscopy . . . 1
1.1.1 History . . . 1
1.1.2 Principle . . . 1
1.1.3 The AFM operating modes . . . 2
Contact mode . . . 2
Tapping mode . . . 2
Force Spectroscopy . . . 4
1.2 Cell mechanical properties . . . 8
1.3 Cell Membrane . . . 11
1.3.1 The membrane organization . . . 11
1.3.2 The lipid raft model . . . 12
1.3.3 Role of the rafts . . . 12
Signal transduction . . . 12
Secretory pathway . . . 13
Endocytosis . . . 13
Cell adhesion and migration . . . 13
Disorders . . . 14
1.4 Aim of the project . . . 15

2 Materials and methods 17
2.1 Biochemistry . . . 17
2.1.1 Proteins . . . 17
2.1.2 Antibodies . . . 17
2.2 Cell cultures . . . 17
2.2.1 293T Cells . . . 17
2.2.2 HeLa Cells . . . 18
2.2.3 Hypocampal neurons . . . 18
2.3 Immunochemistry . . . 18
2.4 AFM . . . 19

2.4.1 AFM Tip 19
2.4.2 In vivio measurements 19
2.4.3 Force Volume measurements 20
2.4.4 Force-Volume analysis software 20
2.4.5 Data representations 21

3 Post Processing software 23
3.1 Introduction 23
3.2 Results 23
3.2.1 Zero Force Image 23
3.2.2 Stiffness Computation 24
3.2.3 Event detection 26
3.2.4 Relative Stiffness 26
3.2.5 Merging the results 28
3.3 Conclusion 28

4 Mechanical properties of GPI domains 31
4.1 Introduction 31
4.2 Results 33
4.2.1 Binding specificity of aerolysin 33
4.2.2 Distribution of GPI-anchored proteins 34
4.2.3 Tip contamination 35
4.2.4 Determining the local stiffness inhomogeneity 35
4.2.5 Relative stiffness of GPI domains 39
4.2.6 Cholesterol extraction effect on GPI domains relative stiffness 39
4.2.7 Cholesterol extraction effect on the number of detected GPI domains 40
4.2.8 Cytoskeleton digestion effect on the relative stiffness of GPI domains 40
4.2.9 Binding protein events influence on the relative stiffness . 43
4.2.10 Cell type influence on GPI domains relative stiffness . . . 43
4.2.11 Protein diffusion influence on GPI-domains relative stiffness 47
4.2.12 Cholesterol extraction effect on membrane global stiffness heterogeneity 48
4.3 Discussion 49
4.3.1 GPI-anchored proteins are specifically detected by the aerolysin coated tips 49
4.3.2 GPI domains are local stiffer zones 49
4.3.3 GPI domains relative stiffness is not dependent on protein-protein interactions 50
4.3.4 GPI domains relative stiffness does not depend on actin cytoskeleton 51
4.3.5 GPI domains relative stiffness has similar properties in several cell lines 51
4.3.6 Protein diffusion does not alter the GPI domains relative stiffness measurement 51
4.3.7 Cell stiffness global heterogeneity is altered by cholesterol extraction 52
4.4 Conclusion 53

5 Stiffness Tomography **55**
5.1 Introduction 55
5.1.1 The theory 55
5.2 Results 58
5.2.1 Finite elements simulations 58
5.2.2 Application of the Stiffness Tomography (ST) to living systems 63
Effect of cytoskeleton digestion 63
Tomographic view of neurites 63
5.3 Conclusion 64

A Blender script **65**
A.1 createTopoYoungEvent.py 65
A.2 TomographyLoader.py 71

B Succellus User Guide **75**

C Succellus Developer's Guide **91**

Bibliography **208**

Curriculum Vitae **218**

List of Figures

1.1 AFM Principle 3
1.2 Contact Mode 3
1.3 Tapping Mode 4
1.4 Force-distance curve 4
1.5 Protein-protein interaction 6
1.6 Force curve on soft sample 7
1.7 Biomechanical assay 10

2.1 Cantilever 19
2.2 Injection setup 20

3.1 Point of contact detection 24
3.2 Indentation curve 25
3.3 Curve segmentation 25
3.4 Event shape 27
3.5 Time spent during relative computation 27
3.6 3D scan result 28

4.1 Binding specificity of Aerolysin 33
4.2 GPI fluorescence distribution 34
4.3 GPI interaction distribution 35
4.4 Tip contamination 36
4.5 Typical neurite scan results 36
4.6 Determining stiffness heterogeneity 37
4.7 GPI domains are stiffer 39
4.8 MeCD effect on the GPI domain relative stiffness. 41
4.9 Change in binding-unbinding population event 42
4.10 Actin depolymerization effects on the GPI domain relative stiffness and absolute membrane stiffness 44
4.11 Actin depolymerization effect on cell stiffness 45
4.12 Cell stiffness 45
4.13 Relative stiffness of non GPI domain 45
4.14 GPI domain on other cell type 46
4.15 GPI domain relative stiffness on fixed cells 47
4.16 Stiffness heterogeneity 48

5.1 AFM Tomography Principle : Soft inclusion 56
5.2 AFM Tomography Principle : Hard inclusion 57

5.3 AFM Tomography Principle : Curve segmentation 57
5.4 Reference 58
5.5 Stiffer columns 59
5.6 Stiffer platforms 60
5.7 Stiffer big platforms 60
5.8 More stiffer platforms 61
5.9 Softer platforms 62
5.10 Cytoskeleton digetion effect on stiffness tomography 63
5.11 Neurites Stiffness Tomography 64

Summary

The aim of this thesis was to explore the cell mechanical properties using the Atomic Force Microscope (AFM).

The cell membrane contains lipids microdomains, called rafts, enriched in cholesterol and sphingolipids. The rafts are believed to play an important role in signal processing by acting as a "signaling platform". Indeed, membrane proteins involved in signal transduction concentrates into these rafts and are coupled with signaling pathways inside the cell. The mechanical properties of these rafts were characterized by targeting one of its component, the glycosylphosphatidylinositol-anchored protein (GPI-AP).

During this work, we found these domains to be stiffer than the surrounding membrane. Several control experiments were performed to consolidate this finding. The extraction of cholesterol, one of the major component of raft, was shown to dramatically reduce the stiffness of the raft to reach the surrounding membrane value. The stiffness specificity of rafts may be related to the lower diffusion rate of proteins and can be, therefore, an important property for its role as a signaling platform.

During this thesis, we also introduced a new AFM imaging mode, which we called "stiffness tomography". With this imaging mode, we were able to distinguish stiff materials inclusion located into the sample. Different control experiments were done to validate this imaging mode. A virtual experiment was performed with the help of the finite element modeling. This permitted us to validate our methodology, but also pointed us its limitations. The stiffness tomography was also used on living cells and showed significant differences between native and cytoskeleton depolimerized cells.

Since no postprocessing tools was available at the beginning of this work, the software development was a very significant part of the project. Its development resulted in a toolbox (a collection of function), that is available for future software development. A non negligible part of the development consisted in the toolbox documentation that is reported in the appendix C. This software permitted to process force volume AFM files and to characterize the elastic properties of the cell membrane with a high precision and reliability.

Keywords : hippocampal neurons, rafts, cholesterol, atomic force microscopy, GPI-anchored proteins, actin, stiffness tomography

Résumé

Cette thèse consistait à explorer les propriétés mécaniques des cellules avec l'aide d'un microscope à force atomique (AFM).

La membrane cellulaire contient des microdomaines lipidiques, appelés rafts, enrichis en cholestérol et sphingolipides. Les rafts sont supposés agir comme plateforme de signalisation en concentrant les protéines impliquées dans la transmission du signal de l'extérieur vers l'intérieur de la cellule et en les couplant avec la cascade de signalisation à l'intérieur de la cellule. Les propriétés mécaniques de ces raft ont été caractérisées en ciblant l'un de ses constituant, les protéines ancrées dans la membrane par glycosylphosphatidylinositol (GPI-AP).

Grace à ce travail, nous avons montré que ces domaines sont plus durs que la membrane adjacente, ce qui a été confirmé par plusieurs expériences contrôle. L'extraction de cholestérol, un des composants majeur des rafts, a montré une nette diminution de la dureté des rafts, allant jusqu'à celle de la membrane adjacente. La dureté spécifique des rafts peut être mise en relation avec la faible diffusion des protéines observée dans ces domaines et peuvent, par conséquent, être une propriété importante dans son rôle de platforme de signalisation.

Cette thèse nous a aussi permi de développer un nouveau mode d'imagerie, que nous avons baptisé "tomographie de dureté". Celle-ci permet la détection de matériaux à l'intérieur d'un échantillon, en se basant sur leurs différence de dureté. Plusieurs expériences contrôles ont été réalisées pour valider ce mode d'imagerie. Une expérience virtuelle, mettant en œuvre la modélisation par éléments finis, a permis de valider notre méthodologie, mais nous a aussi révélé ses possibles limites. La tomographie de dureté a aussi été réalisée sur cellules vivantes et a montré une différence significative entre cellules natives et cellules dont le cytosquelette est dépolymérisé.

Lors du début de ce projet, aucun outil de traitement de données nous permettant de réaliser notre travail n'existait sur le marché. Le développement du programme informatique fut une part significative de notre travail. De son développement s'est constitué une toolbox (une collection de fonctions), disponible pour de futures programmes informatiques. Un part non négligeable du développement a consisté en la réalisation de la documentation de la toolbox, reportée en appendice C. Ce programme nous a finalement permis de caractériser de manière précise et fiable les propriétés élastiques de la membrane.

Mots clés : neurones, hippocampe, rafts, cholestérol, microscope à force atomique, protéines GPI, actine, dureté, tomographie de dureté, propriété mécanique

Remerciements

Un travail de thèse est avant tout un travail d'équipe où idées et certitudes sont constamment échangées et remises en cause. C'est pourquoi je profite de la place qui m'est laissée pour remercier toutes les personnes impliquées de près ou de loin dans ce projet.

Je voudrais tout d'abord remercier mon co-directeur de thèse, le Pr. Stefan Catsicas pour m'avoir accueilli dans son laboratoire, pour sa confiance et pour la liberté qu'il m'a accordé durant ces années. Un grand merci au Dr. Sandor Kasas, mon second co-directeur de thèse, pour m'avoir fait découvrir avec autant de patience et de pédagogie le monde de la recherche et, plus particulièrement celui de l'AFM. Je le remercie aussi pour la modélisation par éléments finis utilisé lors du projet "Stiffness Tomography". Merci aussi à Frank Lafont de l'institut Pasteur, sans qui le projet "GPI-domains" n'aurait probablement pas vu le jour. Merci pour ses remarques et nos discussions constructives sur ce projet.

Je voudrais aussi chaleureusement remercier mes collègues du groupe de neurobiologie cellulaire (LNC), Liliane Glausser, qui a préparé la majorité des cultures cellulaire utilisées dans ce travail, ainsi que les doctorant et ex-doctorant, Karina Kulangara, Michel Kropf et Alexandre Yersin, avec qui j'ai eu nombre de discussions très intéressantes. Un merci particulier à Alexandre Yersin qui a effectué, avec Sandor Kasas et Frank Lafont, les expériences préliminaires utilisées pour le projet "GPI-domains". Merci aussi aux membres du laboratoire de neuroénergétique et dynamique cellulaire (LNDC), avec qui nous avons partagés tant de temps lors de nos "lab-meeting"

Une grande majorité de mon travail s'est fait dans le laboratoire de physique de la matière vivante (LPMV) dirigé par le Pr. Giovanni Dietler que je remercie pour son accueil chaleureux, ses remarques constructives et sa motivation sans faille. Je voudrais aussi remercier tous les membres et ex-membres du groupe LPMV, Aleksandra Radenovic, David Viertl, Erika Ercolini, Eva Bystrenova, Francesco Valle, Giovanni Di Santo, Guillaume Witz, Jozef Adamcik, Kanat Dukenbayev, Kristian Rechendorff, Lilia Chtcheglova, Mélanie Favre, Mounir Mensi, Serguei Sekatski, Susana Tobenãs. Un merci tout particulier à Serguei Sekatski pour avoir élaboré le modèle de Hertz simplifé utilisé dans le cadre du projet "Stiffness Tomography" ainsi qu'à Mélanie Favre d'avoir eu le courage de partager mon bureau. Merci aussi à Christine Vuichoud la secrétaire du groupe, au mécanicien Pyrame Jaquet, ainsi qu'à Michel Kessous, le responsable informatique, qui n'ont jamais hésité à se plier en quatre pour nous faciliter la vie.

Je suis aussi très reconaissant envers Mme Michèle Bonnard Giacobino, secrétaire-administratrice du programme doctorale de neurosciences, pour son habileté à transformer de lourdes charges administratrices en de simples formalités.

Je profite enfin de cet espace pour remercier tous mes proches, ancien collègues de biologie et non biologistes, Jean-Marc, David, Eric, Natalie, Laurence, Lucie, Nicolas, Flo, Odile, Steeve, Thierry, Sonia et ceux que j'oublie, ainsi que mes amis d'Ovronnaz que je vois trop rarement, pour tout ces moment passés en leur compagnie. Merci aussi à tous les membres de ma famille qui se plaignent souvent de mon absence, ainsi qu'à Angéline avec qui je partages de superbes moments et qui supporte mes sauts d'humeur connues d'une grande majorité de doctorants.

Finalement, je voudrais remercier chaleureusement les membres du jury de thèse, le Pr. Bruno Samorì, le Pr. Yves Dufrêne ainsi que le Pr. Ralf Schneggenburger d'avoir accepté de juger mon travail.

Acronyms

AFM	Atomic Force Microscope
BSA	Bovine Serum Albumin
DRM	detergent resistant membrane
EM	Electron Microscope
Er$_{(Rand)}$	Relative Stiffness of randomly selected pixels
Er$_{(GPI)}$	Relative Stiffness of GPI domains
FD	Force-Distance
GFP	Green Fluorescent Protein
GPI	glycosylphosphatidylinositol
GPI-AP	glycosylphosphatidylinositol-anchored protein
MeCD	methyl-β-cyclodextrin
SEM	Standard Error of Mean
ST	Stiffness Tomography
STM	Scanning Tunneling Microscope
TCZ	Transient Confinment Zone
TfR	transferrin receptor
WGA	Wheat Germ Agglutinin

Chapter 1

Introduction

1.1 Atomic force microscopy

1.1.1 History

During history, the development of new imaging technologies provided new perceptions of the living world and permited the science to go steps further. The light microscope, invented in the 17^{th} century, was at the origin of giant steps in medicine with the discovery of cells. In the beginning of the 1940s, the invention of the Electron Microscope (EM) opens new area with the description of new sub-cellular organelles and their functions, and with the identification of viruses. The AFM and its ancestor, the Scanning Tunneling Microscope (STM) [Binnig et al., 1982] are such new imaging technologies which are expected to refine our perception of biology. The STM is based on the tunneling effect and requires the sample to be conductive and is therefore not suitable for most of biological samples.

In 1986 the AFM was derived from the STM [Binnig et al., 1986]. Like the STM, it holds a sharp tip which scans the surface. To measure the distance between the tip and the surface, the AFM feels the interatomic forces between the very end of the tip and the surface. Very soon the AFM has been applied to biological materials since most of them are non conductive [Hansma et al., 1988; Drake et al., 1989]. Fatty acids [Marti et al., 1988] and amino acid crystals [Gould et al., 1988] were the first biological molecules to be imaged, followed soon after by the imaging of plants leafs [Gould et al., 1990], viruses [Zenhausern et al., 1992], DNA [Samori et al., 1993], living cells [Kasas et al., 1993] and spores [Dufrene, 2000]. In addition, fundamental bichemical processes like the polymerase activity have also been observed under nearly physiological condition [Kasas et al., 1997].

1.1.2 Principle

The AFM is a microscope belonging to the scanning probe microscope family. Unlike optical or electron microscopes, a physical probe is in, or nearly in, contact with the studied sample.

The physical probe of the AFM is a sharp tip at the end of a cantilever. A standard cantilever is shown in the figure 1.1 (a). The tip presented on this figure has a pyramidal shape with a base of about 4 µm side length and a radius of curvature of 50 nm or less. The spring constant of a typical cantilever varies from 1 to 0.01 N/m. As comparaison, an aluminum sheet of 3 mm long and 1 mm wide has a spring constant of 1 N/m [Hansma et al., 1988].

The force occurring between the tip and the substrate is monitored by the deflection of the cantilever. Several methods exist to detect the cantilever deflection, but the most widely used one is based on the reflection of a laser beam off the cantilever [Meyer and Amer, 1988]. The laser beam ends its path on a 2 or 4 segment photodiode (figure 1.1(b)), and the difference in the illumination of the segments of the photodiode is used by the controlling computer to calculate the deflection of the cantilever. Despite its simplicity, this technique can detect deflections below 0.1 nm [Digital Instrument, 2000; Butt and Jaschke, 1995].

The vertical and horizontal position of the sample is controlled by a piezoelectric scanner. Depending on the microscope type, the sample or the tip is moved by the scanner. The latter is used when the AFM is mounted on an inverted microscope (figure 1.1(c)) in order to free the space used by the optical microscope.

1.1.3 The AFM operating modes

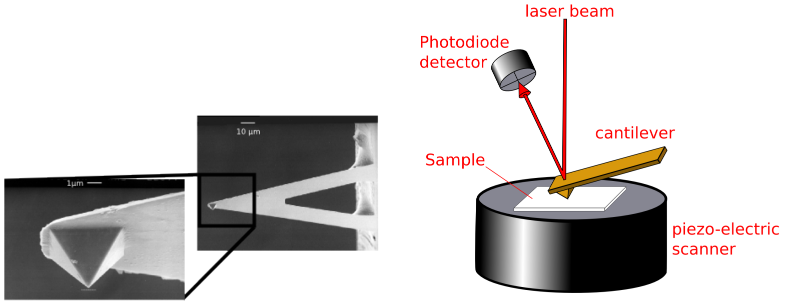
The AFM can be operated in different modes.

Contact mode

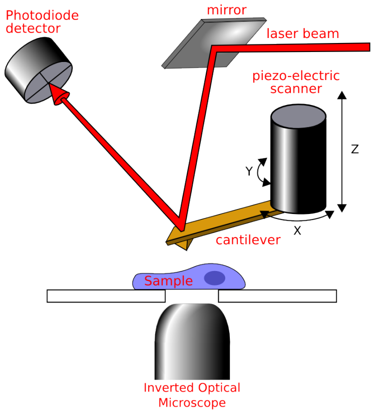
The contact mode, when operated in constant force option, consists in scanning a sample while applying a constant force between the tip and the sample [Binnig et al., 1986]. The microscope moves the tip up and down while the cantilever deflects to keep the tip-sample interaction forces constant. The image is generated by the recording of the piezo position during the scan (Figure 1.2). This mode has been used to image samples like amino acids [Gould et al., 1988], crystallized proteins [Mou et al., 1996], viruses [Drygin et al., 1998] or artificial membranes [Vie et al., 1998]. The contact mode can be enhanced by the error mode (Error Signal in figure 1.2), which records the deflection changes of the cantilever during the scan [Putman et al., 1992]. It gives the information on the efficiency of the feedback control. The advandage of contact mode is its ability to image hard and flat samples with a high resolution whereas it drawback is its tendancy to displace weakly attached structures [Grafstrom et al., 1994]. In addition, soft sample appear lower than they realy are [Weihs et al., 1991].

Tapping mode

In the tapping mode, also called intermittent contact mode, the tip is oscillated near its resonance frequency, to periodically enter in contact with the sample (see figure 1.3). The feedback signal is driven by the oscillating amplitude so if the tip reaches a bump, the oscillation amplitude diminishes, and inversely, if the tip reaches a hollow, the amplitude increases. During the scan, the tip-sample distance is adjusted to keep the cantilever oscillation aplitude constant. The principal advantage of this imaging mode is the reduction of the lateral forces. The tapping mode can be used in air as well as in liquid to image soft and weakly attached samples [Hansma et al., 1993, 1994; Putman et al., 1994].



(a) SEM view of AFM cantilever **(b)** Schema of the AFM



(c) Inverted microscope mounted AFM

Figure 1.1: ***(a)*** *Electron micrography of a standard AFM cantilever. The AFM tip is zoomed. The tip side is about 4 µm size and ended by a curvature radius to about 50 nm.* ***(b)*** *A schematic view of the AFM. The tip is in contact with the sample. A laser reflects off the cantilever, the deflection is detected by the displacement of the laser beam on the photodiode detector. In this schema, the piezo-electric scanner pilots the position of the sample. The photodiode detector and the piezo-electric scanner are connected to a computer (not shown) to integrate the signal of the photodiode and send the feedback to the scanner.* ***(c)*** *A schematic view of an AFM mounted on an optical microscope. The piezo scanner moves the tip to free the space occupied by the optical microscope.*

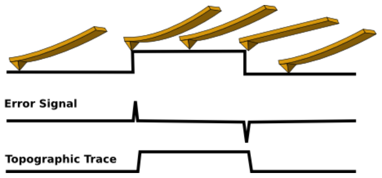


Figure 1.2: *In contact mode, the tip apply a constant force on the substrate and records the change in the piezo height (Topographic trace). The time response of the microscope is not immadiate. The error signal records the deflection change of the cantilever.*

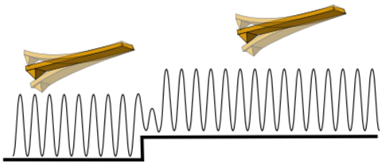


Figure 1.3: In tapping mode, the tip oscillates at a given amplitude. When the tip and the sample come closer to each other, the amplitude decreases. The feedback signal moves then away the sample from the tip, until the initial amplitude is reached.

Force Spectroscopy

Very soon after its discovery, the AFM has been used to perform Force-Distance (FD) measurement [Burnham and Colton, 1989; Weisenhorn et al., 1989]. The figure 1.4, page 4 illustrates the way FD curves are recorded. In the first part of the approach cycle (blue curve), as the tip approaches the surface, the cantilever holds its rest position (straight line 1, 2). When the contact between the tip and the sample occurs, the cantilever deflects upward as the scanner extends (3), and applies a force on the substrate that is proportional to its deflection. During the retraction cycle (red curve), the deflection of the cantilever decreases (4). Just before the lift off, attractive forces between the tip and the sample (capillary forces, van der Waals forces etc...) causes a downward deflection of the cantilever (5). When the retraction force of the cantilever overcomes the adhesion of the tip to the surface, the cantilever returns to its rest position, ending the cycle (6).

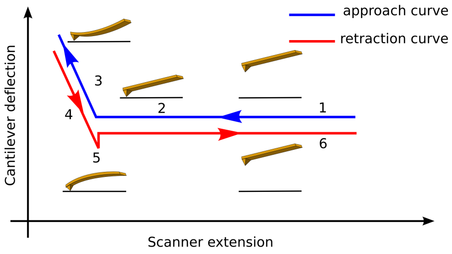


Figure 1.4: During the approach cycle : (1-2) The tip and the surface are off contact. The cantilever is then at its rest position. (3) As soon as the tip and the sample enter in contact to each other, the cantilever deflects itself. During the retraction cycle : (4) The deflection of the cantilever decreases. (5) Attractive forces deflects downward the cantilever until the tip looses contact with the sample. (6) The tip and the surface are off contact, and the AFM is able to perform another cycle.

The force spectroscopy mode of the AFM, also called force volume, simply constits in recording force distance curves allover the sample [Radmacher et al., 1996].

The presence of specific chemical species (receptor) on the surface of the sample can be detected by attaching a ligand onto the tip. This ligand will bind its receptor during the in-contact part of the FD curve (points 2 to 4 in figure 1.5). During the retraction part of the curve, the ligand-receptor complex unfolds (distance c between the points 4 and 5 in the figure), and the cantilevers deflects downwards untill the retraction force of the cantilever overcomes the ligand-receptor binding force. When this bond breaks, the cantilever returns to its rest position. The force needed to break the link can be computed by assuming that the cantilever behaves as an ideal spring. The Hook's law (1.1) describes the force needed to deform it if its spring constant is known.

$$F = k * d \tag{1.1}$$

F being the force in [N], k is the spring constant in [N/m] and d is the deformation of the spring in [m]. The deformation is measured by the vertical distance between the points just before and just after the bound breakage in the force curve (marked as d in the retraction force curve of figure 1.5).

The in-contact part of the FD curve contain information on the mechanical properties of the indented sample and therefore permits to measure its elastical properties [Radmacher, 1997, 2002]. When the tip enters in contact with a hard sample (Figure 1.6, blue curve), the cantilever deflects proportionally to the scanner extension and the tip does not indent into the sample. However, if the sample is soft, the tip indents as the scanner extends (Figure 1.6, red curve). The cantilever deflection is then no more directly proportional to the scanner extension providing a force curve with a rounder shape.

The indentation curve is computed by taking the differences between force-distance curves taken in hard and on soft sample [Tao et al., 1992; Weisenhorn et al., 1993] (figure 3.2 page 25). This curve basically informs on the forces required to indent the AFM tip to a given depth into the sample. By fitting the indentation curve with the Hertz or the Sneddon model [Hertz, 1882; Sneddon, 1965], one can compute the Young's modulus of the sample [Radmacher et al., 1996; Radmacher, 1997; Domke and Radmacher, 1998].

The ability of the AFM to act as a nanoindenter and to probe mechanical properties of samples at the nanometric scale makes this instrument the one of choice to the study of cell mechanical properties.

Several other modes, not used in this study, have been developped, such as the non contact [Martin et al., 1987] or the friction mode [Mate et al., 1987].

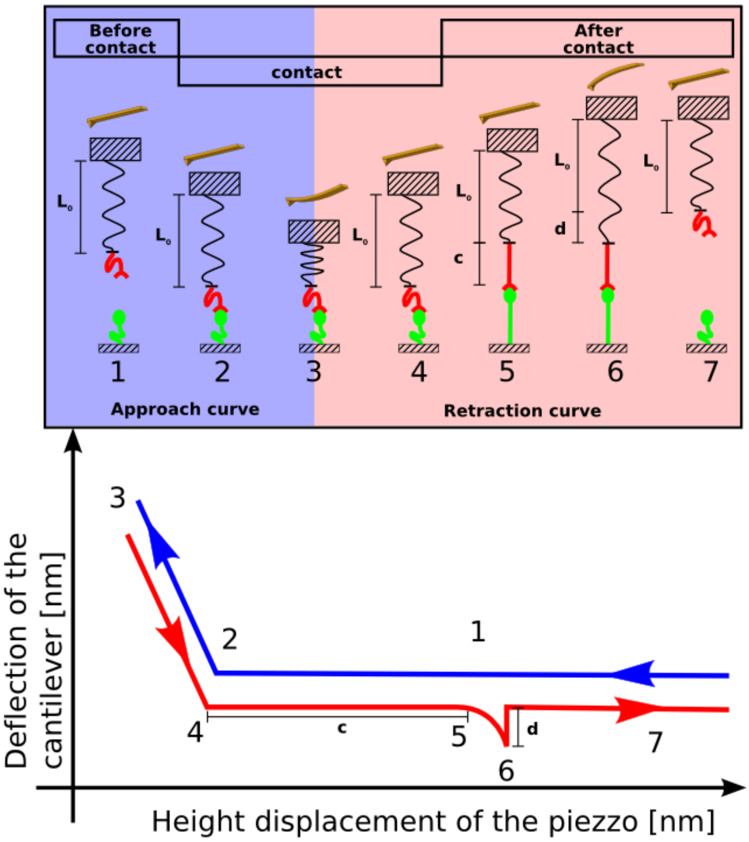


Figure 1.5*: In the top image, the functionalized cantilever is modeled as a spring with a ligand (red) at its end. Its receptor (green) is drawn on the substrate. The blue side represent the approach part of the cycle, and the red side the retraction part of the cycle. The top image is also segmented in three part, the first (left) is before contact, the second (center) represents what happened during the contact and the latter (right) after the contact between the tip and the sample. The bottom image represents the resulting force curve. As the top image, the blue represent the approach and the red the retraction part of the cycle. During the approach cycle : (1) The tip and the surface are off contact. The cantilever is then at its rest position. (2-3) As soon as the tip and the sample enter in contact to each other, the cantilever deflects itself and the two proteins are allowed to interact with each other. During the retraction cycle : (3-4) The deflection of the cantilever decreases. If the ligand and receptor interacts together, they unfold during a certain distance (c) until it beginns to retain the tip (5). At this time, the cantilever apply an increasing force as it tends to move away from the surface until this force is enough to break the ligand-receptor bond (6). The cantilever finally returns at its rest position, ready for a new approach-retraction cycle (7).*

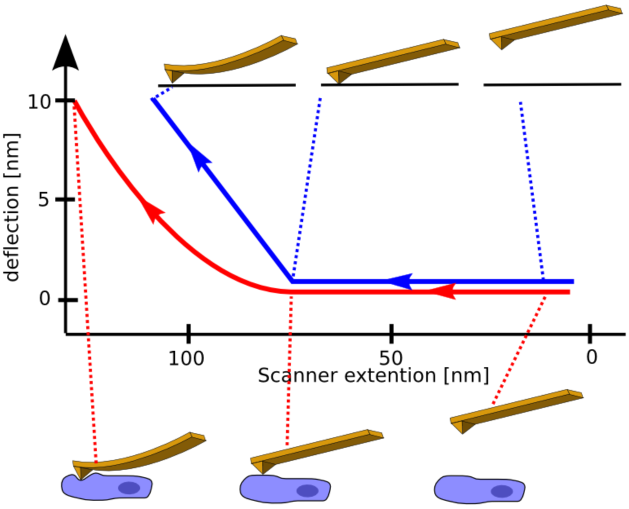


Figure 1.6: *Force curve resulting of indentation into hard (blue) and soft (red) sample.*

1.2 Cell mechanical properties

Physical samples can be caracterized by the Young's modulus and the Poisson ratio. The Young's modulus, also known as elastic modulus, is a measure of the stiffness of a sample. Its definition is the ratio of the rate of change of stress σ with strain ϵ.

$$E = \frac{\sigma}{\epsilon} \tag{1.2}$$

The Young's modulus can vary a lot from a material to another. As an indication, the elastic modulus of some material are reported in table 1.1 [Radmacher, 1997].

When a material is stressed in one direction, it has a tendency to get thinner in the perpendicular direction to the applied stress. The Poisson ration μ is a mesure of this tendency. It is defined as the ratio of the relative contraction strain :

$$\mu = \frac{\epsilon_{trans}}{\epsilon_{axial}} \tag{1.3}$$

ϵ_{trans} being the transversal strain and ϵ_{axial} the axial strain. If the material is incompressible, the Poisson ratio will be 0.5. Most biological materials have a Poisson ratio around 0.3.

Material	Young's Modulus
steel	200 GPa
glass	70 GPa
bone	10 GPa
silk	10 GPa
collagen	1 GPa
protein crystal	0.2-1 GPa
rubber	1.4 MPa
living cells	1-100 MPa

Table 1.1: Elastic properties of typical materials

Many physiologic and pathologic processes alters the biomechanical properties of the tissues they affect. When training, muscles get harder and with aging, skin becomes less elastic. In a wide range of diseases tissue biomechanics abnormalities are observed. Osteoporosis, in which the bones have a low strength and stiffness [Dickenson et al., 1981], osteoarthritis where the vicoelastic properties of cartilage is altered, suggesting a decreased dissipation of elastic energy [Silver et al., 2001], ventricular aneurysm where the viscoelastic properties of ventricular tissues are altered [Kane et al., 1976] is another example where the mechanical properties of tissue plays a key role in the disease. The relationship between tissue mecanics and pathology is used clinically by palpation to detect stiff nodules associated with breast cancer or abnominal hardness due to liver cirrhosis. A number of devices have been developped in order to evaluate the stiffness of soft tissues *in vivo* more quantitatively [Lyyra et al., 1999; Ferguson-Pell et al., 1994].

A non-invasive imaging technique known as elastography [Ophir et al., 1999], based on ultrasound and magnetic resonance imaging methods is able to detect the size and shape of tumors [Ophir et al., 1999; Manduca et al., 2001], can identify stiffness differences in healty tissues [Ophir et al., 1999; Manduca et al., 2001], or detect cardiac deformation due to coronary artery disease [Konofagou et al., 2002].

The mechanical properties changes of the tissue during physiologic and pathologic processes may be visible also at the single cell level. Such changes in the cell mechanical properties of pathologic processes have been reported.

Cancerous cells are reported to be one order of magnitude softer than normal cells [Lekka et al., 1999], an increase in the cell deformability can facilitate the cancer migration. However, the exposure to standard induction chemotherapeutic agents of leukemia cells increased their stiffness and a decreased their passage through microfluidic channels [Lam et al., 2007]. In this case, the cell mechanical properties informs on the efficiency of the drug treatment. For a review in biomecanics of cancer cells, the interessed reader can refer to Suresh [2007]

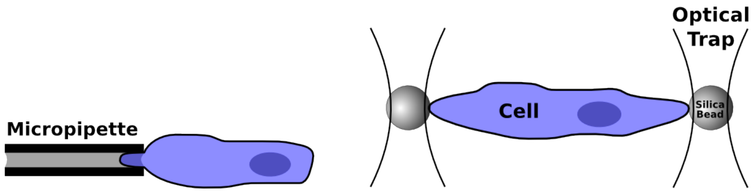
The deformability of erythrocytes is critical for the fluidity of blood in capillaries. The Sickle-cell disease is a genetic mutation that causes a defect in haemoglobin structure. In this disorder, the erythrocytes show a lower deformability. Whereas the treatment with hydroxyurea, used for the medication of the disease, restores the deformability of red blood cells [Brandao et al., 2003]. The stiffness of pathological erythrocyte is also reported to be modified in some disease such as spherocytosis and thalassemias, characterised by abnormal erythrocyte shape [Dulinska et al., 2006].

The mechanical properties were also measured in other organisms. In yeast cells, at the end of the cell division, the mother cell displays a bud scare marking the division site. A significant variation of the cell wall mechanical properties was observed in this region [Touhami et al., 2003]. The bud scare showed a ten times stiffer properties than the surrounding cell surface, consistant with the accumulation of chitin in this area. Mechanical propreties of bacteria can be examined in function of the medium. The turgor pressure, i.e. the pressure of the cell contents against the cell wall, have been measured by Arnoldi et al. [2000] on *Magnetospirillum gryphiswaldense* to be in the range of 85 to 150 kPa depending on the external osmolarity.

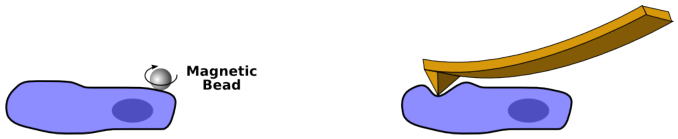
Many tools has been developped to explore the mechanical properties of cells. Some of them will be described here.

Micropipette aspiration The micropipette aspiration technique is used to study the time-dependent deformation of cell (Figure 1.7(a)). The cell is aspirated by a micropipette which inner diameter is a fraction of the nominal diameter of the cell. It can measure the elastic and viscous properties of materials from very soft, like erytrocytes, to stiffer and more viscous cells, such as chondrocytes [Hochmuth, 2000].

Optical tweezers A laser beam is focused to a dielectric object of high refractive index and a radius much higher than the laser wavelength (Figure 1.7(b)). This results in an attractive force between the object and the laser focal point [Ashkin and Dziedzic, 1987]. This phenomenon is a general effect of light on all objects, but is generally negligible on the mesoscopic scale [Williams, 2002]. Two latex beads are positionned at opposite ends of the cell to adhere to it.



(a) Micropipette Aspiration **(b)** Optical Tweezers



(c) Magnetic Twisting Cytometry **(d)** Atomic Force Microscopy

***Figure 1.7**: Schematic illustration of biomechanical assays used to probe cells. (a) illustrates the micropipette aspiration, (b) the optical tweezers, (c) the magnetic twisting cytometry, and (d) the AFM.*

The beads are eventually moved relatively to each other by optically trapping one or both beads [Suresh et al., 2005]. The relative bead displacement exerts a tensile force on the cell up to several hundreds of piconewtons [Mills et al., 2004]

Magnetic traps The magnetic traps are a variation of the optical tweezers where magnetic beads are controlled by an electromagetic field that imposes local force on these beads [Smith et al., 1992]. These magnetic beads, which can be fuctionalized with proteins, are used to apply a local force onto the cell. The advantages over the optical tweezers are that the potential damage on the cell via radiation is eliminated, that out-of-plane rotation of the bead can be considered, allowing to use the magnetic trap as a magnetic twisting cytometry [Wang and Ingber, 1995] (Figure 1.7(c)) and that several beads can be used to stress the cell [Glogauer and Ferrier, 1998].

Atomic force microscope This technique, described in detail in section 1.1 page 1, uses a sharp tip at the end of a cantilever that indents the studied material (Figure 1.7(d)). The principal advantages of the AFM compared to the previously described methods is its spatial resolution. The mechanical properties of nanometric scale sample can be determined [Tao et al., 1992; Laney et al., 1997]. It is the principal reason why we used this instrument to probe the mechanical properties of cell membrane.

1.3 Cell Membrane

The inner part of the cell is a very specialized environment designed to maintain favorable condition for the chemistry of life. This environment has to be controlled and tunned in a very fine way to respond to the outer world. The cell membrane, the frontier between the inner and the outer of the cell, plays a key role in maintaining this fragile equilibrium. Cell surface membrane adhesion plays also a key role in the cell-cell and cell-substratum interaction and, thereby in the cell motility.

1.3.1 The membrane organization

The cell membrane is made of an assymetric lipid bilayer which shows a composition asymmetry from the inner to the outer leaflet. It is composed of phospholipids, glycolipids and cholesterol. Sphingolipids are moslty present in the outer leaflet whereas some glycerophospholipids in the inner leaflet (phosphatidylinositol, phosphatidylethanolamine and phosphatidylserine) [Bretscher, 1973]. This asymmetry in the chemical composition is maintained by an active transport. The localization of the cholesterol is more difficult to determine, its flipping time being about 1 second [Muller and Herrmann, 2002; Steck et al., 2002], but may locates preferentially in the outer leaflet, as it interacts with sphingolipids [Ramstedt and Slotte, 2002].

The organization of lipids occurs also in the lateral dimension, conceptualized by the fluid mosaic model [Singer and Nicolson, 1972]. This model describes the organization of proteins in the lipid membrane and the putative interaction of the lipids with the embedded proteins. Several evidences support the lateral organization hypothesis of the membrane.

1. The first evidence of a lateral lipidic organization came with the study of the transport of newly synthetized sphingolipids in epithelial cells [Simons and van Meer, 1988]. Epithelial cells have bipolar structure where the apical and basolateral membrane have distinct morphologies based on their prominent organic-specific function. The composition of phospholipids is also polarized. The apical membrane is enriched in cholesterol and sphingolipids, whereas the basolateral membrane is enriched in phosphatidylcholine. The newly synthetized sphingolipids have to cluster in the Golgi apparatus before the transport to the apical part of the polarized cell [Simons and van Meer, 1988].

2. The treatment of cell membrane with non-ionic detergent (Triton X-100) at 4°C reveals detergent resistant membrane (DRM) [Brown and London, 1998b]. These DRM have a lipidic composition enriched in sphingolipids and cholesterol, but shows also a protein partitioning with GPI-AP [Skibbens et al., 1989] and doubly acylated tyrosine kinases of the Src family [Brown, 1993; Casey, 1995].

3. Single particle tracking revealed the existence of Transient Confinment Zone (TCZ), where particles are trapped for $\sim$5 to 10 s [Simson et al., 1995]. These TCZ are shown to preferentially trap particles attached to DRM component, such as GPI-AP, and to be dependent on cholesterol [Sheets et al., 1997; Dietrich et al., 2002].

4. Finally, biophysical studies of model membrane shows that pure phospholipids bilayers can exist in different states in the membrane. The solid or "gel" state and two fluid or "liquid" states : the liquid ordered (l_o) and the liquid disordered (l_d) states [Owicki and McConnell, 1980; Smith et al., 1980]. The solid state is not thought to be of physiological relevance, the bilayer being in liquid state at physiological temperature. The l_o phase is characterized by a high degree of acyl chain order. The l_d state can be altered by addition of cholesterol [Yeagle, 1985] and converted to l_o state. Because the cholesterol is a relatively rigid molecule, the sterol causes the acyl chain of lipids to closely pack and the bilayer to be thickened [Kucerka et al., 2007]. l_o and l_d phases can coexist within a single membrane [Li et al., 2001; Ramstedt and Slotte, 2002].

1.3.2 The lipid raft model

The lipid raft model propose that the cholesterol and the sphingolipids of the outer leaflet are not distributed homogeneously in the membrane, but cluster into liquid ordered phase that floats in an liquid disordered bilayer [Brown and London, 1998a]. Proteins present in the lipid bilayer would then locate inside or be excluded from these raft, depending to their physical properties. GPI-AP may be targeted to rafts by small lipid shells (7 nm). These shells may form a relatively stable interaction with the protein and promote their entry into raft [Anderson and Jacobson, 2002].

The biological functional importance of rafts comes from the suggestion that sphingolipids and cholesterol-rich domains, that are located in the outer leaflet, are connected to the components of signal transduction pathways present in the inner leaflet. In particular, G proteins and nonreceptor tyrosine-kinases selectively partition into this inner leaflet part of the raft [Brown, 1993; Harder et al., 1998]. This ability to couple events coming from the outside of the cell with signaling pathways inside the cell would allow rafts to act as a "signaling platform".

Rafts are then defined as liquid ordered domains in cell membrane, enriched in cholesterol and sphingolipids where specialized proteins, such as GPI-AP, have high residency time [Simons and Ikonen, 1997; Brown and London, 1998a].

1.3.3 Role of the rafts

A wide variety of functions are believed to be dependent on rafts. These membrane domains are proposed to play role in the signal transduction, secretory pathway, endocytosis, cell adhesion and motility and are also supposed to be required for pathogen entry and some virus assembly.

Signal transduction

The proposed role of raft-clustering in the triggering of signalling events [Simons and Toomre, 2000] has shed new light on the dynamics of membrane-associated molecular assemblies.

Lipid rafts are involved in the immunoglobulin E (IgE) signalling during the allergic response [Field et al., 1995; Sheets et al., 1999b]. The IgE binds through its Fc segment to the high affinity IgE receptor (FceRI) residing in the plasma membrane of mast cells and basophils. The crosslinking of FceRI is though to increase their raft affinity. This results in an increased phosphorylation by the raft-associated Lyn kinase [Sheets et al., 1999a].

In the nervous system, these domains are believed to be of physiological importance in cell polarization and in the establishment and maintainance of neural-network plasticity. Indeed, raft-associated signalling has been shown to be important for neuronal survival, for membrane polarity and for neuritogenesis [Guirland et al., 2004; Ledesma et al., 1999; Niethammer et al., 2002; Tansey et al., 2000]. For instance, during the establishment of neural networks, the interaction between the GPI-domain, ephrin A, and Ephr mediated forward signalling and synapse formation [Scheiffele, 2003].

Lipid rafts function then as signalling platforms by concentrating positive regulators of signalling

Secretory pathway

The bipolar structure of epithelial and hepatocyte cells is maintained by intracellular machinery that directs newly synthesized material into the correct target membrane. Lipid rafts is one of the actors that play a role in the apical sorting of lipids and membrane proteins. The trans-Golgi network represents a sorting station, were apically targeted proteins are supposed to segregated into rafts [Simons and van Meer, 1988; Simons and Ikonen, 1997].

Endocytosis

Endocytosis is a complicated phenomena that comprise several different and relatively well defined routes of internalization. The clathrin-mediated endocytosis is raft independent.

In the other side, the caveolae-mediate endocytosis is known to be raft dependent, caveolae being categorized as raft subcategory [Simons and Toomre, 2000]. The caveolae formation is dependent of caveolin-1 and caveolin-3 [Drab et al., 2001; Galbiati et al., 2001]. The transport of caveolin to the plasma membrane is supposed to be dependent on its ability to associate with lipid rafts [Ren et al., 2004].

Finally, there exist a third pathway that is clathrin- and caveolae-independent mechanisms which seems to be intimately linked to rafts. GPI-AP was shown to be internalized via such a clathrin and caveolae-independent pathway [Sabharanjak et al., 2002] during the protein recycling.

Cell adhesion and migration

Cell adhesion is critical for the cell-cell interaction and for the tissues and organ morphogenesis. Many adhesion proteins are GPI-AP and are thereby partitioned into membrane rafts.

Individual prot-prot *trans*-interaction (occurring between cells) measured are relatively weak [van der Merwe and Barclay, 1994], a crucial property for dynamic and transient cell-cell interaction. Stable cell adhesion require strong binding forces. This can be reached by oligomerization into zippers of protein adhesion. These oligomers are stabilized through *cis*-interaction (occurring within proteins of the same membrane). The axonin-1, a GPI-AP that mediates cell adhesion during neurogenesis shows a zipper mechanism for neural adhesion [Freigang et al., 2000].

Disorders

Rafts have also been implicated in several disorders of the nervous system, for instance in amyloidogenic processing of the Alzheimer beta-amyloid precursor protein [Ehehalt et al., 2003]. In addition, the prion protein possesses a GPI anchor, which is crucial for its trafficking and hence for its pathogenic conversion [Brugger et al., 2004; Taraboulos et al., 1995]. Furthermore, several neurotoxins require raft-associated lipid species and/or proteins for their binding and entry [Lafont et al., 2004]. These include the botulinum [Lalli et al., 2003], the cholera [Shogomori and Futerman, 2001] and the tetanus toxins [Herreros et al., 2001]. Membrane proteins of several enveloped viruses also localize into rafts. The influenza transmembrane proteins, neuraminidase and hemagglutinin [Barman and Nayak, 2000; Scheiffele et al., 1997], the GP protein of Ebola and Marburg virus [Bavari et al., 2002], and the transmembrane glycoprotein complex Env of HIV-1 [Nguyen and Hildreth, 2000] are also known to partition in rafts.

1.4 Aim of the project

In this thesis, we used Atomic Force Microscope (AFM) to investigate the mechanical properties of lipid rafts on living hypocampal neurons. The instrument was used because of its capabilities to operate in nearly physiological conditions, to asses the mechanical properties of living sample at high resolution, and to target defined proteins on a living samples.

The first part of the project constisted in the development of a specific force-volume data processing software.

The second part of the project addressed the mechanical properties of lipid rafts onto living cell membrane. The measurments were performed by recording force-volume data on living cells with a tip coated with aerolysin, a toxin specific for lipid rafts domains. Numerous controls were conducted to confirm the specificity of our measurments.

The last part of the project consisted in the development of a new imaging mode which we called “stiffness tomography”, and which permits to reveal structures hidden in the bulk of the sample.

Chapter 2

Materials and methods

2.1 Biochemistry

2.1.1 Proteins

Aerolysin and VSG117 was a gift from Pr. F Gisou van der Goot, produced and purified as previously described [Abrami et al., 2002]. Wheat Germ Agglutinin (WGA) and Bovine Serum Albimin (BSA) were purchased from Sigma. The Alexa 546-conjugated aerolysin mutant (ASSP) was a gift from Pr. F Gisou van der Goot, produced and purified as previously described [Fivaz et al., 2002].

2.1.2 Antibodies

Antibodies against the folowing antigens were used : Polyclonal, L1 (IF 1:4000; a gift of Dr V. Lemmon, USA); MAP2 (IF 1:300; Sigma); anti-H (from J.M. Brunner, UNIL); Monoclonal, anti-transferrin receptor (TfR)(IF 1:500; Zymed)

We also used: Cy3-, Cy5-coupled (IF 1:200; Jackson Immunoresearch) and Oregon-Green-coupled (IF 1:200; Molecular Probes).

2.2 Cell cultures

2.2.1 293T Cells culture and transfection

293T cells were grown in a humidified incubator at 37°C and 5% CO_2 in full medium of D-MEM (Invitrogen 41966) with 10% foetal calf serum and 1% Penicillin. For Calcium Phosphate transfection, 2 million cells per 9 cm dish were plated 8-10h before transfection. 5 µg of DNA was mixed with H_2O and 50 µl of 2.5 M $CaCl_2$ (final concentration of 250 mM) for a final volume of 500 µl and which was then added dropwise to HBS 2X, pH 7.1 (280 mM NaCl, 50 mM HEPES (base), 1.5 mM Na_2HPO_4). The complex DNA-Phosphate-Calcium was then added onto the cells over night. The next day, cells were washed once with full medium and replaced then by fresh medium.

For the expression of GPI-GFP, 293T cells were transfected with Aequora GFP. GFP was fused to the GPI-moiety of the folate receptor, which was cloned into the pJB20 vector (a kind gift from C. Zurzolo, Institut Pasteur, Paris). 293T cells were cultured in DMEM (invitrogen 41966) containing 10% foetal calf serum and 1% penicillin at 37°C in a humidified, 5%-CO2 atmosphere. The cells were plated within Petri dishes at a numerical density of 30'000 cell /cm2. After an 8- to 10-hour growth phase, the cells were transfected with the DNA-calcium-phosphate complex. This was prepared by mixing 50 µl of 2.5M CaCl2 with 450 µl of water containing 5µg of DNA. This solution was then added dropwise to souble-strength HBS [280 mM NaCl, 1.5 mM Na2HPO4, 50 mM HEPES (pH 7.1)]. After an overnight incubation with the complex, the cells were rinsed thoroughly with DMEM.

2.2.2 HeLa Cells culture and transfection

Non confluent cell cultures of Hela cells were analysed using aerolysin functionalized tips using similar protocol as that used for neurons and 293T cells. HeLa cells were transiently transfected with a plasmid encoding GPI-GFP (see above) using FuGene (Roche Applied Science) as transfection reagent. Cells were used either after overnight or after 24h transfection.

2.2.3 Hypocampal neurons

Hypocampal neurones derived from rat embryos were prepared and cultured as previously described [Morgenthaler et al., 2003]. The cells were plated within Petri dishes at a numerical density of 2500 / cm^2 and were maintained in K5 medium [128 mM NaCl, 5 mM KCl, 2.7 mM $CaCl_2$, 1 mM $MgCl_2$, 10 mM glucose, 20 mM HEPES (pH 7.4)] at ambient temperature. Each experiment was initiated 15 minutes after inserting the Petri dish into the AFM. This delay was required for the thermal equilibration of the cantilever. When required, a 5 mM solution of MeCD in K5 was prepared and introduced into the incubation chamber using a home-made set-up Kasas et al. [2000b] to yield a final concentration of 2.5 mM (see figure 2.2). Cytochalasin B was likewise prepared in K5 to yield a final concentration of 5 µM.

2.3 Immunochemistry

Hypocampal neuron cells were cultures as described in section 2.2.3 and seeded onto glass coverslips at a numerical density of 5'000 / cm^2 and cultured for 10 days. To label the GPI-anchored proteins, the cells were incubated with the Alexa 546-conjugated aerolysin mutant ASSP at 500 ng/ml for 1hour at 4°C and at 37°C for 5 minutes Fivaz et al. [2002]. The cells were then fixed in methanol for 6 minutes at -20°C. Thereafter, they were incubated with either a dendritic monoclonal antibody against MAP2 (Sigma) or a polyclonal antibody against L1 (kindly provided by V. Lemmon, Case Western Reserve University, Cleveland). They were then treated with Oregon-Green-conjugated (Jackson ImmunoResearch Laboratories, Inc.). The cell nuclei were stained with Hoechst's reagent.

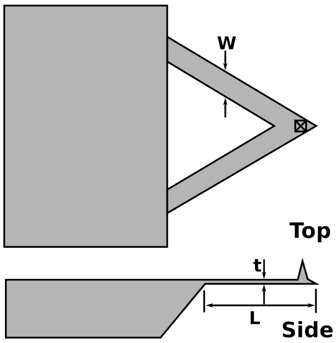


Figure 2.1: *schematic view of a silicone-nitride cantilever (www.veecoprobes.com). L= 196 µm, W = 23 µm, t = 0.6 µm.*

2.4 AFM

2.4.1 AFM tip calibration and coating

Calibration of the cantilever

We used standard triangular silicone nitride cantilevers from Veeco (dnp) with nominal values of L = 196 µm length, W = 23 µm width and t = 0.6 µm thick (figure 2.1). These cantilevers are coated with chromium (layer of 15 nm) in order to provide an optimal laser reflection. The tip was calibrated using the Nanoscope 4.43 facility. The spring constant of the tip with nominal spring constant of 0.06 N/m varied form 0.04 N/m to 0.08 N/m.

Coating protein to the tip and the mica

The coating of mica plates and cantilevers was performed using an established protocol which preserved the functionality of the proteins [Allen et al., 1997, 1999; Yersin et al., 2003]. The proteins were deposited at a concentration of 1 µg/ml. Aerolysin was used as a monomer.

The tip was first washed by dipping in water and soap during 15 minutes. After rinsing three times in fresh water, the tip was immersed into a drop of glutaraldehyde during 15 minutes. After another tree times rinsing in fresh water, a drop of proteins was deposited onto the tip during 15 minutes. Then the tips was finally rinses three times with TBS 1x to block the glutaraldehyde [Allen et al., 1999]. The prepared tips can be directly used or be stored one week at 4 °C.

2.4.2 In vivo measurements

Cells were prepared as described in "Cell Culture" (section 2.2, page 17) in 35 mm petri dishes the day before measurements. Cultures were then washed three times with PBS, and 2 ml K5 were added as medium buffer. Petri dishes were mounted under the AFM for measurements. All measurements were done at room temperature For time lapse experiments, the setup was made by two

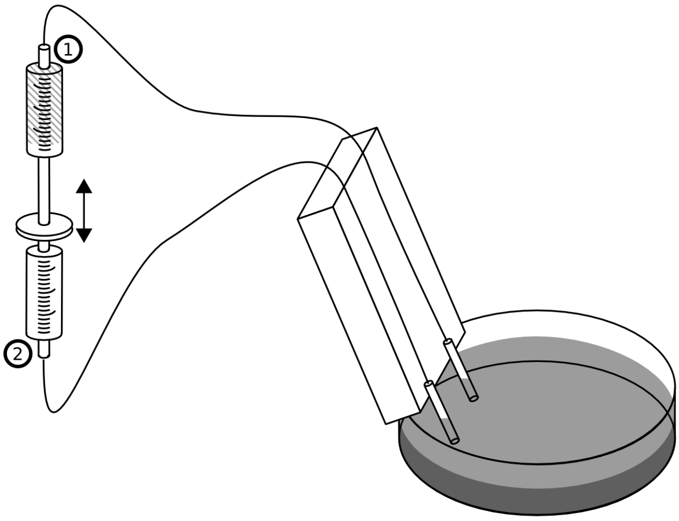


Figure 2.2: *Illustration of the setup used for buffer injection during AFM measurements.*

syringe mounted as described in figure 2.2. Syringe 1. was full of medium to inject. When injecting the additional medium, the same volume of the buffer medium with syringe 2 was removed. To prevent noise record during medium injection, this step was executed between the record of two scans.

2.4.3 Force Volume measurements

The force volume image mode of the AFM was used with a trigger threshold of 50 nm and a ramp size of 1 µm. The size and resolution of the scan were 0 nm and 16×16 pixel square for calibration measurements, done on a zone free of cells, and, if not explicitly specified, 2×2 µm with 32×32 pixel for all measurements on cell. The first positioning of the tip was done off contact with the microscope's motor, then the tip was set to enter in contact with the surface. During the first scan, the scanner position was adjusted to the desired position with the "x offset" and "y offset" of the scanner.

2.4.4 Force-Volume analysis software

The force volume mode of the AFM generates a huge amount of files and force curves. In order to analyze it, we had to automate the processing. A software was written under Matlab®7.1.0 on a GNU/Linux platform. All files of a single experiment were loaded and merged in a single metafile. A description of the software can be found in section 3.2, page 23.

Stiffnesses were computed by fitting the very first 50 nm of the indentation curve. The Hertz model chosen for the fit correspond to a sphere with a radius of 40 nm.

Protein-protein interaction detection was done through the home-made software powered by a fuzzy logic algorythm (see section 3.2 page 23). The detection parameters were tuned for each experiments and the same parameters were used for the whole experiment. Force distance curves that contained the characteristic signature of a protein-protein interaction were marked for further analyses. Results were stored in the aex file generated.

Relative Young's moduli were computed by dividing each marked pixels by their neighbors. All the relative Young's moduli of a scan were then averaged to give the mean relative stiffness of a whole scan. A more detailed explanation of the relative Young's modulus can be found on section 3.2.4 on page 26. The relative Young's moduli of pixels of interest were then computed for each scans of the time lapse and stored in a text file. Two sets of relative Young's modulii were generated. One considering pixels containing protein-protein interaction as marked pixels, and a second with randomly selected pixels considered as marked pixels. The latter value was used as a numeric negative control.

The result of each experiments was stored in a text files and was merged with an external home made software. The final file, containing all time lapses, was analyzed with a spreadsheet (Gnumeric) in order to draw graphs and make statistics.

2.4.5 Data representations

The three dimensional views were done with Blender, an open source 3D creation software. A python script was written to automatically import images, topography and events positions (see appendix A page 65 for more information)

Chapter 3

Post Processing software

3.1 Introduction

Force volume experiments generates a huge amount of data that requires a specific postprocessing step to give comprehensive results. Since the AFM manufacturers do not provide dedicated software to process this type of data, we developed our own software tools. The software, written in Matlab®, automatically detects binding-unbinding events, map their location onto the topography of the sample and computes the stiffness of every pixel of the sample. This chapter will describe the software we developed during this thesis.

3.2 Results

The software was written in Matlab®1.7.0 under a GNU/Linux operating system. In order to be extensible, the software was written with an object oriented programming method. Each force volume file of an experiment is loaded as an object, called "forceVolume" object. Typical experiments done during this thesis were made of timelapses composed of several force volume files. An experiment is then composed by all the forceVolume object of such a timelapse. The software saves these experiment as a whole to a zipped file containing XML files that describes each forceVolume objects. This software is composed by 152 function and modules and counts $\tilde{1}8000$ lines.

3.2.1 Zero Force Image

The zero force image, or topographic image, of the sample is constructed by recording the piezzo position when the tip touches the sample whitout applying any force on it.The point of contact between the tip and the sample is detected on the approach curve and is defined after a linear fit of the "off contact" part of the curve. The spot where the FD curve "takes off" the linear fit of the horizontal part of the curve is considered by the software as the contact part (figure 3.1).

At the end of this computation, the zero force image is stored as a matrix composed of the altitude of each indented pixels. For a more detailed explanation in the software side, see appendix C, "Compute Young moduli" section.

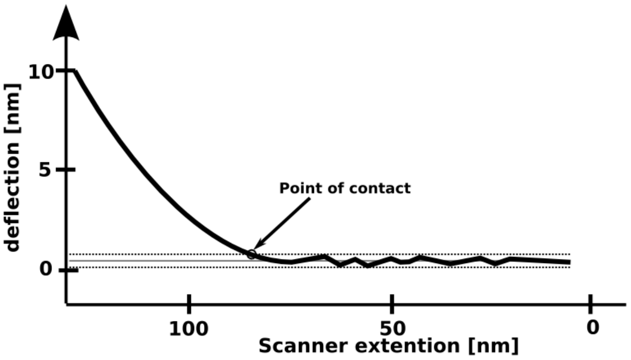


Figure 3.1: *Determination of the point of contact. The thin black line represents the fit of the "off contact" part of the force curve. The two dotted lines represents the noise detected. The arrow points where the algorithm finds the point of contact.*

3.2.2 Stiffness Computation

The stiffness measurement beginns by the detection of the point of contact between the tip and the sample as explained in the previous section. It is followed by the computation of the indentation curve. This step is performed by calibrating the slope of the "on contact" part of a force curve recorded on a hard substrate such as glass or mica. The indentation curve is finally obtained by substrating this slope from the in contact part of the force curve (figure 3.2).

The next step consist on segmenting the indentation curve. The number and size of each segments to extract can be chosen from the graphical user interface. In the figure 3.3, we fragmented the indentation curve into 10 segments, beginning from the point of contact I_0. ΔS_i is defined as the segment located between the indentation points I_i and I_{i+1}. The force needed to indent this portion of curve is then ΔF_i. For each segment ΔS_i, from the point of contact, to the end of the indentation, a force ΔF_i can be associated.

In the last step, the Young modulus of each segments of the curve is calculated by fitting it to the Hertz model [Radmacher, 1997, 2002] according to equation 3.1 for the sphere model and 3.2 for the cone model.

$$F = \frac{4}{3}\frac{E}{1-\nu^2}R^{\frac{1}{2}}\delta^{\frac{3}{2}} \tag{3.1}$$

$$F = \frac{2}{\pi}\frac{E}{1-\nu^2}\tan(\alpha)\delta^2 \tag{3.2}$$

Where ν is the Poisson coeficient of the material, δ the indentation distance into the material, E the Young Modulus, R the radius of the sphere and α the semi-opening angle of the cone in radian.

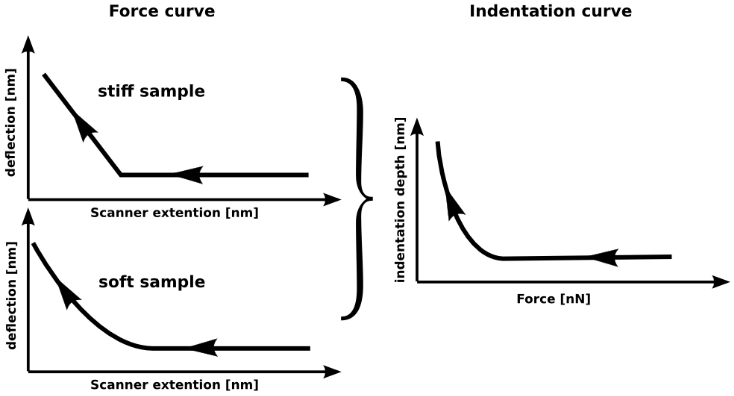


Figure 3.2: *Computation of the indentation curve. By removing the slope of the "on contact" part of the force curve recorded on stiff sample from the force curve recorded on the studied sample (soft), the indentation curve is generated.*

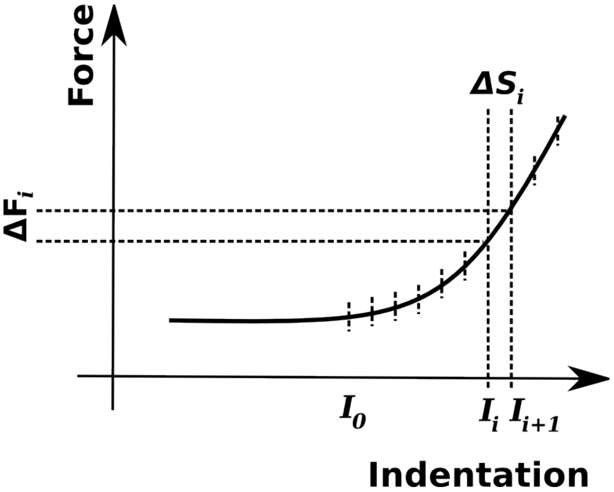


Figure 3.3: *The indentation curve is divided in several segments of defined depth,* ΔS *from the point of contact* I_0 *to the end of the indentation curve. At the depth i, the force needed to indent into the segment* ΔS_i *is* ΔF_i.

By passing E on the right hand side, we obtain the Young modulus :

$$E = \frac{3}{4\sqrt{R}} \frac{F}{\delta^{\frac{3}{2}}} (1 - \nu^2) \tag{3.3}$$

$$E = \frac{2}{\pi \tan(\alpha)} \frac{F}{\delta^2} (1 - \nu^2) \tag{3.4}$$

Instead of applying the Hertz model to the whole curve, we apply it on each segments ΔS_i, which gives for the sphere and the cone model respectively :

$$\Delta E_i = \frac{3}{4\sqrt{R}} \frac{\Delta F_i}{\Delta S_i^{\frac{3}{2}}} (1 - \nu^2) \tag{3.5}$$

$$\Delta E_i = \frac{2}{\pi \tan(\alpha)} \frac{\Delta F_i}{\Delta S_i^2} (1 - \nu^2) \tag{3.6}$$

Where ΔS_i is the length of the i^{th} indentation segment and ΔF_i is the additional force needed to indent the i^{th} segment.

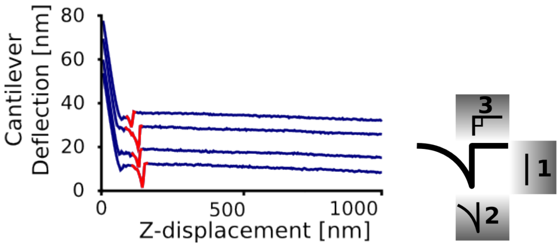
3.2.3 Event detection

The protein-protein interactions are detected on the retraction part of the FD curves (see figure 1.5, page 6). The shape of the event is easily recognized by a human, but quite hard to be cathegorized by a computer (figure 3.4(a)). A characteristic event is composed of a vertical segment at the rupture of the interaction (number 1 in the figure 3.4(b)), a right angle shape on the right hand side of this vertical segment and a characteristic v-shape in its lower part (figure 3.4(b), number 2 and 3 respectively). The strategy we adopted was to use a Fuzzy logic algorithms [Zadeh, 1965]. It detects the "events" and attributes them a grade between 0 and 1 according to their similarity with an "ideal" shape [Kasas et al., 2000a]. The Fuzzy detection module introduced in this work is an adaptation of the previously developed one by S. Kasas.

Attributing such grades to each detected events permits to solve, in a reproducible way, the paradox one is facing in this type of experiments and wich consist in considering the maximum number of events to obtain a good statistic for the calculation of the interaction forces, and the wish to exclude artefacts and noise from the data set.

3.2.4 Relative Stiffness

The stiffness of a cell varies a lot allover its surface. It is mostly due to the inhomogeneous distribution of its cytoskeleton, its shape and the more or less random distribution of its organels. To get rid of this effect, we decided to consider only the relative stiffness of the considered pixels. The relative stiffness is computed by comparing the stiffness of the pixel of interest with the stiffness of the pixels located in its vincinity. In order to diminish the time between two measurement (see figure 3.5), we considered only the neighboring pixels located along the fast scanning axis of the microscope.



(a) Event in curve **(b)** Event shape

Figure 3.4: ***(a)*** *Force-distance curves recorded on mica coated with VSG using an aerolysin-funstionalized tip.* ***(b)*** *Characteristic shape of protein-protein interaction. In 1 is described the vertical segment, in 2 the v-shape and in 3 the right angle part of the event in the force curve.*

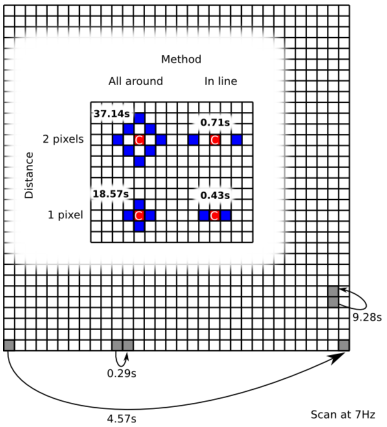


Figure 3.5*: Explanation of the time required to measure relative stiffness values. “All around” depicts the time required to measure relative stiffness when considering pixels located in all direction (blue) around the pixel of interrest (red). “In line” depicts the time required to measure relative stiffness when considering only pixels located in the fast scan axis (blue) around the pixel of interrest (red). The times shown are calculated with a scan rate of 7Hz.*

3.2.5 Merging the results

On the zero force topography, image of the stiffness map and the event position can be merged into one single image as depicted on figure 3.6. This image covers a surface of 2 µm per 2 µm and represents the neurite of a living neuron. The zero force image is represented in 3D. The stiffness is mapped as a color scale where soft regions are colored in blue and hard regions in red. Regions colored in white depicts no measurable indentation which occurs on the hard substrate. The protein-protein interaction events are marked with the red arrows.

These three dimentional images were made with Blender, an external 3D rendering software. Its scripting capabilities permited to automate the data importation. A complete description of this script can be found in the appendix A.

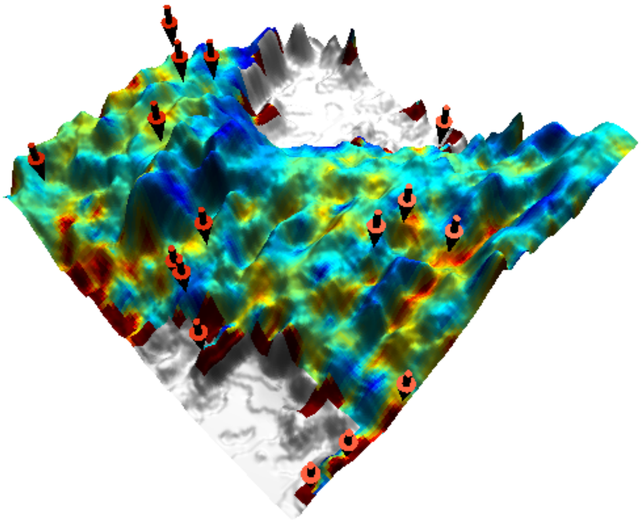


***Figure 3.6**: Scan in force-volume of a neurite. The color scale represent the stiffness (blue: soft, red : hard). The arrows represent the presence of a protein-protein interaction.*

3.3 Conclusion

The automation of the postprocessing step allows to dramatically reduce the time required to display the results.

The software has been developped as an assembly of functions (a toolbox). Every single function is therefore independant and can be reused as a building block for an other software. It considerably reduces the developing time and increases its readability. The toolbox architecture permits also to introduce new function without altering the software stability.

In order to make the toolbox available to other developpers, a non negligible part of the software development was dedicated to its documentation. This developer's documentation is available as an appendix in page 91. It is splitted in several parts. The first, called "software walking" describes the software operationg mode. The second, called "Function definition" explains and describes each functions of the toolbox in detail. The two next sections describes the variables of the software. The last one describes the AFM Exchange XML file format (.aex), the format used to store the result.

Chapter 4

Mechanical properties of GPI domains

4.1 Introduction

Many approaches have been developed to characterize the heterogeneity of membranes in living cells. In the present study, the elastic properties of specific membrane domains in living cells were characterized by AFM.

We focused on GPI-APs, which play important roles in membrane trafficking and cell signalling under both physiological and pathological conditions, and which are known to partition preferentially into rafts.

The characterization method basically consisted in depositing a specific ligand for GPI-AP, the Aerolysin. The surface of a neuron was eventually explored by force volume imaging. The spots where the protein fixed onto the tip interacted with those present onto the neurons were considered as glycosylphosphatidylinositol (GPI) domains. The stiffness of these domains was finally compared with the stiffness of the spots located in their vicinity (section 4.2.5). In order to selectively measure the mechanical properties of the membrane without being influenced by the cytoskeleton, we only considered the very beginning of the indentation curve (section 4.2.2).

Since several methodological steps described above were never used before, we conducted numerous control experiments to validate our methodology.

1. In a first step, we checked the ability of the AFM to detect the GPI anchored proteins on the cell membrane (section 4.2.1) and compared the results to fluorescence images obtained with standard labeling (section 4.2.2).

2. Since the typical timelapse experiments done in this study can last up to two hours, we checked the tip sensitivity to contamination during scans of 2h at the top of living cells (section 4.2.3).

3. We studied to which extend the cholesterol extraction affects the relative stiffness of the GPI domains (section 4.2.6).

4. In a next step, we study to which extend the bond between the proteins present onto the AFM tip and those located in the membrane influence

the relative stiffness measurements (section 4.2.9).

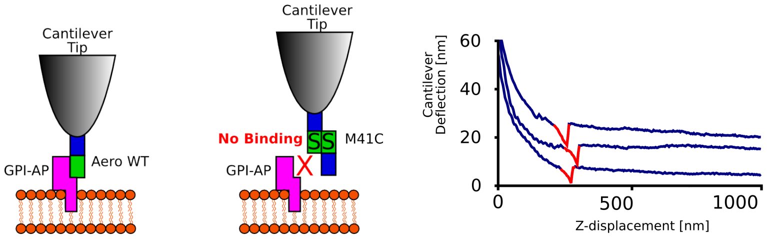
5. To exclude a hippocampus neurons specificity, the measurements were repeated on two different cell lines (section 4.2.10).
6. In a final control, we tested the influence of a possible protein diffusion during the measurements (section 4.2.11).

4.2 Results

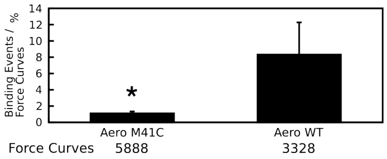
4.2.1 Binding specificity of aerolysin

In order to observe the presence of lipid rafts on hippocampal neurons, we targeted GPI-APs, which are known to partition in rafts microdomains [Skibbens et al., 1989]. The surfaces of the living cells were then explored using the AFM in the force-volume imaging mode (see section 1.1.3, page 4) and in conjunction with tips coated with the aerolysin, a GPI-AP specific binding bacterial toxin from *Aeromonas hydrophila* (figure 4.1(a)). Since the GPI anchor is common to all GPI-APs, this detection method is therefore not protein-specific. For a detailed description of the protein-protein unbinding event detection, see the section 3.2.3. Typical retraction force curves recorded on cell and that contains protein unbinding events is shown on figure 4.1(b), with events highlighted in red color.

The capability of the AFM to detect specific binding between the aerolysin and the GPI anchor was assessed by comparing the binding capacity of the wild type aerolysin to a mutant one (M41C) which lacks its affinity to the GPI anchor (figure 4.1(c)). The experiment was done on neurites of living neurons. AFM tip coated with aerolysin shows a binding efficiency of about 8%, whereas M41C, which differs only on the GPI-anchor binding capability, has only 1% of binding efficiency. The result clearly shows the binding specificity of the aerolysin to the GPI-anchor, and the AFM capability to detect such specific binding-unbinding event.



(a) Experimental setup **(b)** Force curves binding events



(c) Aerolysin binding specificity

***Figure 4.1**: **a)** Experimental set up describing the tip coated with either the aerolysin wild type (Aero WT), which binds GPI domains on the cell membrane, or with the mutant M41C, which does not bind to GPI domains. **(b)** Force-distance curves examples. Force-distance curves recorded on neurites with binding – unbinding events highlighted in red. **c)** Number of binding-unbinding events / force curves with aerolysin (Aero WT) – coated tip and with the non GPI domain binding mutant Aero M41C – coated tip tested on hippocampal neurons. The number of force curves (FC) analyzed is indicated. Data are means ± SEM; asterisk, $p < 0.03$, two-tailed t-test.*

4.2.2 Distribution of GPI-anchored proteins

The presence of GPI-APs within all neuronal compartments, viz., axons, soma and dendrites, was established by fluorescence microscopy and AFM.

For fluorescence microscopy, cells were stained with the aerolysin conjugated with Alexa 546 to mark the distribution of the GPI-APs along the neurons (left images and red channel in right image, figure 4.2). Neurons were stained either with the dendritic marker MAP2 (center and green chanel right images, figure 4.2 (a)-(b)) or with the antibody against L1, an axonal marker (center and green channel right images, figure 4.2 (c)-(d)). The Hoechst's reagent revealed the nuclei to localize the soma (blue channel right images, figure 4.2 (a) and (c)).

The figure 4.2 shows localization of aerolysin within dendrites (revealed by MAP2, (a) and (b)), within axon and soma (revealed by α-L1 (c) and (d)) providing that, GPI-APs are not confined to a compartment of the neuron but are clearly located on soma, axon and dendrites.

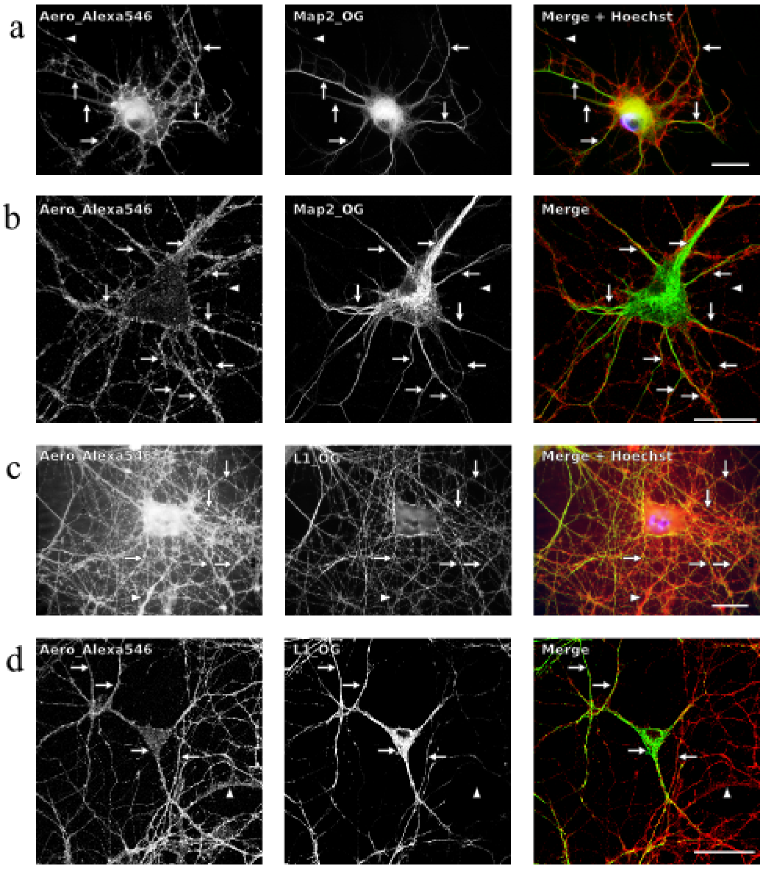


Figure 4.2: *Distribution of aerolysin on the membranes of cultured hippocampal neurons. Arrows indicate regions of co-localization. Dendritic marker MAP2 and axonal marker L1. Arrowheads denote aerolysin-positive neurons that are MAP2- or L1- negative. Bars = 20 μm.*

The GPI-anchor distribution detected by the AFM with aerolysin coated tip showed similar results to those observed by fluorescent microscopy. Moreover, the AFM detection revealed a difference in the number of unbinding event according to whether the axon, the dendrite or the soma were examined (figure 4.3). We found that 58% of the detected unbinding event are located on the axon, 35% on the dendrite, and only 7% on the soma (N_{cell}=4, p-values < 0.03, two-tailed ttest).

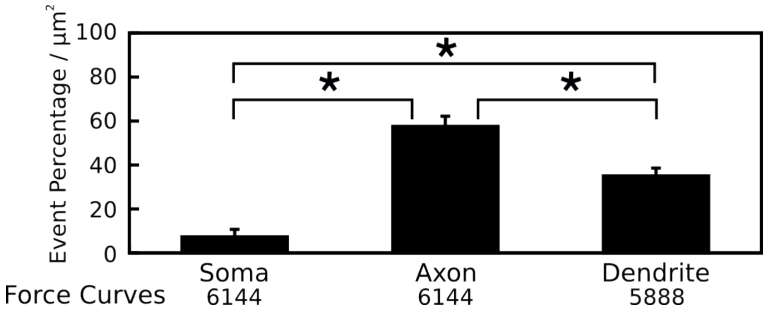


Figure 4.3: *Percentage of GPI-anchored binding events per surface unit on soma, axon and dendrite. Data are means ± SEM,* n_{cell}*=4; The stars indicates the p-values* < 0.03 *; two-tailed t-test.*

4.2.3 Tip contamination

In these experiments, we tested whether the aerolysin coated AFM tip was contaminated by the biological sample during the scan and whether the binding capacity of the aerolysin on the tip was conserved after scanning the biological sample (figure 4.4). We used the glycoprotein VSG117 from Trypanosoma brucei, Bovine Serum Albumin (BSA), and the Wheat Germ Agglutinin (WGA) as a control for tip contamination by cell membrane debris. All reagents were used at 1 µg/ml. Results are expressed as binding events / FD curves before and after scanning the biological sample. BSA was used to check unspecific bindings before and after neurite scanning. WGA, a sialyc acid-binding lectin, was used to follow the tip contamination by sialic acid residue from proteins originating from the neuronal membrane. To check a possible alteration of the binding capacity of the tip to GPI-APs, VSG117, a GPI anchored variant surface glycoprotein of Trypanosoma brucei, a well-characterized GPI anchored aerolysin binding protein was used as a positive control ($p<0.03$, two-tailed t-test vs. BSA and vs. WGA before scanning the biological sample). After scanning the biological sample, event though the tip could be contaminated by sialic acid residues, the binding to the GPI-AP VSG117 was kept similar.

4.2.4 Determining the local stiffness inhomogeneity

The Young's modulus was calculated by fitting the indentation curves obtained from the force-volume files to the Hertz model (see section 3.2.2, page 24). At zero force, topographic data (i.e., the three-dimensional image of the sample) were obtained by reading the piezo z-position at the point of contact between the tip and the sample. Using the software described in the section 3.2 (page 23), the topographic data were combined with those pertaining to the absolute

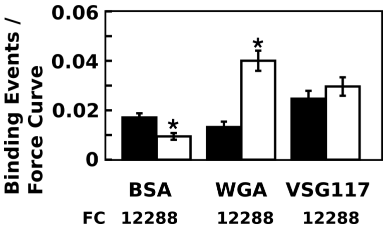
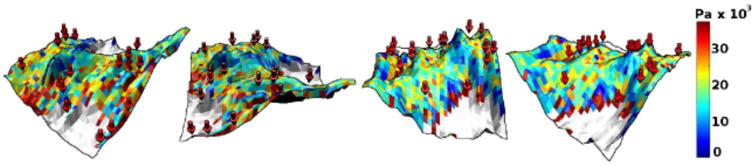
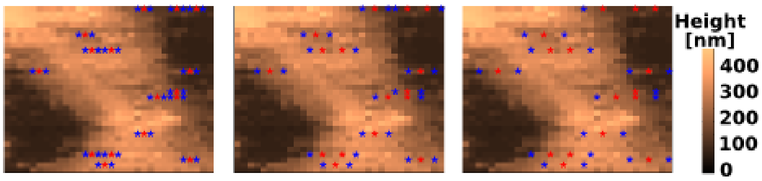


***Figure 4.4**: Control experiments done before (black) and after (white) timelapse experiment on neurons. The number of force curves (FC) analyzed is indicated. Data are means ± SEM. The stars indicates the p-values < 0.03 ; two-tailed t-test.*

stiffness (see below) and to the position of the binding-unbinding events to yield images such as those depicted in figure 4.5(a). The membrane Young's modulii are shown in "false" colors, which are mapped on the cell-surface topography in area units 2 µm × 2 µm with a resolution of 32 × 32 pixels. Pixels embracing a binding-unbinding event were considered as "GPI-domains" and are labeled with red arrows.



(a) 3D sequence

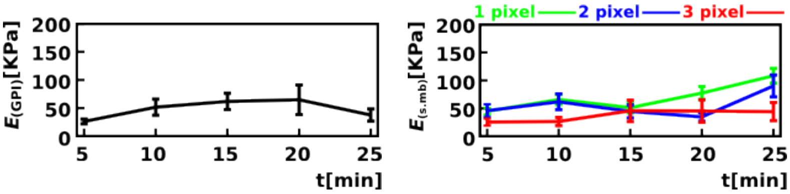


(b) Pixels selection

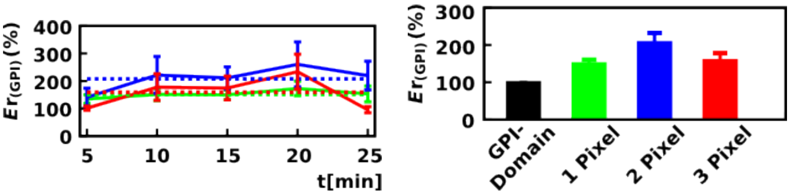
***Figure 4.5**: (a) Rotation series of 3D reconstructed images showing the stiffness (expressed in Pascal (Pa) according to a false colors scale) mapped on the topography. Red arrows indicate the location of GPI domains (specific events).*
(b) Topography ((a) top view, false colors) with red stars indicating GPI domains and blue stars indicating the immediately surrounding membrane at 1, 2 and 3 pixels distance afar.

Figure 4.5(b) illustrates the pixels that were taken into consideration for the computation. Red stars indicate the regions in which binding-unbinding events are detected, referred to as GPI domains. Blue stars represent the surrounding region at distances of 1, 2 and 3 pixels from the GPI domain.

In Figure 4.6(a) and (b), the mean stiffness values for the GPI domain (Figure 4.6(a)) and the 3 surrounding pixel regions are represented as a function of time. These data revealed no difference between the average stiffness of GPI domains (Figure 4.6(a)) and that of the surrounding membrane (Figure 4.6(b)). This finding reflects large-scale variations in cell stiffness, which result from lo-



(a) Global stiffness of events **(b)** Global stiffness around events



(c) Relative stiffness, time-lapse **(d)** Relative stiffness, histogram

Figure 4.6: ***(a)*** *Absolute and relative stiffness evolution of the GPI domains shown as a function of time (E(GPI)).* ***(b)*** *Absolute stiffness values of the surrounding membrane (E(s.mb)) at 1, 2 and 3 pixels (green, blue and red, respectively).* ***(c)*** *Relative stiffness of GPI domains ($Er_{(GPI)}$) in percent. For each binding - unbinding event, the $Er_{(GPI)}$ (red star in (B)) was calculated by dividing the E(GPI) by the mean of the E(s.mb) (blue stars in (B) for the corresponding GPI domain at 1, 2 and 3 pixels afar (green, blue and red, respectively). The mean of all the $Er_{(GPI)}$ was calculated and reported on the graph (12 to 30 values/time point). Data are means ± SEM. The means over the 25 min are indicated in dotted line and reported on the histogram ((d)) together with the SEM.*

cal differences in topography (membranes appear stiffer near the periphery than close to the center of the cell) and organellar content, and from the absence or presence of cytoskeletal filaments. To minimize these influences, we decided to consider only relative stiffness. This parameter was determined by dividing the absolute stiffness of each GPI domain (red star) by the stiffness of their respective surrounding membrane at distances of 1, 2 and 3 pixels (blue stars). That is to say that for instance, at one pixel distance afar, the absolute stiffness of the GPI domain (red star) was divided by the mean of the absolute stiffness of the two pixels along the x-axis immediately apposed (blue stars).

This type of measurement therefore only tells to which extent a GPI domain is stiffer or softer than the membrane located around it. Considering the relative stiffness of the GPI domain instead of its absolute value has the advantage that there is no need to follow a specific domain from scan to scan. The physical properties are extracted from the statistical behavior of the whole GPI domains of the scan frame and this ensemble can be used to determine the evolution of their properties as a function of time or as a function of chemicals, which could be added during the experiment. In addition, relative stiffness values can directly be compared and statistically processed independently of the location, of the cell or the moment when the measurement has been accomplished.

It is also well known that several cellular structures influence the mechanical properties of a cell (cytoskeleton, organelle etc..). By indenting a cell with an AFM tip and by fitting the resulting indentation curve with the Hertz model, one integrates the mechanical contribution of the different structures present under the cell membrane. The resulting value therefore reflects the global mechanical contribution of all the actors and hides the particular role of the cell membrane for example. In order to avoid interferences with cellular structures located deep into the cell, we only considered (i.e. we fitted with the Hertz model) the first 50 nm of the indentation curve after the point of contact. This approach has recently been validated on the components of the cytoskeleton by Kasas et al. [2005].

An additional difficulty in this type of measurements comes from the fact that membrane proteins and GPI domains are known to move in the plane of the cell membrane and that, unfortunately most of the currently available AFMs have a low scanning speed. As explained in section 3.2.4 page 26, we therefore compared only the stiffness of a GPI domain with the membrane located along the fast scan axis of the image (i.e. at the left and the right of the GPI domain in our images). This procedure reduces the number of reference measurements (pixels located around a GPI domain) but increases the temporal resolution. With this compromise, the time required to measure a GPI domain stiffness and its two surrounding references points drops to a reasonably short time scale permitting to assume that the GPI domain is immobile during the measurement.

The measurement of the mechanical properties of the GPI domains is expressed on a relative stiffness scale, a value below 100 indicates that the pixel of interest is softer than the surrounding region; a value above 100 indicated that it is stiffer. In Figure 4.6(c), the mean relative stiffness of all binding-unbinding events ($\mathrm{Er}_{(GPI)}$) is depicted as a function of time. The green, blue and red curves represent the stiffness of the GPI domains relative to that of the surrounding membrane at distances of 1, 2 and 3 pixels, respectively. This type of processing discloses only local changes in stiffness. The data reveal GPI domains to be stiffer than the surrounding membrane. The average relative stiffness values ($Er_{(GPI)}$) are depicted as column in Figure 4.6(d) using the same color codes. In this single-cell experiment, the GPI domains were on average 40% stiffer than the surrounding membrane at a distance of one pixel (Figure 4.6(d) : green bar). This value reflects the statistical properties of the entire population of GPI domains within the analyzed scan-frame, it does not correspond to the Er of a single GPI domain. Henceforth, all data are expressed as average values for several independent experiment (viz., several independent cells).

4.2.5 Relative stiffness of GPI domains

We applied the methodology explained in the previous section on living neurons with aerolysin functionalized tip in order to detect the GPI domains. The results indicated that the GPI domains are stiffer than their surrounding membrane.

As a control, the GPI domains relative stiffness, $Er_{(GPI)}$, was compared with the relative stiffness Er of randomly selected spots on the membrane ($Er_{(Rand)}$, Figure 4.7, (Random)). During analysis, GPI domains (i.e. spots with interactions) were excluded from the randomly selected data set. Here again, GPI domains were found significantly stiffer at one pixel afar (about 35% stiffer in average with N_{cells}=5, p-values < 0.03 , two-tailed ttest). It should be noticed that this value is consistent with the data shown in Figure 4.6, which concerned only a single cell.

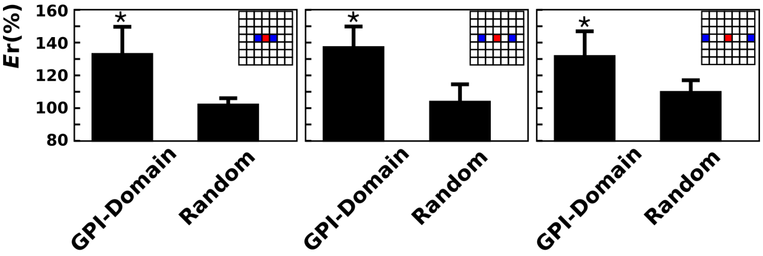


***Figure 4.7**: Relative stiffness of GPI domains when comparing at 1, 2 and 3 pixel distance. The corresponding random result is shown. Stars indicates p-values < 0.03 (two-tailed ttest) when comparing GPI domains with their respective random relative stiffnesses, N_{cells}=5.*

4.2.6 Cholesterol extraction effect on GPI domains relative stiffness

GPI domains are known to be enriched in cholesterol, the extraction of which from the cell membrane induces their disruption. This event can be used to gauge the mechanical properties of the GPI domain in the AFM. We therefore applied the cholesterol-extracting drug, methyl-β-cyclodextrin (MeCD), to living neurons during time-lapse AFM measurements. MeCD has already been used to study lipid rafts in living hippocampal neurons [Ledesma et al., 1998, 1999]. Using this model, 40-56% of the cholesterol was extracted using MeCD concentration in the range 1 - 10 mM [Shogomori and Futerman, 2001; Sooksawate and Simmonds, 2001]. In preliminary experiments, observation in the

AFM revealed some shrinkage of neurites using an MeCD concentration of 5 mM. At 2.5 mM, no such effect was observed. Hence, we opted for the lower dose.

The $\mathrm{Er}_{(GPI)}$ was monitored for 30 minutes after injecting medium alone (K5) and for 50 minutes after injecting MeCD into the cell chamber (Figure 4.8(a), GPI domains). These graphs were obtained by averaging five independent experiments (i.e. five different cells). Intact GPI domains were stiffer than the surrounding membrane at a distance of one pixel (N_{cells}=5, $p < 0.03$, two-tailed ttest). The injection of incubation medium (-30 min) had no influence on these relative stiffness values, no statistical difference was found when comparing $\mathrm{Er}_{(GPI)}$ values before and after injecting the K5 medium over 30 minutes recording (N_{cells}=5, $p > 0.05$, two-tailed t-test). However, the injection of MeCD has a proximate effect on the relative stiffness of the GPI domains. The disrupted GPI domain relative stiffness drops to 100%, meaning that these domains becomes as stiff as their surrounding membranes. The difference between the two population (GPI domains / Disrupted GPI domains) is significatively relevant (N_{cells}=5, $p < 0.03$, two-tailed t-test). The MeCD injection had no effect on the relative stiffness of randomly selected pixels, meaning that the observed change was specific to GPI domains.

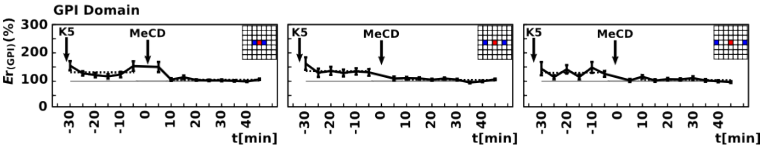
As an additional control, the comparison between the $\mathrm{Er}_{(GPI)}$ after the cholesterol extraction with the $\mathrm{Er}_{(rand)}$ shows no more significant differences between these two population (N_{cells}=5, $p > 0.05$, two-tailed t-test).

4.2.7 Cholesterol extraction effect on the number of detected GPI domains

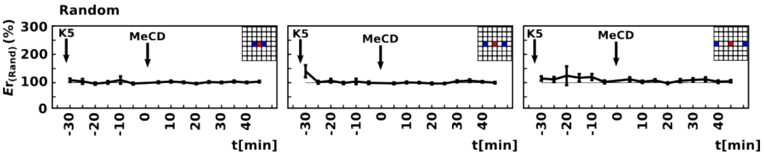
An additional possible effect of the membrane cholesterol extraction is the modification of the number of GPI-AP present on the cell membrane. One can claim that due to the disruption of rafts, the previously confined proteins can freely diffuse on the cell membrane, resulting in an increase of the detected GPI-aerolysin unbinding events. Therefore, we analyzed the number of unbinding events detected during the timelapse experiment. As depicted in figure 4.9(a) no such change was observed in the number of detected unbinding events (N_{cells}=5, $p > 0.05$, two-tailed t-test). Because several unbinding events can be present on a single curves, we also examined the number of event that are present per positive curves. Here also, no changes was observed when comparing before versus after the injection of MeCD [N_{cells}=5, $p > 0.05$, two-tailed t-test, figure 4.9(b)].

4.2.8 Cytoskeleton digestion effect on the relative stiffness of GPI domains

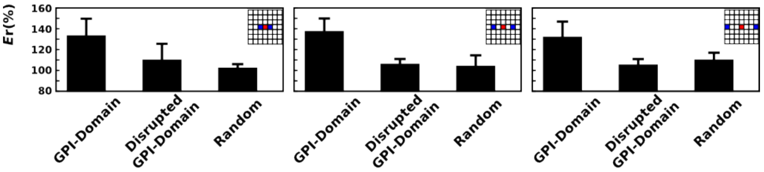
The cellular membrane is underlain by a cortical network of actin, which is separated from the inner leaflet by a distance of a few nanometers [Morone et al., 2006]. High-speed video microscopy has revealed lipids and proteins to diffuse within 100 – to 200 nm – diameter domains and to move form one domain to another. These domains were proposed to be delineated by an actin fence [Kusumi et al., 2004]. The spatial and temporal resolution of our instrument did not permit us to confirm or refute this proposal. Nevertheless, we wished to assess the influence of the actin cytoskeleton on measurements of $\mathrm{Er}_{(GPI)}$. To this end, we



(a) GPI



(b) Random

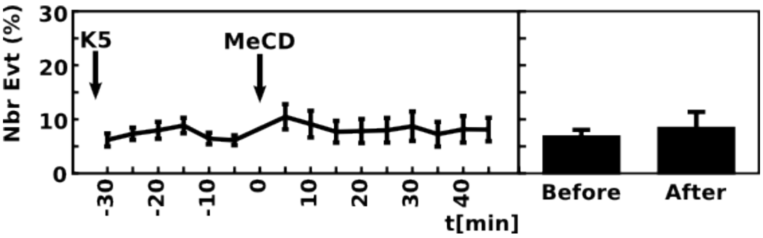


(c) Histogram

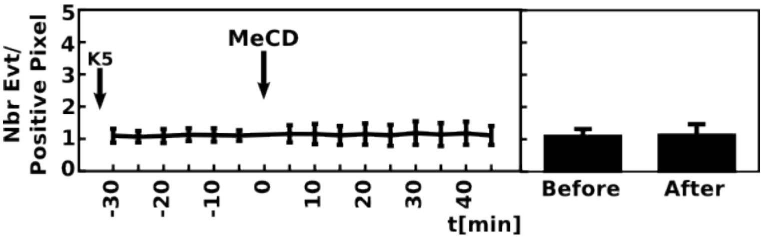
***Figure 4.8**: **(a)** $Er_{(GPI)}$ (GPI domain, inset: red squares) indicated in percent and plotted in function of time before and after 2.5 mM MeCD injection. $Er_{(GPI)}$ are shown at 1, 2, 3 pixels afar (inset: blue squares). The dotted line refers to the mean of $Er_{(GPI)}$ for the time periods [-30;-5] and [10;45] min.*

***(b)** $Er_{(Rand)}$ of randomly chosen pixels containing no detected GPI domains plotted as in **(a)**.*

***(c)** Histograms of the means of the $Er_{(GPI)}$ (GPI-domain) and of the $Er_{(GPI)}$ after MeCD treatment (Disrupted GPI domain) corresponding to the $Er_{(GPI)}$ over the [-30, -5] and [10-45] min periods shown in **(a)**, respectively, and of the $Er_{(Rand)}$ (Random) shown in **(b)**. 5 independent experiments were analyzed. Error bars indicate SEM.*



(a) Number events



(b) Number events per positive curves

Figure 4.9: *(a) Mean of the total number of binding – unbinding events recorded with aerolysin coated tips in function of time. The means of total number of binding events before and after MeCD injection are reported on the histogram.* $p>0.03$ *(b) Mean of the number of binding events per positive pixel (i.e. in which binding-unbinding events occurred) in function of time. The means of the number of binding-unbinding events per positive pixel before and after MeCD injection are reported on the histogram. Notice that for most of pixels in which binding-unbinding events were detected only one specific event occurred.* $p>0.03$

treated hippocampal neurons with cytochalasin B (5 µM), which depolymerizes the actin cytoskeleton. Treatment with cytochalasin B had no influence either on the number of binding-unbinding events detected or on the number of events per positive curve [$p > 0.03$; $N_{cells} = 5$, figure 4.11(a)]. Remarkably, no difference in $Er_{(GPI)}$ was observed before and after actin depolymerization [$p > 0.03$ at a distance of 1, 2 and 3 pixels; $N_{cells} = 5$ (Figure 4.10(a))]. Even 30 - 90 minutes after treatment with cytochalasin B, the Er value was higher in GPI domains than in randomly-selected control regions of the membrane [$p < 0.03$ at a distance of 1, 2 and 3 pixels; $N_{cells} = 5$ (Figure 4.10(a), (b))]. These data indicate that the differences observed on GPI-APs spots were not exclusively the result of changes in the cytoskeleton but that our measurements also reflect the mechanical properties of the membrane itself. As it can be noticed in Figure 4.10(c), after treatment with cytochalasin B, the difference between $Er_{(GPI)}$ and $Er_{(Rand)}$ was less at a distance of one pixel than at 2 or 3 pixels (see discussion).

To verify the effect of the cytochalasin on the absolute stiffness of the cell, the Young's modulus of the cell compartment located within the scan-frame was calculated. A statistically significant difference in this parameter was revealed with cytochalasin ($p < 0.03$; $N_{cells} = 5$), thereby demonstrating the actin cytoskeleton disrupting effect of the drug (Figure 4.12).

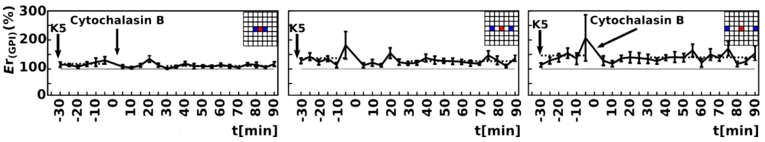
This result shows that fitting only the very beginning of the indentation curve with Hertz model is not enough to get rid of the actin cytoskeleton contribution. However, considering the relative stiffness instead of the absolute one, permit to detect differences in the membrane stiffness only.

4.2.9 Binding protein events influence on the relative stiffness

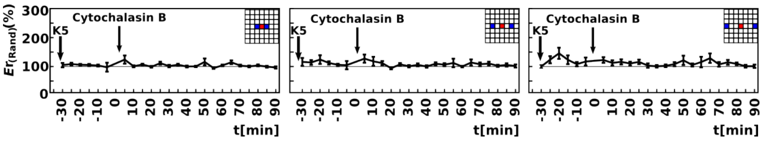
It can be speculated that measurements of cell membrane stiffness are influenced by the interaction between proteins present on the tip and those located within the cell membrane. To refute the existence of this phenomenon, we coated the tip with an antibody against the TfR, which is documented to reside in non-GPI-domains [Simons and Ikonen, 1997]. These experiments revealed no differences between the Er values for transferrin receptor-containing domains and randomly-selected control ones ($p > 0.05$, $N_{cells} = 5$, Figure 4.2.9). As a further control, we conducted measurements on neurons using tips functionalized with WGA. This protein binds sialic-acid residues, which are homogeneously distributed on cell membranes. The Er values obtained for WGA-mediated binding-unbinding events were similar to those registered in randomly-selected control regions at distances of 1, 2 and 3 pixels ($p > 0.05$, two-tailed ttest, $N_{cells} = 5$, Figure 4.2.9). Hence, the greater stiffness of GPI domains compared to surrounding regions of the membrane is apparently specific for this class of protein.

4.2.10 Cell type influence on GPI domains relative stiffness

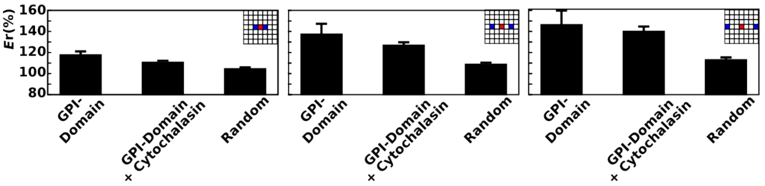
To exclude a peculiar behavior of hippocampal neurons, we repeated the same experiment with 293T cells line derived from primary cultures of human embryonic kidney cells. Here again, we noticed that the GPI domains were stiffer and



(a) GPI Timelapse



(b) Random Timelapse



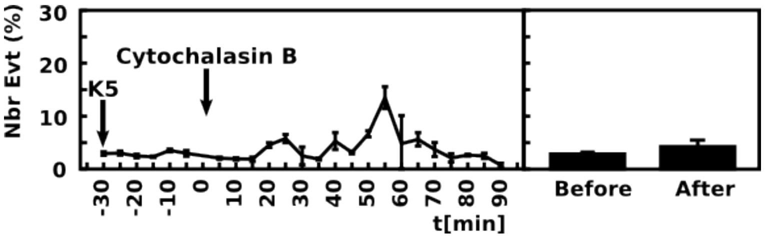
(c) Histogram

***Figure 4.10**: (a) Er(GPI) (GPI domain, inset: red squares) indicated in percent and plotted in function of time before and after cytochalasin B treatment (5 μM). Er(GPI) are shown at 1, 2, 3 pixels afar (inset: blue squares). The dotted line refers to the mean of Er(GPI) for the time period [-30;-5] and [30;90] min.*

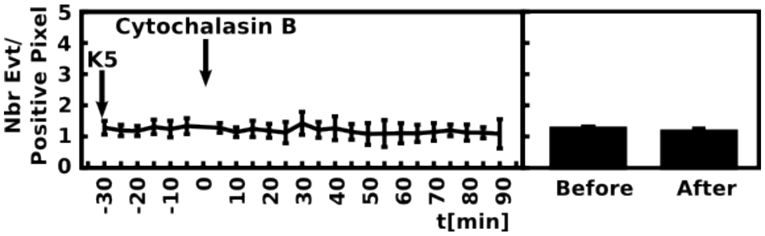
(b) Er(RanI) of randomly chosen pixels containing no detected GPI domains plotted as in (a).

(c) Histograms of the means of the Er(GPI) before (GPI-Domain) and after cytochalasin B treatment (GPI-Domain + Cytochalasin) corresponding to the Er(GPI) over the [-30, -5] and [30-90] min, respectively shown in (a), and of the Er(Rand) shown in (b) (Random).

5 independent experiments were analyzed. Error bars indicate SEM.



(a) Number events



(b) Number events per positive curves

Figure 4.11: *Description*

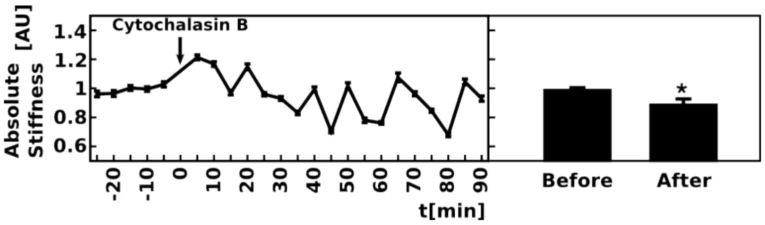


Figure 4.12: *Absolute Young's modulus (Absolute Stiffness) of neurites treated with cytochalasin B. The average of the Young's modulus for the pixels covering the entire neurite in the scan frame was calculated at each time point and then plotted in function of time. Between 4,636 and 5,086 force curves were analyzed per time point with a total of 34,492 and 63,245 force curves analyzed for the time periods prior and after Cytochalasin B treatment, respectively. Time points for injection of K5 and MeCD are indicated by arrows on the plot. The means of the absolute stiffness (in arbitrary unit) before and after MeCD injection are reported on the histogram. 5 independent experiments were analyzed. Error bars indicate SEM.*

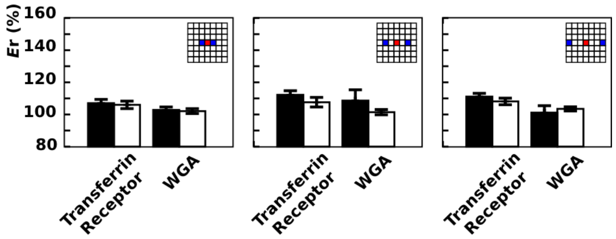


Figure 4.13: *Histograms indicate the mean of the Er calculated with anti-Transferin-coated tips and WGA-coated tips. Means of the Er for the corresponding random controls are indicated in white filled boxes. The three histograms represent the results obtained when analyzing membranes located at 1, 2 and 3 pixels afar (see insets). 5 independent experiments were analyzed. Error bars indicate SEM.*

that a similar stiffness drop occurred after cholesterol extraction (Figure 4.14). The use of aerolysin offered the advantage of analyzing GPI-APs without discriminating any specific protein. However, to test whether similar results could indeed be obtained with a specific protein, we overexpressed a GPI-GFP construct in 293T cells and functionalized the tips with an anti-Green Fluorescent Protein (GFP) antibody. In this case, the GPI-GFP domain had no significant difference in stiffness with the surrounding membrane at one pixel afar, which could be explained by the oligomerization of the construct. However, GPI-GFP membrane domain appeared clearly stiffer than the surrounding membrane at two and three pixels afar (Figure 4.14). After treatment with MeCD, the stiffness of the GPI domains decreased to the same value as that recorded for the rest of the membrane (Figure 4.8(a)), and of 293T cells (Figure 4.14).

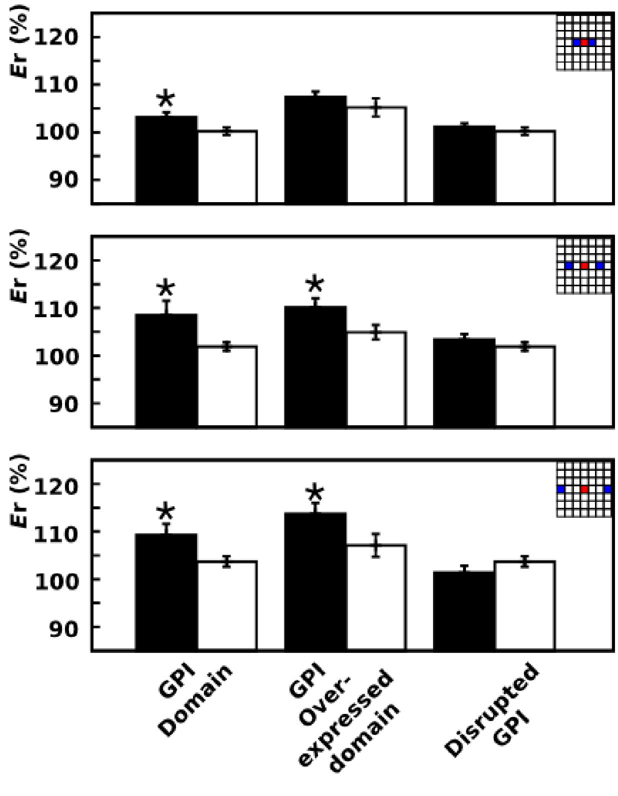


*Figure 4.14: Non-confluent cultures of 293T cells were probed with aerolysin functionalized tips using a similar protocol to that described for neurons. The cells were analyzed 15 minutes after injecting MeCD into the cell chamber. For experiments with these transfected 293T cells (n = 5-10), the AFM tips were coated with a rabbit polyclonal antibody against GFP (Molecular Probes) at a concentration of 2 μg/ml. The Figure shows histograms of the relative stiffness values recorded during a time-lapse experiments. Values for the GPI domains were collected between -30 minutes and -5 minutes (before treatment with MeCD). Values for disrupted GPI domains were collected between 5 minutes and 40 minutes (after injection of MeCD). Mean values (± SEM) are represented for 3 (293T cells expressing the GPI-GFP chimera) and 4 (293T cells ± MeCD treatment) independent experiments. Statistical significance was evaluated using the two-tailed t-test. * : $p < 0.05$*

4.2.11 Protein diffusion influence on GPI-domains relative stiffness

It is very important to note that there are two time scales involved in the experiments. One is the time (5 min) to take a full 2 x 2 microns size AFM image. The other time scales (140 ms), which is the most relevant in our experiment, relates to the time required to take one force curve on one pixel of the image. Previous studies have shown that clustered GPI-APs diffuse at $6*10^{-4}$ µm^2/s, whereas non clustered GPI-APs diffuse more rapidly i.e. at 3.9 µm^2/s [Nohe et al., 2006]. A relative stiffness measurement at one pixel distance requires 420 ms (700 ms at two and 980 ms at three pixels afar). During this time, a clustered GPI-APs can diffuse over a surface of $2.52*10^{-4}$ µm^2. This surface correspond to a circle of 0.0179 µm of diameter, whereas the AFM tip displacement is 0.1875 µm. The diffusion rate of the clustered GPI-APs is therefore one order of magnitude slower than the AFM tip to which the GPI-APs look as immobilized structures. To get rid of the protein diffusion, we repeated the experiments on beforehand fixed HeLa cells (paraformaldehyde for 15 minutes at 4°C) that overexpressed GPI and with aerolysin coated tip. We obtained very similar results as for the overexpressed 293T cells. GPI domains appears stiffer than the surrounding membrane, whereas treatments with MeCD before cell fixation decrease the relative stiffness to the same value as the surrounding membrane (Figure 4.15).

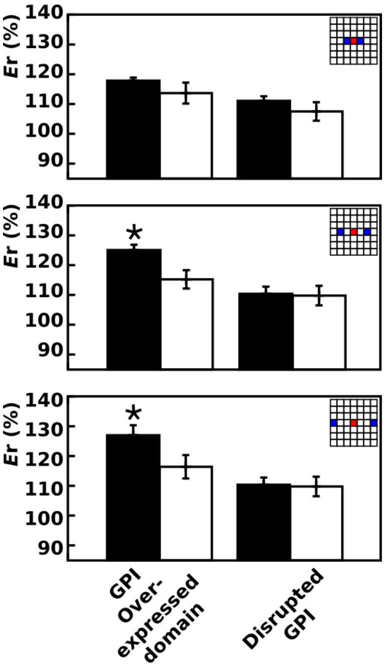


*Figure 4.15: The figure shows histograms of relative stiffness values recorded on fixed cells. For disrupted GPI, cells were treated with MeCD before fixation. Mean values (± SEM) are represented for 5 independent cells. Statistical significance was evaluated using the two-tailed t-test. * : $p < 0.05$*

4.2.12 Cholesterol extraction effect on membrane global stiffness heterogeneity

An additional method was used to assess the cell membrane mechanical properties. The global stiffness heterogeneity of the cell can be calculated without taking care on the position where binding-unbinding events occurred. During this analysis, the relative stiffness of all the pixels composing the force volume image are computed. Eventually, a threshold is applied to conserve only the stiffer zones. In our case, we defined a relative stiffness of 120 (20% stiffer) and more as stiffer zones, whereas others are ignored. On these images, groups of two and more pixels were noticed to form clusters.

The figure 4.16 depicts the results of such an analysis. At one pixel distance, before cholesterol extraction, 4.1% ± 0.4 of the stiffer zones were forming clusters. The cholesterol extraction reduced the number of these clusters to 1.8% ± 0.6 ($p<0.03$, two-tailed t-test). The relative stiffness at two and three pixel distance, showed similar differences between native and cholesterol extracted cells. Before cholesterol extraction, the proportion of clusters are respectively 43.3% ± 1.6 and 36.6% ± 1.9. These population significantly decreased to 34.0% ± 1.1 and 31% ± 1.3 respectively ($p<0.03$, two-tailed t-test) after cholesterol extraction.

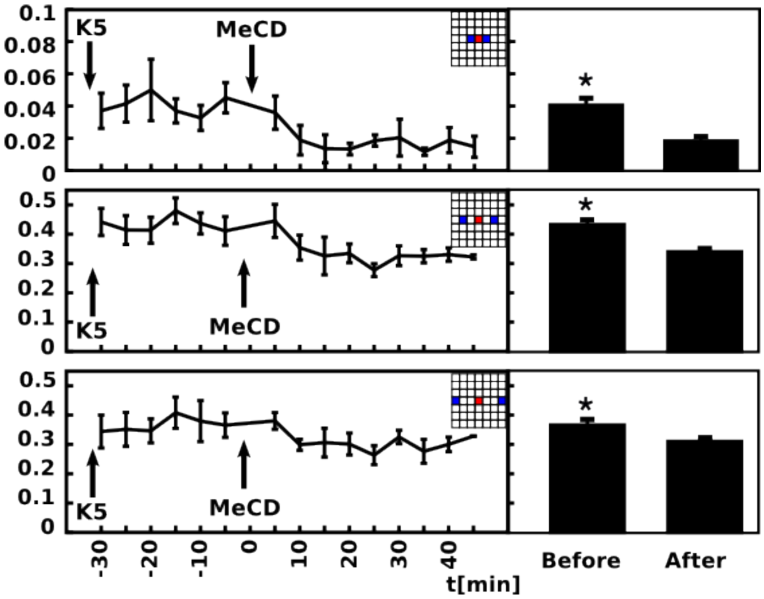


Figure 4.16: Cholesterol extraction effect on stiffness heterogeneity of the cell. Three relative stiffness are reported (top : relative stiffness at one pixel distance, middle : at two, bottom : at three pixel distance). The left part describes the time lapse. The right parts is a resume of the time lapse. "Before" is the mean values between medium buffer (K5) and MeCD injection, "After" is the mean value after the MeCD injection.

4.3 Discussion

In this chapter we present a methodology we developed to assess the cell membrane local mechanical properties. The method has been successfully applied to living neurons and permitted to estimate the average GPI domains size as well as their average mechanical properties.

4.3.1 GPI-anchored proteins are specifically detected by the aerolysin coated tips

Aerolysin is known to bind to the GPI anchor of proteins. Therefore aerolysin coated tips permit us to target a very large spectrum of proteins while being specific to raft-associated proteins.

The capability of our AFM setup to specifically detect the GPI-domains and hence the lipid raft microdomain was first assessed by control experiments.

The binding specificity of aerolysin coated tip was assessed by the use of tip coated with M41C, an aerolysin mutant impaired in GPI anchor affinity. The binding efficiency, 8% for aerolysin coated tips and 1% for M41C coated tips, indicates the specificity of aerolysin to the GPI anchor of the proteins.

We compared the GPI-anchor distribution on neurons obtained with the fluorescence microscopy with the one obtained with our AFM setup. In both cases the GPI-APs were shown to be present on the whole neurons. In addition, the AFM experiments put in evidence a significant difference in the GPI-anchor density between the soma, the axon and the dendrites. This is in accordance with the hypothesis of rafts acting as signaling platforms. The axons and dendrites, specialized on signal transmission, display a high concentration of rafts under the AFM, whereas the soma displays a lower rafts concentration.

Finally the tip contamination was tested. The binding affinity of the aerolysin coated tip was compared before and after timelapse experiments. The affinity with VSG117, a GPI-AP, is not affected by the use of the functionalized tip. Nevertheless, the increase in the number of binding event with WGA points a contamination of the tip with sugar moiety. However the decrease in the number of binding events with BSA indicates that this contamination does not affect the binding specificity.

4.3.2 GPI domains are local stiffer zones

The measurement of the mechanical properties of the GPI domains was the major challenge of the project. The two major difficulties we faced were the specific measurement of the cell membrane and the large stiffness variations occurring over the surface of living cells. The first issue was addressed by considering only the very first 50 nm of the indentation curve. It permitted to restrict the measurement depth to the surface of the cell. Concerning the second issue it is well known that the cell stiffness changes dramatically all over the cellular surface and that, thermal drift of the instrument, creep, or motions of the cell dramatically compromises any attempt to measuring subtle stiffness variations. We therefore did not measure the absolute stiffness of a single GPI domain rather we compared its stiffness with the stiffness of the spots located in its immediate vicinity and which were not identified as GPI domains (i.e. with no interactions). The use of the relative stiffness allowed thus to get rid of several problems among which those previously mentioned ones and some others such as the topology of the cell, the height of the cytoplasm above the substrate, the large scale local stiffness variations or the requirement to follow the same domain after successive scans. By considering the relative stiffness, it became possible to compare and to average the different values and data we present herein, therefore, reflects the statistical behavior of thousands of GPI domains. For these measurements, we made the assumption that there was no correlation between adjacent pixels. This is based on the notion that the small size of the tip (nominal tip radius of curvature = 20 nm) should not introduce pixel correlation beyond one pixel distance (62.5 nm). Pixels correlation should have led to reproducible results independently of the treatment or of the tip coating.

The time resolution of the method is another important issue, which influence has been minimized by considering only the neighboring pixels located along the microscope's fast scanning axis. Thus a measurement at one pixel distance required only 420 ms per binding-unbinding events. Since GPI-APs were documented to move much slower.

The relative stiffness measurements revealed thus that that GPI domains are 30% stiffer than the rest of the membrane. It should be pointed out that the method used herein does not exclude that the GPI domains could even be stiffer than the value we actually measured. It is due to a phenomenon, which can be illustrated by a coin glued at the surface of an inflated balloon. The stiffness one measures by pushing on the coin looks higher that the rest of the balloon but this apparent stiffness is far lower than that of the coin itself.

The measurements also permitted also to estimate the size of the GPI domains according to their stiffness difference with the surrounding membrane. The domains were found inferior to 70 nm in agreement with the reported size for these domains using other approaches [Kusumi et al., 2004; Simons and Ikonen, 1997; Anderson and Jacobson, 2002; Sharma et al., 2004] and AFM on artificial membranes [Tokumasu et al., 2003].

4.3.3 GPI domains relative stiffness is not dependent on protein-protein interactions

The protein-protein influence on the relative stiffness was assessed by several control experiments. Firstly, we disrupted the GPI-domains by extracting the cholesterol from the membrane. Secondly, we coated the tip with an antibody against the TfR, a non-raft marker. Thirdly, we coated the tip with WGA that binds to sugar moiety unspecifically distributed on the cell membrane.

The GPI domains disruption was performed by the use of MeCD, a chemical compound known to extract the cholesterol out of the membrane but which leaves the GPI-APs unaffected. The experimental results clearly show that without cholesterol, GPI domains recover the same stiffness as the rest of the membrane, whereas the number of interactions remained unchanged, giving an additional credit to our method. Our results are also in agreement with earlier studies on the role of cholesterol on artificial membranes stiffness [Sharma et al., 2004].

As the second control we repeated the experiments with AFM tips coated with anti-transferrin receptor antibodies. TfR has been chosen because it is known to be excluded from the GPI domains. The relative stiffness of the TfR-anti-TfR antibody interaction spots was the same as the one obtained on the randomly selected spots.

The last control was performed using WGA coated tip. The lectin binds to sugar moiety distributed on the membrane. This unspecific labelling shows again a relative stiffness identical to the one computed on randomly selected pixels.

These experiments therefore demonstrate that the stiffness increase we previously measured on GPI domains is specific to the domains and does not reflect any putative increase in the stiffness, which would be induced by the creation of a link between the tip and the cell membrane.

4.3.4 GPI domains relative stiffness does not depend on actin cytoskeleton

The next point to elucidate, was whether the GPI domains stiffness, which we measured was not a consequence of the actin cytoskeleton filaments that are present underneath the membrane. We therefore exposed the cells to cytochalasin, a chemical known to affect actin filaments. Here again, the GPI domains relative stiffness was higher than the rest of the membrane whether the actin cytoskeleton was disrupted or not. The cytochalasin action was confirmed by the fact that the absolute stiffness of the all scan frame dropped significantly after injection, in agreement with Rotsch and Radmacher [2000]. This last point demonstrates that considering the first 50 nm of the indentation curve permits not only to measure the stiffness of the cell membrane (by relative measurements) but also permits to apprehend the actin cytoskeleton (by absolute measurements). It should be noticed that the absolute stiffness of the overall scanned area did not significantly vary after cyclodextrin injection, the treatment which extracts cholesterol but has no action on the actin filaments.

4.3.5 GPI domains relative stiffness has similar properties in several cell lines

The validity of our measurement on other cell type was assessed. The results we obtained on 293T and HeLa cells give similar results to the presented experiments on neurons.

4.3.6 Protein diffusion does not alter the GPI domains relative stiffness measurement

The diffusion rate of proteins in the membrane is very fast, but dramatically slows down when inside rafts [Nohe et al., 2006] to diffuse slower than the velocity of our microscope. To verify the contribution of the protein diffusion on our results, we repeated the experiments on fixed HeLa cells where all protein diffusion is definitely absent. Results were very similar than in the living cells. The stiffness differences were however less important than on living HeLa cells, probably due to the fixation procedure, where all proteins are crosslinked.

4.3.7 Cell stiffness global heterogeneity is altered by cholesterol extraction

The cell stiffness global heterogeneity was introduced to get rid of the GPI-domains detection. The results shows that, looking at one pixel distance, 4% of the stiffer domains are contained into clusters. Cholesterol extraction reduced this population down to 1.8%. Similarly, when looking at two and three pixels distances, the population of clusters dropped from 43% and 37% down to 34% and 31% respectively. These results suggest an homogenization of the cell membrane mechanical properties and reinforce the results obtained on GPI domains.

4.4 Conclusion

One main goal of this study was to determine whether AFM could detect difference in stiffness of membrane domains in living cells. This is an important point in the context of the debated issue of the behavior of proteins and lipids in raft microdomains. This approach allows to address the issue of the relationships between membrane mechanical property, as stiffness, membrane structure and protein and lipid diffusion. It is clear that raft-associated protein-protein interaction may participate in driving molecular assemblies as suggested by the work of Douglass and Vale [2005] though participation of the lipid environment is not excluded.

We demonstrated that the GPI domains are stiffer than the surrounding membrane and many reports have documented that the diffusion of lipids and proteins are slower in so-called raft microdomains on living cells [Kusumi et al., 2004; Mayor and Rao, 2004] and artificial membranes [Simons and Vaz, 2004]. It is thus attractive to propose that the stiff areas we observed correspond to specific platforms into which the diffusion of these lipids and proteins is modified allowing formation of signalling complexes involved in a variety of raft-dependent physiological events and infectious diseases [Lafont et al., 2004; Simons and Vaz, 2004]. A prediction of this hypothesis is that lowering changes in stiffness between domains should impair signalling. Indeed, several reports have shown that upon MeCD activation of signalling cascades are inhibited [Simons and Toomre, 2000]. More studies are obviously needed in order to understand the physics of the relationship between diffusion and stiffness. However, our data strongly suggest that difference in stiffness clearly occurs in biological membranes and that one class of proteins often considered as raft-associated proteins, i.e. GPI-anchored proteins, partition into stiff domains.

Altogether these results demonstrate that it is possible to measure the relative stiffness of plasma membrane domains in living cells. The method described herein provides thus perspectives for investigating biophysical properties of the cell plasma membrane with a broad spectrum of applications. It permits to study, at the nanometrical level, the role of adhesion mechanisms on the membrane of living cells. Further applications could include the study of mechanical properties of receptors before and after stimulation with physiological ligands or pharmaceutical compounds. Hence, a link can be established between mechanical properties of the lipid bilayer and the fate of surface proteins leading to signalling activation shedding new lights on the study of membrane-coupled signal transduction.

Chapter 5

Stiffness Tomography

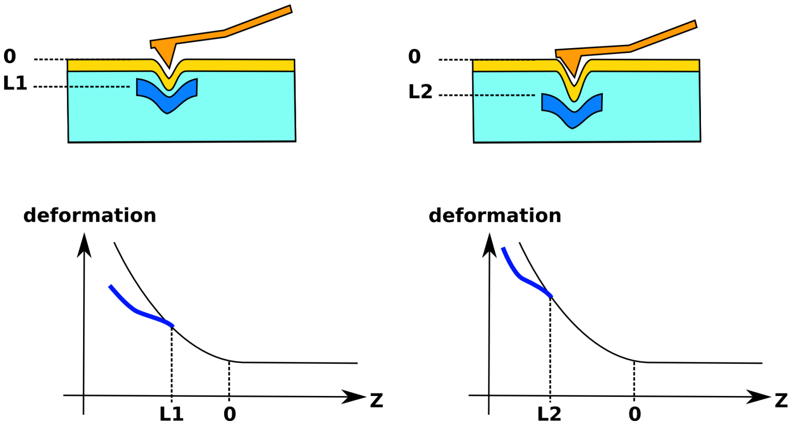
5.1 Introduction

In this chapter, we present a new atomic force microscopy imaging technique which permits to distinguish structures of different stiffness buried into the bulk of the sample. The technique is based on the analysis of the vertical deformation of the microscope's cantilever while the AFM tip indents the sample (section 5.1.1). The working principle of this new imaging technique has been verified by finite element models (section 5.2.1) and subsequent, by experiments applied to living cell (section 5.2.2).

5.1.1 The theory

This last chapter describes the Stiffness Tomography (ST), a new imaging mode we developed in our laboratory. Very soon after its invention, the AFM has also been used to measure the mechanical properties of the sample with a nanometric resolution [Tao et al., 1992]. These measurements are accomplished by pushing the AFM tip into the sample and by monitoring the cantilever deflection during the process, as described in page 5. The curve displaying the cantilever deformation as a function of the tip position is referred to as a FD curve. The shape of this curve permits to estimate the stiffness of the sample, assuming that the shape of the tip and the spring constant of the cantilever are well defined.

FD curves have been used to examine mechanical properties of bone tissue, cartilage platelets, synaptic vesicles and different types of living cells [Tao et al., 1992; Laney et al., 1997] (see Introduction, section 1.2). In all these latter studies cells were considered as homogeneous objects and the FD curves had to fit to the so called Hertz model which assumes a homogeneous isotropic infinite sample. Therefore, when using such a model, the mechanical changes occurring at different depths are averaged and cannot be related to specific regions of the sample. However, the FD curves do actually contain information about the stiffness of the sample at different depths: during the penetration into a non-homogeneous soft sample, the tip successively encounters different stiffness regions which modify the path of the FD curve as depicted onto figure 5.1(a) and 5.2(a) where a softer and stiffer inclusion is present inside a homogeneous sample. The location at which the path diverges from the "ideal homogeneous path" (thin black line on the figure) is a function the depth at which the tip encountered a different stiffness region during its penetration process. The subfigure 5.1(a) and 5.2(a) with inclusion at depth L1 compared with their respective subfigures (b) with inclusion at the depth L2 depicts the the different path modification according to the depth of the inclusion.



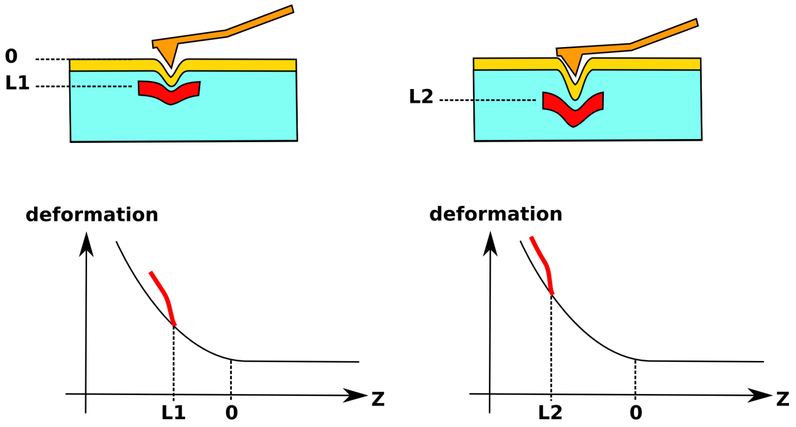
(a) Soft inclusion **(b)** Soft inclusion deeper

__Figure 5.1__: __(a)__ The presence of a soft inclusion in a homogeneous sample (top) and its influence on the recorded force curve (bottom, blue curve) compared to a homogeneous sample (bottom, black curve).
__(b)__ The presence of a soft inclusion deeper inside a homogeneous sample (top) and its influence on the recorded force curve (bottom, blue curve) compared to a homogeneous sample (bottom, black curve). Observe the difference due to the deepness of the inclusion between __(a)__ and __(b)__.

The validity of this concept has recently been demonstrated on living cells [Kasas et al., 2005]. In order to extract information about the stiffness of the different regions encountered during the indentation process, we computed the indentation curve, which informs on the force applied by the tip in function of its indentation into the sample (see section 3.2.2). Such a curve can be fragmented, in the indentation axes, in several segments of the same size. In the figure 5.3(a), we fragmented the indentation curve into several segments, beginning from the point of contact I_0. We can then define the segment ΔS_i, as the segment located between the indentation point I_i and I_{i+1}. The force needed to indent this portion of curve is then ΔF_i. For each segment ΔS_i, from the point of contact to the end of the indentation, a force ΔF_i can be associated.

The figure 5.3(b) describes the case of a hard inclusion into a homogeneous and isotrop material. The indentation curve deviates from the ideal one from a similar way than explained in figure 5.2. The green dotted line describes the indentation curve if the material were homogeneous. As soon as it feels the hard inclusion, the indentation curve, as the FD curve, deviates from the ideal curve, resulting in the plain red line. The application of the Hertz model on the segmented indentation curve results in the Young modulus of the indented sample (see equations 3.6 and 3.5 in section 3.2.2 page 3.2.2).

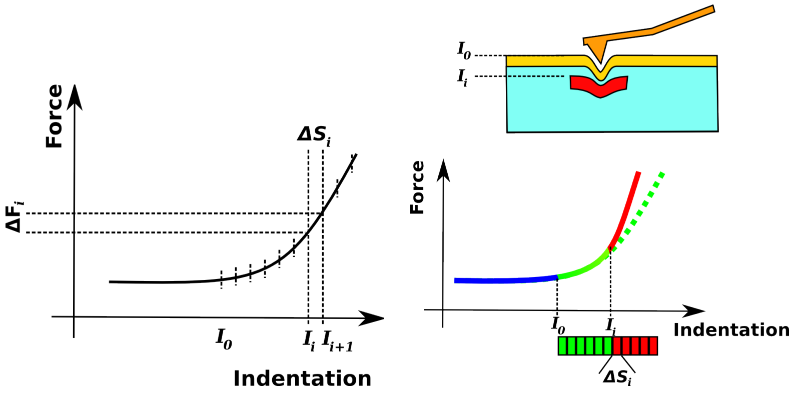
For each segment ΔS_i, we can compute its Young modulus ΔE_i. By displaying in false colors the calculated stiffness of each fragment at its corresponding depth, we obtain the “stiffness slice” of the sample, as depicted under the x axes



(a) Hard inclusion **(b)** Hard inclusion deeper

Figure 5.2: ***(a)*** *The presence of a hard inclusion in a homogeneous sample (top) and its influence on the recorded force curve (bottom, red curve) compared to a homogeneous sample (bottom, black curve).*

(b) *The presence of a hard inclusion deeper inside a homogeneous sample (top) and its influence on the recorded force curve (bottom, red curve) compared to a homogeneous sample (bottom, black curve). Observe the difference due to the deepness of the inclusion between* ***(a)*** *and* ***(b)****.*



(a) Curve segmentation **(b)** Stiffness segmentation

Figure 5.3: ***(a)****The indentation curve is divided in several segments of defined depth, ΔS from the point of contact I_0 to the end of the indentation curve. At the depth i, the force needed to indent into the segment ΔS_i is ΔF_i.*

(b) *The indentation curve into a homogeneous sample (green line) and indentation curve into a sample with hard inclusion (red line). The dotted line depicts the divergence from the homogeneous sample. The force, and then the Young modulus, can be computed for each segment ΔS_i. The stiffness is reported as a colorscale, soft (green) and hard (red), boxes above the indentation axes and represents the stiffness of each segments ΔS_i.*

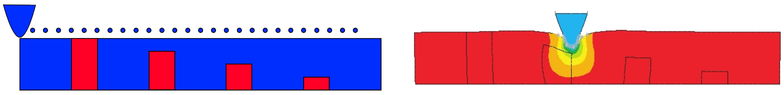
in the figure 5.3(b). The color green depicted zones with a lower Young modulus and red a higher Young modulus.

5.2 Results

5.2.1 Finite elements simulations

The description depicted in the previous section reflects an ideal case where a single inclusion is present inside a homogeneous sample. This theoretical simple case has to be validated. Since indentation in non-homogeneous samples is a highly non-linear phenomenon which can not be modelized by analytical means, we verified our method on a well defined finite elements model. We therefore simulated a virtual AFM and a virtual sample containing various "inclusions". This simulation has been carried on a commercially available finite elements program (ANSYS™ 9.0).

Several diferent models were simulated. One of those is shown in figure 5.4(a). It represents a homogeneous sample (blue box) with several inclusion column (red columns) starting at different depth. The modelized tip is shown to the left of the figure. The small circles above the sample indicate the spots where the indentations were simulated. The finite element software calculated the FD curves shape during the indentation of the tip. Figure 5.4(b) shows the tip and the deformed sample during the indentation process. The magnitude of the total displacement vector is depicted in false colors.



(a) Simulation model **(b)** Intermediate result

Figure 5.4: *Simulation of the indentation process by using the finite elements method. The sample contains inclusions (a) colored in blue which have a Young modulus tree times higher than the bulk of the sample colored in red. The AFM tip and the spots where indentation was simulated are also represented in blue. During the indentation process the sample deforms as depicted on (b). The deformation sum vector is displayed in false color, no displacement, red and high displacement, blue.*

Since the finite element simulation was done in 2D, a simplified Hertz model was elaborated, that describes a 2 dimensional tip which indents in a 2 dimensional sample (equation 5.1).

$$E = \frac{2}{\pi}\frac{F}{\delta}(1-\nu^2) \tag{5.1}$$

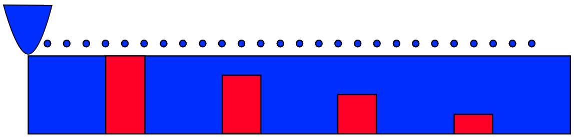
In this equation, ν is the Poisson ratio of the material, δ the indentation inside the material, E the Young Modulus of the sample, and F the force applied by the tip.

This 2 dimensional model was included to the software for processing the files generated by the finite element simulation.

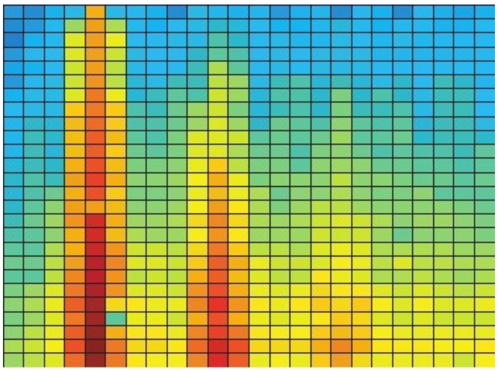
We tested the ST concept on different finite element models.

Basically, the samples consisted of a "homogeneous gel" which contained different type of inclusions. These inclusions had a shape which could be a "column", such as those depicted on figure 5.5(a), or a "plateau", such as those on figure 5.6(a) which were inserted at four different depth. The Young modulus of the inclusions was set higher or lower than the bulk of the sample (i.e. the gel). The different simulated models, as well as their corresponding ST, are depicted on figures 5.5 to 5.9.

The first simulation, shown on figure 5.5, concerned columns that are three times stiffer than the bulk of the sample. On the ST result, one can clearly see the two first columns that are located at the surface and near the surface of the sample. The third column was still visible, but close to the background noise. Compared to the other simulations, this one offered the most contrasted results in the ST images.



(a) Simulation model

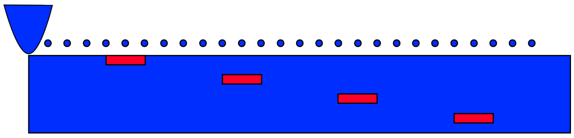


(b) Tomographic result

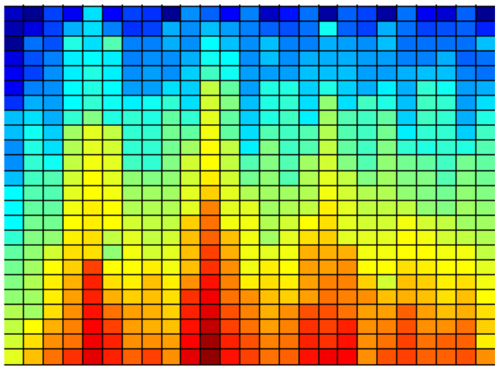
***Figure 5.5**: **(a)**Simulation model and **(b)** resulting tomographic view. The stiffness is indicated as color scale from soft (blue) to hard (red).*

The plateau configuration was set with different sizes and stiffnesses. The one depicted on figure 5.6 concerned thin plateaux that are three times stiffer than the gel. The case of bigger platforms that are three times stiffer than the gel is reported on figure 5.7, and finally, thin plateau that are ten times stiffer is described on figure 5.8. The result of these three simulations shows the ability of our method to detect plateau located under the surface. In all the results, the tomography revealed the presence of the plateau inside the sample, as deep as the third plateau. Nevertheless, our method did not premit to differenciate between a change of the size (figure 5.6 compared to figure 5.7) and a change of the stiffness (figure 5.6 compared to figure 5.8). As it could be expected, a stiff "shadow" appeared above the floating platforms.

The last simulation depicted on figure 5.9 concerned platforms that are three times softer than the surrounding material. The ST image reveals the ability of our technique to detect smoother materials into the sample, but also reveals the stiffness shadow underneath the plateaux as observed with stiffer platforms.

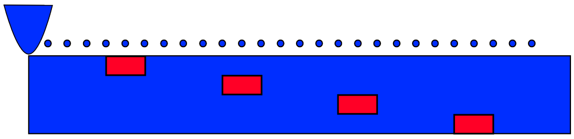


(a) Simulation model

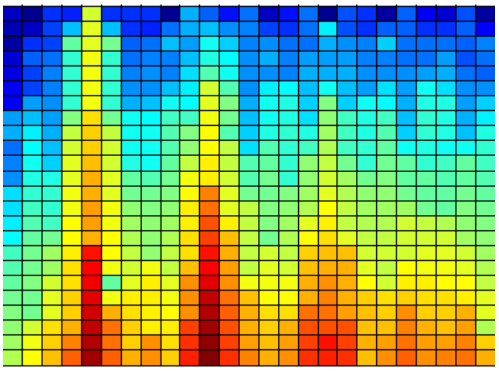


(b) Tomographic result

***Figure 5.6**: **(a)**Simulation model and **(b)** resulting tomographic view of thin platform included in different depth into a material. Platforms are three times stiffer than the surrounding material. The stiffness is indicated as color scale from soft (blue) to hard (red).*

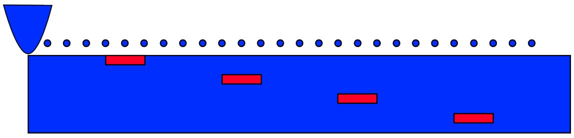


(a) Simulation model

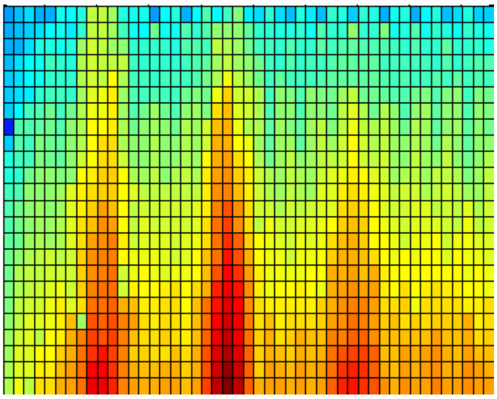


(b) Tomographic result

***Figure 5.7**: **(a)**Simulation model and **(b)** resulting tomographic view of thick platforms included in different depth into a material. Platforms are three times stiffer than the surrounding material. The stiffness is indicated as color scale from soft (blue) to hard (red).*

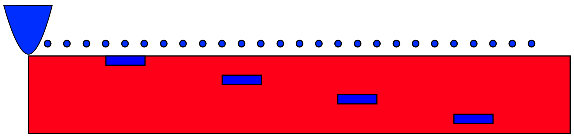


(a) Simulation model

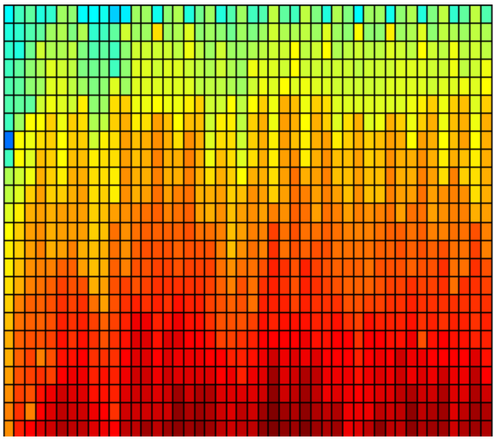


(b) Tomographic result

***Figure 5.8**: **(a)**Simulation model and **(b)** resulting tomographic view of thin platform included in different depth into a material. Platforms are ten times stiffer than the surrounding material. The stiffness is indicated as color scale from soft (blue) to hard (red).*



(a) Simulation model



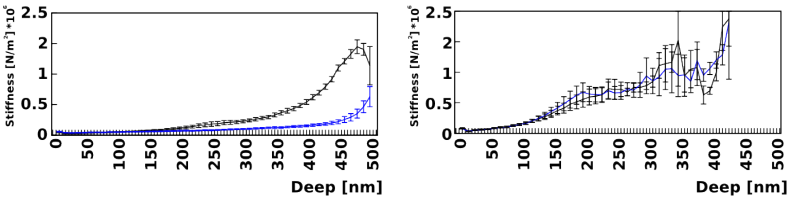
(b) Tomographic result

Figure 5.9: ***(a)****Simulation model and* ***(b)*** *resulting tomographic view of thin platform included in different depth into a material. Platforms are three times softer than the surrounding material. The stiffness is indicated as color scale from soft (blue) to hard (red).*

5.2.2 Application of the Stiffness Tomography (ST) to living systems

Effect of cytoskeleton digestion

In a next step, ST was applied on living systems. In order to test ST ability to detect a stiffness contrast and to monitor stiffness changes as a function of the depth, we compared the stiffness of 293T cells before and after the injection of cytochalasin B at 5 µM, a chemical which is known to depolymerize the actin cytoskeleton. To monitor the depolymerization process, we calculated the stiffness of each segment of the FD curve as a function of its depth under the membrane. The values of all the segments located at the same depth were finally averaged. Fig. 5.10(a) depicts the change of the average cell stiffness, as a function of the depth under the membrane, before and after the injection of 50 µM cytochalasin. The graph showing the stiffness as a function of the depth contains the average of the 4 force volume files which were recorded between 25 and 5 minutes before the injection of cytochalasine (five scans). Whereas the graph displaying the stiffness after the injection, contains the average of the 6 force volume files recorded between 30 and 55 minutes after the injection of cytochalasine. The two curves show the development of a very clear softening starting at a depth of about 150 nm under the membrane in the cell after the arrival of the actin depolymerizing agent.

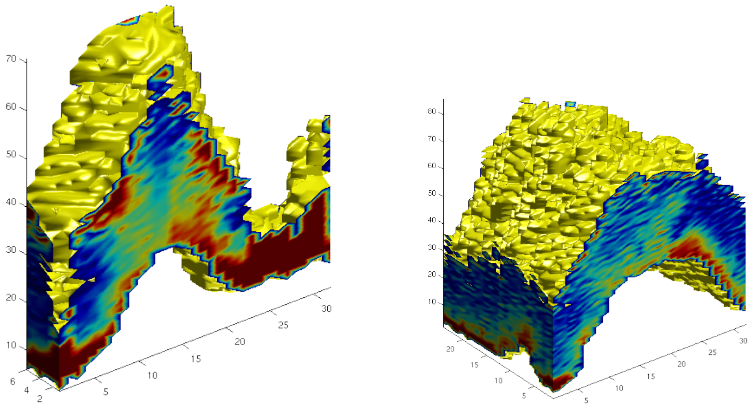


(a) Cytochalasin effect **(b)** Control

***Figure 5.10**: **5.10(a)** Mean stiffness in function of the depth of cell before (blue) and after (red) cytochalasin injection.*
***5.10(b)** Control experiment, same buffer medium injection*

Tomographic view of neurites

In a next step we applied the stiffness tomography imaging technique to living hipocampal neurons. During the data processing stage every FD curve was divided in segments of 10 nm. The AFM files were eventually processed with the same home made software which we used for the previously described finite element model. Fig. 5.11(a) and (b) depicts two “stiffness slices” recorded on two different living cells. As one can notice the contrast indicates the presence of hard structures buried into the cytoplasm of the cell. In (a), two stiff sheets are visible in the surface at both sides of the neurites. In the second tomography example (b), the inside of the cell appears more homogeneous. However, some structures are visible and contrast well from the background. These structures could correspond to the cortical actin cytoskeleton which is known to lie under the cellular membrane.



(a) Example 1 **(b)** Example 2

***Figure 5.11**: Stiffness tomography example of two neurites*

5.3 Conclusion

We developped a new imaging mode which we called “Stiffness Tomography”. With this mode, it becomes possible to image stiffness differences inside a soft sample. The validity of the concept has been successfully tested on virtual samples as well on living cells. We are aware that the mathematical model we are using for the moment is an oversimplification of a highly complex and non linear phenomenon. However, despite this simplification, as we could demonstrate here, with the finite elements simulations and the application on living cells, that stiffness tomography can highlight structures located underneath the surface of the sample, a domain up to now invisible to the AFM.

This method can of course be applied to other domains such as polymers. Since the method does not require any additional hardware it can easily be implemented on any AFM by adding an additional step to the data processing chain.

Appendix A

Blender script

Blender is an open source 3D modeling software. The scripting capabilities of this software permit us to automate some repetitive operations. The two following scripts loads files generated by the force volume analysis software to output three dimentional images and animations.

A.1 createTopoYoungEvent.py

This script generates the 3D shape of the object with topographical image in gray scale, and maps the Young's modulii with the presence of protein-protein interaction pointed with arrows.

```
#!BPY
## Script that import grey scale images and convert in into a 3D
    plane
## in which each verticle height correspond to the gray intensity
    of the
## original image
##
## (c) Charles Roduit, 2006
## release under the GNU GPL
## Blender 2.41

## Module importation
from array import array
import Image, Numeric, csv
import Blender
from Blender import NMesh, Scene, Object
## Needed to attribute an image as a texture
from Blender import Material, Texture
from Blender import Mathutils

#############################
# Path to the used files.
#
# To convert tif files into jpg, use the script
# ChangeImages.sh
filePath='/home/croduit/Neurones/FR21054B.014/'
Topofile='FR21054B.014_Topo.png'
Youngfile='FR21054B.014_Young.png'
EventFile='Event.csv'
pathSeparator='/'
```

```
#
####### Creation of the geometric objects #####
##
##
def circle(_diam,_nbPoints):
        _pasEntrePoints=2.*Numeric.pi/_nbPoints
        listOfPoints=[]
        for points in xrange(_nbPoints):
                posX=(_diam/2.)*Numeric.cos(_pasEntrePoints*points)
                posY=(_diam/2.)*Numeric.sin(_pasEntrePoints*points)
                listOfPoints.append([posX,posY,0])
        listOfFaces=[]
        for points in xrange(_nbPoints):
                if points<(_nbPoints-1):
                        listOfFaces.append([points,points+1,
                            _nbPoints])
                else:
                        listOfFaces.append([points,0,_nbPoints])
        circleMesh=NMesh.GetRaw()
        for composants in listOfPoints:
                Vertics=NMesh.Vert(composants[0],composants[1],
                    composants[2])
                circleMesh.verts.append(Vertics)
        #Central point creation
        circleMesh.verts.append(NMesh.Vert(0,0,0))
        for face_courante in listOfFaces:
                face=NMesh.Face()
                for numero_vertex in face_courante:
                        face.append(circleMesh.verts[numero_vertex
                            ])
                circleMesh.faces.append(face)
        return circleMesh
##
##
def cylindre(_diam,_haut,_nbFaces):
        cylindre=circle(_diam,_nbFaces)
        ceilCircle=circle(_diam,_nbFaces)
        for vertsNbr in xrange(len(ceilCircle.verts)):
                ceilCircle.verts[vertsNbr].co[2]=_haut
        for vertsNbr in xrange(len(ceilCircle.verts)):
                cylindre.verts.append(ceilCircle.verts[vertsNbr])
        for facesNbr in xrange(len(ceilCircle.faces)):
                cylindre.faces.append(ceilCircle.faces[facesNbr])
        ## The central point is at the position _NbrFace and 2*
            _NbrFace
        ## because we start from 0
        listOfNewFaces=[]
        for point in xrange(_nbFaces):
                if point < (_nbFaces-1):
                        listOfNewFaces.append([point,point+1,
                            _nbFaces+point+2,_nbFaces+point+1])
                else:
                        listOfNewFaces.append([0,point,_nbFaces+
                            point+1,_nbFaces+1])
        for face_courante in listOfNewFaces:
                face=NMesh.Face()
                for numero_vertex in face_courante:
                        face.append(cylindre.verts[numero_vertex])
                cylindre.faces.append(face)
        return cylindre
##
##
```

```
def cone(_diam,_haut,_nbFaces):

        _cone=circle(_diam,_nbFaces)
        ## Ascend the circle
        for vertNb in xrange(len(_cone.verts)):
                _cone.verts[vertNb].co[2]=_haut
        _cone.verts.append(NMesh.Vert(0,0,0))
        listOfFaces=[]
        for faces in xrange(_nbFaces):
                if faces<(_nbFaces-1):
                        listOfFaces.append([faces,faces+1,_nbFaces
                            +1])
                else:
                        listOfFaces.append([faces,0,_nbFaces+1])
        for face_courante in listOfFaces:
                face=NMesh.Face()
                for numero_vertex in face_courante:
                        face.append(_cone.verts[numero_vertex])
                _cone.faces.append(face)

        return _cone
##
##
def fleche(_diamP,_hautP,_diamB,_hautB,_nbFaces):
        ## _diamP, hautP = Tip diameter and height
        ## _diamB, hautB = Cylinder diameter and height (base)
        _fleche=cone(_diamP,_hautP,_nbFaces)
        ## Base creation
        baseFleche=cylindre(_diamB,_hautB,_nbFaces)
        for vertsNb in xrange(len(baseFleche.verts)):
                baseFleche.verts[vertsNb].co[2]=baseFleche.verts[
                    vertsNb].co[2]+_hautP
        ## Merge of the tip and the base
        for vertsNb in xrange(len(baseFleche.verts)):
                _fleche.verts.append(baseFleche.verts[vertsNb])
        for faceNb in xrange(len(baseFleche.faces)):
                _fleche.faces.append(baseFleche.faces[faceNb])
        return _fleche
##
####### Definition of the function that imports the datas
##
def OpenTopoImage(_file):

        ## _file is a string containing the file path and name to
            import
        OImage=Image.open(_file)
        (sizeX,sizeY)=OImage.size
        # Gray scale convertion.
        gsImage=OImage.convert('L')
        # Recovery of the grayscales
        # white=0, black=255
        height=[]
        for posY in xrange(sizeY):
                height.append([])
                for posX in xrange(sizeX):
                        # stricly speaking recovery...
                        print(str(posX)+':'+str(posY))
                        height[posY].append((gsImage.getpixel((posX
                            ,posY))/255.)*30)
        return height
##
##
```

```
def createTopo(_height,original=[]):
        sizeX=len(_height)
        sizeY=len(_height[0])
        ## definition of the list of the points that compose the
            plane
        listOfPoints=[]
        for posX in xrange(sizeX):
                for posY in xrange(sizeY):
                        listOfPoints.append([posX,posY,_height[posX
                            ][posY]])
        ## definition of the list of the faces that compose the
            plane
        listOfFaces=[]
        for y in xrange(sizeY-1):
                for x in xrange(sizeX-1):
                        listOfFaces.append([y+x*sizeY,y+x*sizeY+1,y
                            +(x+1)*sizeY+1,y+(x+1)*sizeY])
        ## Plane definition
        TopoMesh=NMesh.GetRaw()
        for composants in listOfPoints:
                Sommet=NMesh.Vert(composants[0]/10.,composants
                    [1]/10.,composants[2]/20.)
                TopoMesh.verts.append(Sommet)
        ## Faces definition
        for face_courante in listOfFaces:
                face=NMesh.Face()
                for numero_vertex in face_courante:
                        face.append(TopoMesh.verts[numero_vertex])
                TopoMesh.faces.append(face)
        return TopoMesh
##
##
def createMaterial(_TextureFile):

        _thisMaterial=Material.New('YoungMaterial')
        _thisTexture=Texture.New('YoungModulus')
        ## We specify the texture type
        _thisTexture.setType('Image') ## equvalent to YoungTexture.
            Type='Image'
        ## We load the image to map
        _thisImageToMap = Blender.Image.Load(_TextureFile)
        ## and we link the image to the texture
        _thisTexture.image=_thisImageToMap
        _thisTexture.setExtend('Extend')
        ## We link the texture to the material
        _thisMaterial.setTexture(0,_thisTexture,Texture.TexCo.ORCO,
            Texture.MapTo.COL)
        return _thisMaterial
##
##
def modifyTopo(_NewHeight,_OldTopo):
        _temp=NMesh.GetRaw('_OldTopo')
        sizeX=len(_NewHeight)
        sizeY=len(_NewHeight[0])
        listOfPoints=[]
        for x in xrange(sizeX):
                for y in xrange(sizeY):
                        listOfPoints.append([x,y,(y+x*sizeY)/100.])
        print 'number of vertics in input topo : '+str(len(_OldTopo
            .verts))
        print 'number of vertics deduced with height : '+str(len(
            listOfPoints))
        print 'The value of first old verticle height : '+str(
            _OldTopo.verts[0].co[2])
        print 'The value of first new verticle height : '+str(
            listOfPoints[0][2])
        for numberVerts in xrange(len(_OldTopo.verts)):
                _OldTopo.verts[numberVerts].co[2]=listOfPoints[
                    numberVerts][2]
        return _OldTopo
##
##
def OpenEventPosition(_file):
        csvEventGride=csv.reader(open(_file,'rb'))
        eventGride=[]
        for row in csvEventGride:
                eventGride.append(row)
        return eventGride

def getArrowForrest(_positionGride,_topoPlane):
        # return an object composed by arrows that takes position
            regarding
        # the values included in _positionGride. They are
            distributed in a
        # rectangle of size _sizeX, _sizeY.
        # The size _sizeX and sizeY are the extreme point
            coordinate.
        _sizeX=_topoPlane.verts[len(_topoPlane.verts)-1].co[0]
        _sizeY=_topoPlane.verts[len(_topoPlane.verts)-1].co[1]
        _nbrVerts=len(_topoPlane.verts)
        NbYVerts=0
        oneLine=0
        while not oneLine:
                NbYVerts+=1
                if _topoPlane.verts[NbYVerts].co[1]==0:
                        oneLine=1
        NbXVerts=_nbrVerts/NbYVerts
        GESizeX = len(_positionGride)
        GESizeY = len(_positionGride[0])
        ## multiplicator to position the cursor to the correct
            place
        MultX=_sizeX/GESizeX
        MultY=_sizeY/GESizeY
        #ArrowForrest=NMesh.GetRaw()
        listOfArrows=[]
        NumberOfArrows=0
        print(len(_positionGride))
        print(_positionGride)
        for posX in xrange(len(_positionGride)):
                for posY in xrange(len(_positionGride[0])):
                        if int(_positionGride[posX][posY]):
                                ## Convert the positions
                                _PosXInPlane=int((posX+0.5)*(
                                    NbXVerts/GESizeX))+1
                                _PosYInPlane=int((posY+0.5)*(
                                    NbYVerts/GESizeY))+1
                                NewZPosition=_topoPlane.verts[(
                                    NbYVerts*NbXVerts)-(
                                    _PosXInPlane+_PosYInPlane*
                                    NbYVerts)].co[2]
                                ##########
                                ## arrow creation...
                                _thisArrow=fleche
                                    (0.1,0.1,0.05,0.1,10)
                                ## Moving the arroy
                                for numberVerts in xrange(len(
                                    _thisArrow.verts)):
                                        _thisArrow.verts[
                                            numberVerts].co[0]\
                                        =_thisArrow.verts[
                                            numberVerts].co[0]+((
                                            posX+0.5)*MultX)
                                        _thisArrow.verts[
                                            numberVerts].co[1]\
                                        =_thisArrow.verts[
                                            numberVerts].co[1]+((
                                            posY+0.5)*MultY)
                                        ## and in Z
                                        _thisArrow.verts[
                                            numberVerts].co[2]\
                                        =_thisArrow.verts[
                                            numberVerts].co[2]+
                                            NewZPosition+0.2
                                listOfArrows.append(_thisArrow)
        return listOfArrows
        pass


##############################
# Topography Creation
#
image=OpenTopoImage(filePath+pathSeparator+Topofile)
plane=createTopo(image)

#
##############################
# Application of the Young modulus texture
#
PlaneMaterial=createMaterial(filePath+pathSeparator+Youngfile)
## and we link the material to the plane...
plane.setMaterials([PlaneMaterial])

#
##############################
# Place the arrows where the event are.

eventGride=OpenEventPosition(filePath+pathSeparator+EventFile)

arrowPosition=getArrowForrest(eventGride,plane)
NumberOfEvents=0

# Material creation for the arrows
ArrowMat=Material.New('Material')
ArrowMat.rgbCol = [1,0,0]
ArrowMat.setAlpha(1.0)

for arrows in arrowPosition:
        ## assignation of a material
        arrows.materials.append(ArrowMat)
        thisArrow=NMesh.PutRaw(arrows,'Fleches '+str(NumberOfEvents
            ),1)
        NumberOfEvents+=1


################################
#
```

```
# Test to catch the created objects !

#test=Object.Get('Mesh.001')
#print test.getEuler()
#test.setEuler([180,0,0])
#test.setName('Arrow.001')
##############################
# Update the 3D scene (to be visible)
NMesh.PutRaw(plane,'TopoPlane',1)
Blender.Redraw()
```

A.2 TomographyLoader.py

This scripts is written to display informations from the tomographic computation made by the force volume analysis software. The series of topographical wiews in gray scale is translated in an animation where the shape changes according to the image series. A similar image series of the Young's modulii is used to animate the change of the stiffness.

```
#!BPY

## Script that import grey scale images and convert in into a 3D
    plane
## in which each verticle height correspond to the gray intensity
    of the
## original image
##
## When the script ended, finalize the animation :
## * In the Editing menu (F9), the Mesh tab uncheck "Relative Keys
    ".
## * Go to : "SRC:1-Animation" or SHIFT-F6
##   to display the IPO curve editor.
## * The plane being selected, choose the ipo type : Shape
##   (by default, Object is selected)
## Make a CTL+LMB to add the Basis curve (Twice to have 2 points)
    and enter
## the coordinate of the first point (Vertex X:0.00 and Vertex Y
    :0.00)
## Make the same for the second point, with the coordinate of
    Vertex X being
## the number of the last image of the animation and the Vertey Y
    the height
## of the blue curve (Key XXX) that define the last image mesh.
##
## (c) Charles Roduit, 2006
## release under the GNU GPL
## Blender 2.41

## Modules importation
from array import array
import Image
import Blender
from Blender import Mesh, NMesh, Scene, Object #, Ipo, Key # Peraps
     for the 2.4x version : Key
baseFileName = '/home/charles/Biologie/Animation/
    StiffnessTomography/Chromosome/First/TEST.000/TEST.000_Deep_'
def OpenImage(_file):

        ## _file is a string that contain the file path and name
        ## to import
        OImage=Image.open(_file)
        (sizeX,sizeY)=OImage.size
        # Gray scale convertion.
        gsImage=OImage.convert('L')
        # Recovery of the grayscales
        # white=0, black=255
        height=[]
        for posX in xrange(sizeX):
                height.append([])
                for posY in xrange(sizeY):
                        # stricly speaking recovery...
                        height[posX].append((gsImage.getpixel((posX
                            ,posY))/255.)*60)
        return height

def createTopo(_height,original=[]):
        ## Mesh object creation :
        ScannedTopography=Mesh.New('ScannedTopography')
        ## Recovery of the data from 'ScannedTopography'
        TopoMesh=NMesh.GetRaw('ScannedTopography')

        ## Plane creation
        #
        # Size initialization...
        sizeX=len(_height)
        sizeY=len(_height[0])
        print('Size of the plane : '+str(sizeX)+' * '+str(sizeY))
        ## Definition of the list of the points that compose the
            plane
        listOfPoints=[]
        for posX in xrange(sizeX):
                for posY in xrange(sizeY):
                        listOfPoints.append([posX,posY,_height[posX
                            ][posY]])
        ## Definition of the list of the faces that compose the
            plane
        listOfFaces=[]
        for y in xrange(sizeY-1):
                for x in xrange(sizeX-1):
                        listOfFaces.append([y+x*sizeY,y+x*sizeY+1,y
                            +(x+1)*sizeY+1,y+(x+1)*sizeY])

        ## Definition of the points of the plane
        for composants in listOfPoints:
                Sommet=NMesh.Vert(composants[0]/50.,composants
                    [1]/50.,composants[2]/50.)
                TopoMesh.verts.append(Sommet)
        ## Definition of the plane faces...
        for face_courante in listOfFaces:
                face=NMesh.Face()
                for numero_vertex in face_courante:
                        face.append(TopoMesh.verts[numero_vertex])
                TopoMesh.faces.append(face)
        ## and we return the Mesh object
        ScannedTopography=NMesh.PutRaw(TopoMesh,'ScannedTopography'
            )
        return ScannedTopography

def modifyTopo(_NewHeight,_OldTopo):
        sizeX=len(_NewHeight)
        sizeY=len(_NewHeight[0])
        print 'input Topo length :'+str(len(_OldTopo.verts))
        listOfPoints=[]
        for x in xrange(sizeX):
                for y in xrange(sizeY):
                        # Vertex Repositionning
                        _OldTopo.verts[y+x*sizeY].co[2]=_NewHeight[
                            x][y]/50.
        return _OldTopo

for tmpImageNb in xrange(22):
        imageNb=tmpImageNb
        if imageNb==0:
                frameNb=1
        else:
                frameNb=imageNb*25
        Blender.Set("curframe",frameNb)
        frameNo=Blender.Get('curframe')
        if frameNo==1:
                ## First passage ==> topo creation
                fileNo=1
                fileName=baseFileName+str(fileNo)+'Topo.tif'
                image=OpenImage(fileName)
                print('Opening image '+fileName)
                planeObject=createTopo(image)
        elif int(frameNo/25)==frameNo/25.:
                fileNo=int(frameNo/25)
                fileName=baseFileName+str(fileNo)+'Topo.tif'
                image=OpenImage(fileName)
                print('Opening image '+fileName)
                plane=NMesh.GetRaw('ScannedTopography')
                plane=modifyTopo(image,plane)
                plane.update()
                plane.insertKey(frameNb,'absolute')
        else :
                fileNo=0
                print('Not a valid frame selected, frame '+str(int(
                    frameNo/25)*25)+' or '+str((int(frameNo/25)+1)
                    *25)+ ' should have been selected')
```

Appendix B

Succellus User Guide

Ecole polytechnique fédérale de lausanne

Institute of Physics of Complex Matter
Laboratory of Physics of Living Matter

Succellus

Succellus User's Guide

Author:
Charles Roduit

Contents

1 Introduction **1**

2 First Use **2**
- I First run 2
- II Create a new experiment 2
- III Understand the user interface 4

3 Advanced Use **8**
- I Young Modulus 9
  - A) Sensitivity Calibration 9
  - B) Save the result 9
  - C) Display the result 9
- II Protein-protein interaction 10
  - A) Tune the event detection 10
  - B) How events are displayed 10
  - C) Detect the events 10
  - D) Post-process 10
- III Tomography 10
  - A) compute the stiffness 11
  - B) Save the result 11
  - C) Display it 11

List of Figures

2.1 Launch Succellus 2
2.2 New experiment 3
2.3 Create new experiment 3
2.4 Succellus Interface 4
2.5 Slice example 6

3.1 sensitivity calibration 9
3.2 Event calibration 10

Chapter 1

Introduction

Succellus is a software to process force-volume files of atomic force microscope (AFM). It can display force curves and topography image recorded by the microscope. This software can also make post-processing, such as computing the Young Moduli map of the scanned area, map of the tip-surface interaction. Those computed data can be displayed by several manner, like map,time-lapse curve, histograms, and exported to spreadsheet software (such as open office calc)

This software was done during my PhD study at the swiss institute of technology of Lausanne (EPFL) on the direction of Sandor Kasas. The main subject was the study of the mechanical properties of the cell in vivo. As no software existed to postprocess AFM files to make such study, we decided to write it. This software was first an extention of a previous one developed in the laboratory by Sandor Kasas, but the need of rewriting from scratch becomes rapidly necessary to implement all the need we have.

As we want to be as multiplatform as possible, we developed this software under Matlab® 7.0. This documentation is done using LaTeX , figures were made using Inkscape, screenshots were taken on GNU/Linux under Enlightenment DR17 and improved with The Gimp.

Chapter 2

First Use

I First run

In order to launch Succellus, you have to run it through Matlab®. Once Matlab® running, go to the directory where you installed Succellus (figure 2.1, ①). Then type Succellus (②) to open the main window (③).

II Create a new experiment

To create a new experiment, select from the file menu the "new" from the "experiment" submenu (figure 2.2). A new window appears to select the path and file you want to load.

Once you have chosen the file to load, a progress bar informs you on the progression of the file loading. Two new windows appears. The first one (Figure 2.3 ①) called "Experiment Properties" permit to save all you want about your experiment (Which kind of tip yo used, the experimental procedure ...) and also the identities of the author(s) of the experiment. The second window (Figure 2.3 ②) ask you if Succellus have to load similar files from this directory. When saving Force-Volume files from Veeco AFM, the nanoscope records the

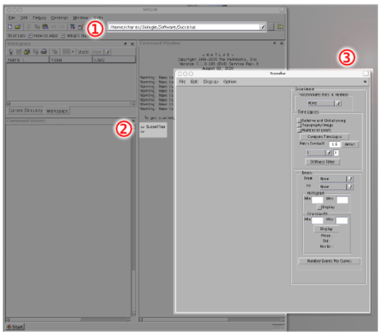
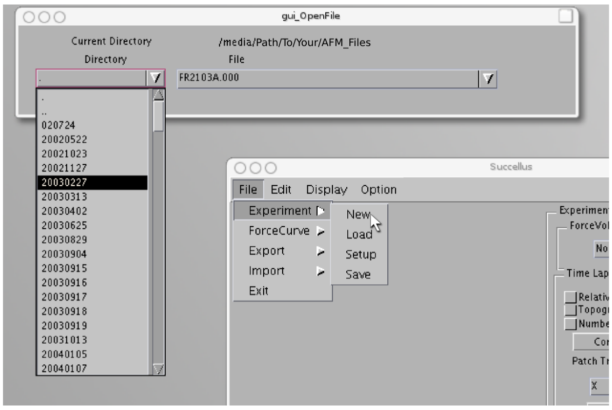


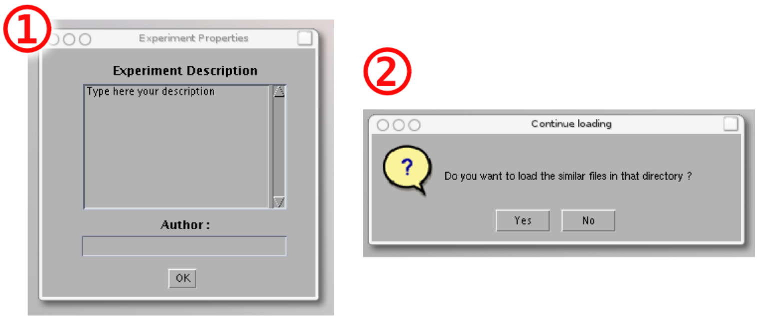
Figure 2.1: To run Succellus, ① go to the directory where Succellus is installed. ② Write the command `Succellus` and ③ the main window appears.



Current Directory /media/Path/To/Your/AFM_Files

File FR2103A.000

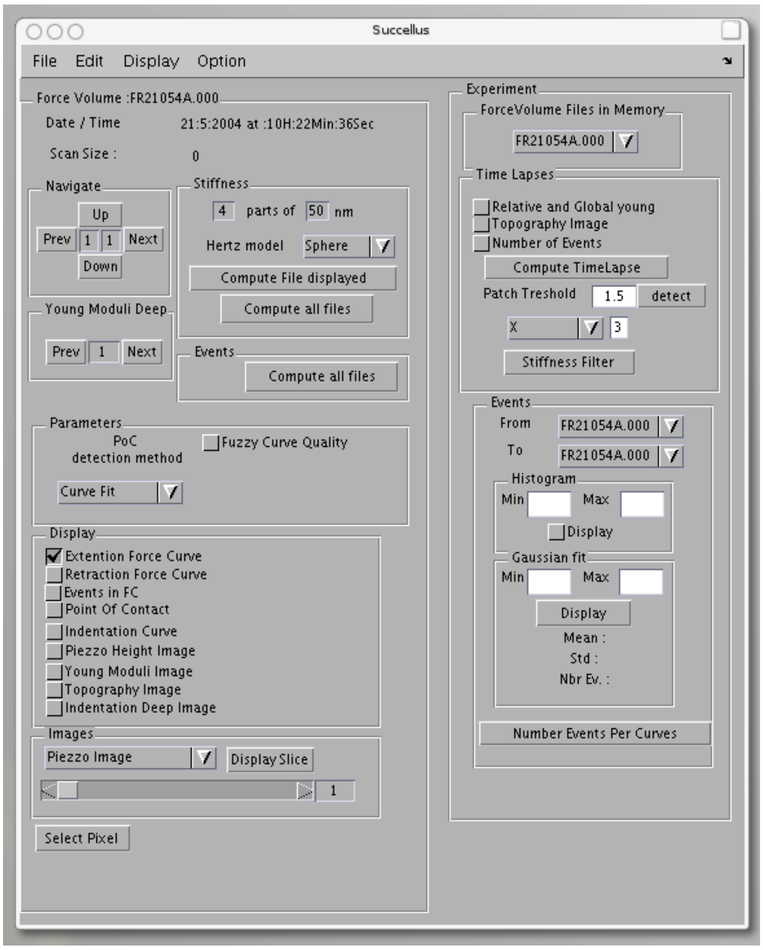
Figure 2.2: *Create a new esperiment*



Type here your description

Do you want to load the similar files in that directory ?

Figure 2.3: *① Window to enter experiment properties and author. ② Popup window asking to automatically load similar files from the current directory.*



Force Volume :FR21054A.000

Date / Time 21:5:2004 at :10H:22Min:36Sec

Scan Size : 0

Navigate: 1 1

Young Moduli Deep: 1

Stiffness: 4 parts of 50 nm

Hertz model: Sphere

PoC detection method: Curve Fit

☐ Fuzzy Curve Quality

Display:
- ☑ Extention Force Curve
- ☐ Retraction Force Curve
- ☐ Events in FC
- ☐ Point Of Contact
- ☐ Indentation Curve
- ☐ Piezzo Height Image
- ☐ Young Moduli Image
- ☐ Topography Image
- ☐ Indentation Deep Image

Images: Piezzo Image, 1

ForceVolume Files in Memory: FR21054A.000

Time Lapses:
- ☐ Relative and Global young
- ☐ Topography Image
- ☐ Number of Events

Patch Treshold 1.5

X 3

Events:

From FR21054A.000

To FR21054A.000

Histogram: Min Max ☐ Display

Gaussian fit: Min Max

Mean :

Std :

Nbr Ev. :

Figure 2.4: Succellus Interface

files by incrementing the file number (e.g. filename.000 is the first file recorded, filename.001 is the second, filename.xyz is the xyz'th file recorded). Succellus understand this and if the first file you load is filename.003 and press 'Yes' to the prompt, it will load filename.004, filename.005, ... untill the end.

III Understand the user interface

The Succellus graphical user interface (GUI) is divided in several zones. In the figure 2.4 is represented how it will present after loading.

The interface is segmented vertically in two principal zones. The left part, the ForceVolume panel, controlls the force volume file in memory. It's name correspond to the file in memory (in the figure, the panel is called `Force Volume :FR21054A.000`). The right part, the Experiment panel groups command to the whole experiment.

ForceVolume panel

The ForceVolume panel is also segmented by several panels.The Navigate, Young Moduli Deep, Stiffness, Events, Parameters, Display, and Images panels.

Navigate panel

Informs from which pixel comes the displayed curve. It allows also do navigate into the matrix by pressing the 'next', 'prev', 'up' and 'down' buttons.

Young Moduli Deep panel

After computing the Young moduli of the force volume files, you can display it. With this panel, you can control which deep is displayed in the 'Young Moduli Image' grid.

Stiffness panel

This panel controlls parameters necessary to perform Young moduli computation. The number of parts and the deep of each parts are defined here. Different Hertz models are avaliabe in order to compute the Young moduli (sphere and cone models). Finally, two computation button are present. The first one 'Compute File displayed' starts the computation of the file in memory. This is useful to control the parameters. The second, 'Compute all files', starts the computation from the first file of the experiment to the last one (if no errors stops the software ;-)

Events panel

This panel has only one button, 'Compute all files' to start the event detection of all files in the experiment. We call here "event" an interaction between protein on the tip and protein on the substrate.

Parameters panel

In this panel can be chosen the point of contact detection method. The standard one is 'Curve Fit' where a fit of the off contact part of the curve determines the position of the contact between the tip and the substrate. Another one, more experimental is the 'Slope Change' where the point of contact is determined by looking at the slope of the force curve.

Display panel

The majority of the data to display can be acccible here. The 'Extention Force Curve' and 'Retraction Force Curve' allows to display the extention (blue) and retraction (red) part of the force curve.

The 'Event if FC' allows to put in evidence, on the retraction force curve, what the software considers as 'Events'. Very useful to tune the event detection.

The 'Point Of Contact' buton activates the display of the point of contact in the extention curve.

The 'Indentation Curve' button allows to display the indentation curve computed from the extention curve.

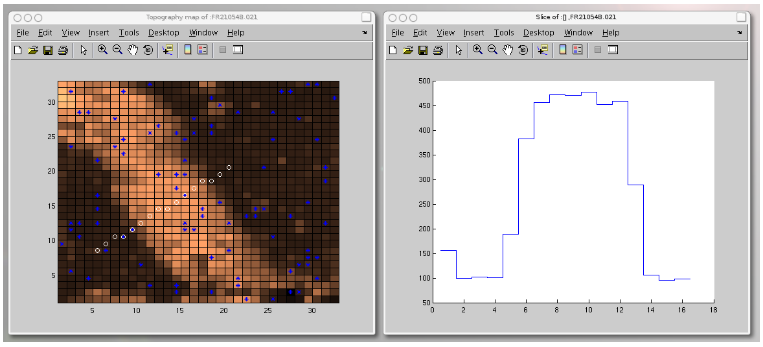

Topography map of :FR21054B.021
Slice of :▯ ,FR21054B.021
File Edit View Insert Tools Desktop Window Help
30
25
20
15
10
5
5 10 15 20 25 30
500
450
400
350
300
250
200
150
100
50
0 2 4 6 8 10 12 14 16 18

***Figure 2.5**: Example of a topographical slice. The white circles describe the path of the slice. The blue stars represent protein-protein interaction detected by Succellus*

The 'Piezzo Height Image' displays the image of the piezzo at the end of the indentation. This image is loaded directly on the forcevolume file from the AFM.

The 'Young Moduli Image' displays the computed young moduli grid. If several depth were computed, the deep displayed can be controlled in the 'Young Moduli Deep' panel.

'Topography Image' displays the "zero force" image. This is a correction of the piezzo height image by the point of contact position in the indentation curve.

'Indentation Deep Image' displays the deepth of indentation image. This correspond to the distance of the point of contact from the end of the indentation curve.

Images panel

The contrast of the images displayed by Succellus can be modified easily through this panel. First choose the image to modify with the popup menu and play with the slider behind it. At the right of the slider is a number (1 by default) that controlls the exponent of the slider value. For example, if there is the number `1` the editable zone, the slider's values comes from 1 (slider is to the left) to 10 (slider is to the right), if there is `5`, the slider's values comes from 10^4 to 10^5.

The button "Display Slice" allows to display a slice view of the targeted window (figure 2.5).

Finally, the Select Pixel allows to point to a particular pixel in the targeted matrix.This is another way to navigate into the matrix.

Experiment panel

The experiment panel allows to control the entire experiment. The commands are also grouped in several horizontal panels.

ForceVolume Files in Memory panel

There is only one popup menu in this panel. All forceVolume scans are listed into this popup. The one selected is controlled trough the ForceVolume panel, the others are silent.

Time Lapses panel

As experiments are principally time lapses, there is a special panel called "Time Lapses". Through it you can display the topography of all sequential scans, (check box "Topography Image").

The first check boxes, "Relative and Global young" is related to the "Compute TimeLapse" button. When pressing this button, a popup window ask for the first file after injection. This means that if you injected a chemical during the time lapse, you can inform the software at what time it is injected. For example, if you inject something between the scan "filename.005" and "filename.006", just write the number "6" and "OK".

After the TimeLapse computation, the "Relative and Global young" check boxe displays three windows containing several graphs. The first one is called "Time Lapse Global". As the title inform us, it displays the evolution of the global stiffness of the scanned area. The second one is called "Time LapseEvent1" and represent the relative stiffness of the pixels presenting protein-protein interaction. Tree graphics are displayed, the relative stiffness at 1, 2 and 3 pixel distance. A value of 1 in the second graphic means that events are as stiff as their neigbours placed 2 pixels afar from it. A value higher than one indicates a stiffer events and lower than one means softer. The last window is called "Time LapseRandom1". As the second window, it displays 3 graphics similarly. The only difference is that the relative stiffness is computed on randomly selected pixel. This is a negative control very usefull to check the specificity of the relative stiffness changes.

The patch threshold is another way to look at the sample stiffness heterogeneity. It computes the relative stiffness map of all scans. Pixels that have a relative stiffness higher than the value entered are considered as stiffer.

Events panel

The events panel permit to view the histogram of the event's force. The check box "Display" displays the whole histogram if no values are entered in Min and Max, or is limited between Min and Max. A Gaussian fit can be estimated by entering the maximum and minimum of the values to consider and pressing the "Display" button. The mean, standard deviation and the number of event considered are displayed under the button, and the fit is drawn on the histogram.

Chapter 3

Advanced Use

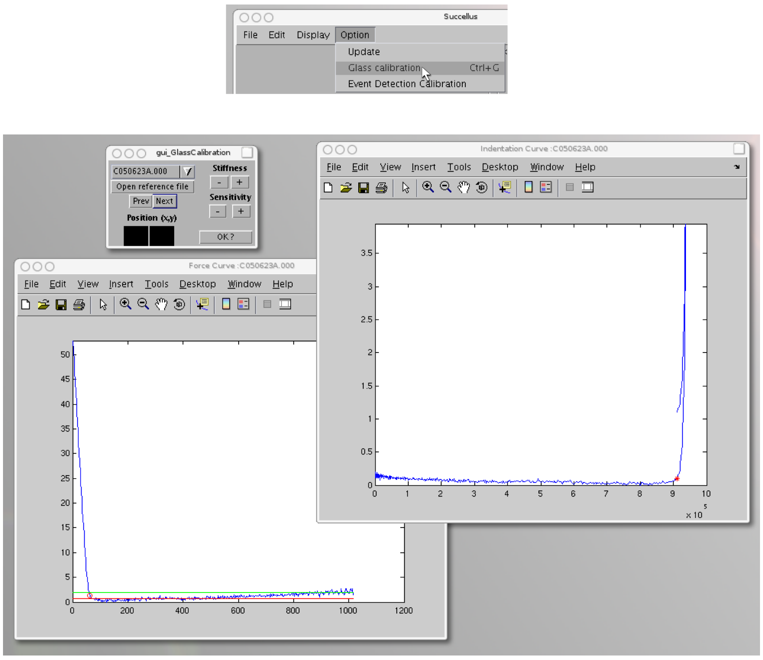


__Figure 3.1__: The top figure shows the menu to get the sensitivity calibration. On the down figure, the three windows shows how to perform the calibration

I Young Modulus

A) Sensitivity Calibration

On figure 3.1 is represented the sensitivity calibration on Succellus. Option → Sensitivity Calibration (or CTL-G)

Click on Stiffness to change the slope of the indentation curve (Sensitivity to change the amplitude of the slope change), so that an indentation on hard sample (e.g. plastic of the petri dish) looks vertical.

In the stiffness panel, choose the size of segment and the number of segment you want to compute.

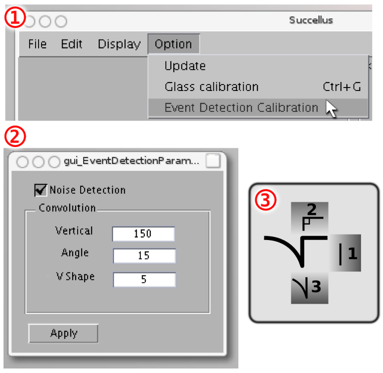
Click on ”Compute file displayed” to compute only the file in memory (i.e. the file chose in the panel ”Experiment” under ”ForceVolume Files in Memory”) or on ”Compute all files” to compute all the files in experiment.

B) Save the result

After the computing is done, you can save the experiment in File → Experiment → Save.

C) Display the result

The Young moduli can be displayed as a color map. To view it, click on the ”Young Moduli Image” check box on the ”display” panel of the ”ForceVolume” group (see figure 2.4 page 4)



☑ Noise Detection

Vertical 150

Angle 15

V Shape 5

Figure 3.2: *To calibrate the event detection, ① choose the "Event Detection Calibration" and ② tune it. The convolution are explain in ③*

II Protein-protein interaction

A) Tune the event detection

To display the detection of protein protein detection, check the "Retraction Force Curve" and "Events in FC" checkboxes in the Display panel (see figure 2.4 page 4).

To tune the event detection, go to the "Event Detection Calibration" option (① in figure 3.2). A new window will appear, it looks like ② in figure 3.2. The checkboxe "Noise Detection" allows you to take care of noise, very usefull for noisy curves, but not recommended for very clean curves. The convolution values permit to tune the parameters of the fuzzy logic algorythm that detects the events. These parameters are explain in the subfigure ③.

B) How events are displayed

When activating the event detection with the checkbox, Succellus detects online the protein protein interaction. A new window will appear, with the name "`Events in Force Curve`". This window is empty if no event is detected and displays the retraction curve if at least one event is detected. The retraction force curve is drawn in red and the detected event in blue, so that, you can see exaclty what the software considers as "event".

C) Detect the events

To do so just click on "compute all files" in the "Events" panel.

D) Post-process

III Tomography

In order to perform stiffness tomography you have to do the following steps :

A) compute the stiffness

Option → Glass Calibration (or CTL-G)

Click on Stiffness to change the slope of the indentation curve (Sensitivity to change the amplitude of the slope change), so that an indentation on hard sample (glass) looks vertical.

In the stiffness panel, choose the size of segment and the number of segment you want to compute. For tomography, choose 0 parts and it will automatically segment all the force curve with segment of the desired size.

Click on "Compute file displayed" to compute only the file in memory (i.e. the file chose in the panel "Experiment" under "ForceVolume Files in Memory") or on "Compute all files" to compute all the files in experiment.

B) Save the result

After the computing is done, you can save the experiment in File → Experiment → Save.

C) Display it

To display 3D view of the tomography, click on the menu Display → Stiffness → Tomography (3D). It will open a new window with a green button "Show Figure". You can click on it to display the whole scan.

With the sliders in the "Grid selection" panel you can slice the scan. With the "Young Selection" and "Deep Selection" you can tune what to display. With the "Color scale adjustment" you can . . . adjust the color scale.

Finally, to have nice animation of the slicing, you can click on "Animate in X" and it generates a series of images slicing in the X direction. You can then make animated gifs with an external software (like with ImageMagick : `convert -delay 20 -loop 0 *.tif anim.gif`).

Appendix C

Succellus Developer's Guide

Ecole polytechnique fédérale de lausanne

Institute of Physics of Complex Matter
Laboratory of Physics of Living Matter

Succellus

Succellus Developer's Guide

Author:
Charles Roduit

Contents

1 Introduction 4

2 Software walking 5
2.1 Opening the software 5
2.2 Creation of a new experiment 5
2.3 Save an experiment 7
2.4 Load an experiment 7
2.5 Navigate into the matrix 8
2.6 Display a force-curve 8
2.7 Display an indentation/force curve 9
2.8 Display matrix 9
2.9 Compute the Young moduli 9
2.10 Detect Events 11
2.11 Compute Time Lapses 12

3 Function Definition 13
3.1 check_AroundMe 14
3.2 compute_CalibrationConstant 15
3.3 compute_CurvePartition 16
3.4 compute_curveSegmentCarac 17
3.5 compute_DeepGride 18
3.6 compute_DeepnessFromPoc 19
3.7 compute_event 20
3.8 compute_eventSlope 22
3.9 compute_extractFromMatrix 23
3.10 compute_gaussFit 25
3.11 compute_GrideMeanPixel 26
3.12 compute_Indentation 27
3.13 compute_meanForEachDeep 28
3.14 compute_MeanSem 29
3.15 compute_PointOfContact 30
3.16 compute_RelativeYoungProperties 31
3.17 compute_SimulateFile 33
3.18 compute_SlopeChange 34
3.19 compute_StiffnessOnDefinedZone 35
3.20 compute_YoungModulus 36
3.21 convert_cellToText 37
3.22 convert_matrixToText.tex 38

3.23 count_numberEventGride 40
3.24 count_numberEventTimeLapse 41
3.25 create_TopoYoungMatrix 42
3.26 delete_FVobjInList 43
3.27 detect_AttractiveEnds 44
3.28 display_plot3D 45
3.29 display_Slice 46
3.30 export_TimeLapseEvents 47
3.31 export_TimeLapseGlobal 48
3.32 export_TimeLapseRelative 49
3.33 export_TimeLapseStiffnessTomo 50
3.34 file_chsuppr 51
3.35 file_clearUpFVNode 52
3.36 file_clearUpTimeLapseRelativeNode 53
3.37 file_extractHeaderDate 54
3.38 file_extractNum 55
3.39 file_lectHeader 56
3.40 file_loadXmlFCFile 58
3.41 file_makeTimeLapseRelativeNode 59
3.42 file_save 60
3.43 find_CurveJump 61
3.44 find_events 62
3.45 forceVolume 63
3.46 generate_EventGride 64
3.47 generate_grideAroundSeries 65
3.48 generate_grideOfAxes 67
3.49 generate_Patches 68
3.50 generate_RelativeTimeLapseVector 69
3.51 generate_vectorForceEvent 70
3.52 get 71
3.53 image_Flatten 72
3.54 image_ThresholdCorrection 73
3.55 import_ExperimentYoungEvent 75
3.56 import_RaftIngEventGride 76
3.57 import_RaftIngYoungGride 77
3.58 load_bioForceCurve 78
3.59 load_bioImageGride 79
3.60 move_FVobjInList 80
3.61 navigate_GridePosition 81
3.62 plot_EventsInFC 82
3.63 plot_ExpSeries 83
3.64 plot_FC 84
3.65 plot_Gride 85
3.66 plot_grideOfPlot 86
3.67 plot_Indentation 87
3.68 plot_Stiffness 88
3.69 plot_TimeLapse 89
3.70 select_Path 90
3.71 select_PixelGride 91
3.72 set 92

4 Modules Description **95**
4.1 Gui 96
4.2 gui_LoadExperiment 100
4.3 gui_LoadForceVolume 101

5 Variables Description **102**
5.1 MainExp 103
5.2 MainFc 106

6 AEX
Afm Exchange XML file Format **112**

Chapter 1

Introduction

Succellus is a software written to postprocess force curves recorded from atomic force microscope. It is develop under Matlab on a Linux platform and known to work under MacOS X and probably under Windows. This software analyses the force curve recorded to extract usefull information. It can detect protein-protein interaction between functionnalized tip and any kind of substrate (coated mica, cell membrane, ...) with the help of a fuzzy algorythm. The mechanical properties of the scanned area is also easily computed with the help of a reference curve taken on a hard sample (glass) from which the user calibrate the indentation curve.

The files generated by this software are called aex files (**A**fm **E**xchange **X**ml). The aex files are organized as experiment. This means that each files are composed by all the AFM files recorded during a single experiment, the description of the experiment, the author, and some results of computations done on the experiment, like time lapses.

This documentation is the developper's documentation. If you are an end user and want to know how to use this software, you should read the User's documentation.

Chapter 2

Software walking

2.1 Opening the software

In the Opening of this software, matlab first needs some initiations. The command to run is `Succellus`. It loads the file `Succellus.m` and `Succellus.mat`. As it is a graphical user interface (GUI), it needs at least Matlab v7.x to run because of a incompatibility with the previous versions. The first action (l.50) is to load the file `Directories.ini` which contains the last used directories to load files, experiments and others and stores it in the global variable `directories`. After this initiation, it waits for the user interaction.

2.2 Creation of a new experiment

Files are organized in the software as "experiments". Each experiment is made of several files saved during the scan of a surface. When building a new experiment (menu File → Experiment → New, calls `gui_NewExperiment`), the software first reinitialize all the values and then calls the function `gui_LoadForceVolume` to load the original files saved by the AFM.

This module makes the three variables `MainFC`, `MainExp` and `directories` as global. These are the only variables that are blobal in this software. Then the user is asked to point the file to load. When a file is given, two action are possible : creation of a new experiment or the addition of a file to an existing experiment. If it is a new "experiment", as it should be in our case, the variables `MainFC` and `MainExp` are empty. `MainExp` is then initialized.

`MainExp.NumberExperiment` contains the number of file(s) in the experiment and is initiate to `0`. `MainExp.Glass.Fit` is the vector defining the fit to a force curve taken in a hard sample. The default value is set to `[1 0]`. `MainExp.Glass.Sensitivity` stores the steps to do when calibrating the curve on hard sample. `MainExp.Figure.GrideOfPlot` contains the pointer to a figure generated when the user wants to display all the scans after the stiffness and event computation.

If a file is given, the software creates the forceVolume object by calling the `forceVolume` function with the pathname and the type of the file (here : 'nanoscope file'). The creation of the object first beginns with the initialization of all its variables. According to the 'nanoscope file' type of the file, it calls `file_lectHeader` to read the header of the file. Two types of AFM files are supported, the "Force Volume" and the "Image", `file_lectHeader` can differentiate between these two and direct the `forceVolume` function to the right place.

In the case of a "Force Volume" file, the `forceVolume` function that creates the object, first loads the force curves by calling `load_bioForceCurve` with several parameters and returns two 3D matrices, `fv.FVDeflectionAvMatrix` and `fv.FVDeflectionReMatrix` which contain respectively all the forward force curves and retraction force curves recorded during the scan. The scale is in nm. Of course, these are only the Y axis of the curves.

The `forceVolume` object creator then calls the `load_bioImageGride` with all its parameters to obtain the `fv.ImageMatrix.Piezzo` which contains all the height values of the scanner at the end of the indentation of each indented points in the scan.

To conclude the forceVolume object creation the X axis of the force curves is generated. First it has to compute a calibration constant, done through the function `compute_CalibrationConstant` and the curve is generated and stored in `fv.FVCurveX`.

The new object, known as `MainFC`, is finally returned to the module `gui_LoadForceVolume`. This new object is then stored at the end of the MainExp.ForceVolumeClass cell, its location inside this cell is also stored inside the object itself through the `set(MainFC,'mainExpIndice', MainExp.NumberExperiment)` command. Some other manipulations are done on the graphical user interface to make visible the presence of a new file inside the experiment.

When the first file is loaded, the software ask the user to load similar files present in the folder. If the user answer 'yes' to this question, the software construct the next file name by incrementing the number at the end of the filename and try to open it. The software continue this loop until the file doesn't exist.

2.3 Save an experiment

To save an experiment (menu File → Experiment → Save, which calls `gui_SaveExperiment` module), the software first updates the MainExp structure and then the user is asked to point the file where to save and calls the function `file_save` with the MainExp structure and the file path to save it.

The `file_save` function is a simple XML parser. For the details of the XML file generated, please report to the DTD file (section 6, page 112). A little remark on the way matlab generate an XML tree. In fact it is no more matlab, but Java.

Node creation

```
docNode = com.mathworks.xml.XMLUtils.createDocument('YourNode');
docRootNode = docNode.getDocumentElement;

%% Create a new element
Element = docNode.createElement('YourFirstElement');
%% put attribute to the element
Element.setAttribute('AttributeName',YourString);
Element.setAttribute('AnotherAttributeName',AnotherString);
%% creating data node
data=docNode.createElement('data');
data.appendChild(docNode.createTextNode(YourDataAsString);
%% linking data node to Element
Element.appendChild(data);
%% link your element to the rootNode
docRootNode.appendChild(Element);
```

2.4 Load an experiment

To load an experiment (menu File → Experiment → Load), the module `gui_LoadExperiment` ask the user to point the file to load through the `gui_OpenFile` function. I had to make this homemade file selector because of a bug in the built-in file selector on the linux platform. If a file is selected, the software clears all variables and loads the experiment file by calling `file_loadXmlFCFile` with the pathname of the file and has in return the MainExp structure which contains all the forceVolume objects and other variables.

The `file_loadXmlFCFile` treats the .aex files as follow. First it uncompress the file which is composed by several xml files. The first one is the experiment.xml file and contains informations about the experiment such as time lapses, description of the experiment and, especially, names of the files that compose the experiment. All these files are stored in xml files which name correspond to the name of the original afm file. For example, information from the file afmfile.001 is stored in afmfile.001.xml. These two types of files are treated differently. The file `experiment.xml` is send to `file_clearUpFVNode` and returns the MainExp structure without the forceVolume classes. The name of the AFM files number n is constructed from the `MainExp.name{n}` from which it adds the `'.xml'` extention. The xml tree is opened from the file and sent to the class constructor, `forceVolume.m` with the string parameter `'xml'`. The `forceVolume` constructor scans the xml tree to store the variables in the class and returns the complete forceVolume class. Finally, when all the forceVolume files are loaded and stored in the `MainExp.forceVoluem` cell, the function `file_loadXmlFCFile` returns the `MainExp` structure to the `file_loadExperiment` function.

2.5 Navigate into the matrix

To navigate inside the scanned area, the user can play with the navigation buttons. The 'Prev' button to move backward, 'Next', forward, 'Up' to move upward and 'Down' to move down in the gride. This graphical manipulations calls the internal functions of `Gui.m` : `prevbutton_Callback`, `nextbutton_Callback`, `Upbutton_Callback` and `downbutton_Callback` respectively. These internal functions modify the ForceVolum object in memory by calling its `set` function with the strings `'navigateFC'` and the direction (`'prev'`, `'next'`, `'up'` or `'down'`) for parameters. This manipulation has for effect to put the desired curve in `fv.FVCurveYAv` and `fv.FVCurveYRe` vectors. In detail, the `set` function calls the `navigate_FC` function with the givent parameters. This latter function refresh the `fv.gridePos.X`, `fv.gridePos.Y`, `fv.gridePos.Abs` and `fv.gridePos.end` informing the position of the curve in the gride, and computed through the `navigate_GridePosition` function. The force curves are directly taken from the `fv.FVDeflectionAvMatrix` and `fv.FVDeflectioReMatrix` matrices.

Finally, the internal function of `Succellus` calls the `refreshAll` which refresh the displayed plots. Thi latter function just call twice the `set` function with once the `'plotFC'` and secondly `'plotIndent'` string parameters. The set function check by itself the curves the user choosed to display in accordance of the state of the switches `fv.CurveToPlot.Av`, `fv.CurveToPlot.Re` and `fv.Display.Indentation`.

2.6 Display a force-curve

To display a force curve, the forceVolume object first needs to be informed on what the user wants to display. There is several switches accessible through the `set` function of the forceVolume object. The first one is to choose which curve to display.

```
% turn on the forward force curve display
MainFC = set(MainFC, 'FC', 'Av', 'On');
% turn off the  forward force curve display.
MainFC = set(MainFC, 'FC', 'Av', 'Off');
% turn on the retraction force curve display
MainFC = set(MainFC, 'FC', 'Re', 'On');
% activate the point of contact display
MainFC = set(MainFC, 'PoC', 'Display', 'On');
```

And finally the `set` function includes a plotFC parameters to plot the activated curve(s) :

```
MainFC = set(MainFC, 'plotFC');
```

It automatically check what to display and where to display it.

2.7 Display an indentation/force curve

Similarly to the force curve display, the `set` function has a switch to turn the indentation display on or off.

```
% to turn on the indentation curve display
MainFC = set(MainFC,'Indentation','Display','On')
% and to plot the curve :
MainFC = set(MainFC,'plotIndent')
```

If the indentation curve display is set to 'off', nothing will happend when using the `set` function to plot the indentation curve. It automatically check what to display and where to display it.

2.8 Display matrix

The scan resulting by the AFM in the force volume mode is an area composed of pixels. These pixels are organized as two dimentional matrices. One matrix is directly recorded during the AFM scan, the piezo height image. The piezo height image is the record of the height of piezo-electic scanner at the end of the indentation. It is stored in the forceVolume object `fv` as `fv.ImageMatrix.Piezzo`. Other matrices are constructed during the post-processing of the file. The deep matrix (`fv.ImageMatrix.Deep`) stores the deepness of indentation computed from the detected point of contact between the tip and the sample. The topography matrix (`fv.ImageMatrix.Topography`) is the corrected image computed from the piezo height matrix and the deepness matrix. It represents the zero force image. The stiffness matrices `fv.YoungModulus` store the stiffness of the scanned area at each depth. For example, the stiffness matrix of the depth n is stored in `fv.YoungModulus[:,:,n]`. All these matrices are viewable through the `set` function with the appropriate string parameter (respectively `'plotPiezzoImg'`, `'plotDeepGride'`, `'plotTopography'` or `'plotStiffness'`. To display the stiffness, the forceVolume object contains the `fv.Display.Stiffness.Deep`, changable by `MainFC = set(MainFC,'Stiffness Deep',DeepNbr` that stores the deep number the user wants to display.

The `set` function calls the `plot_Gride` function to plot the grides. The parameters needed by this funtion is the gride to be displayed, the text displayed as the title of the window, the contrast (a number) and the handler of the window (0 if it doesn't exist). Additionnaly to the display of the matrix, the `plot_Gride` function returns the handler of the window. The stiffness displaying is a little bit different as it calls the `plot_Stiffness` function which takes and additional parameters, the depth the user choosed to diplay.

2.9 Compute the Young moduli

In order to compute the Young modulus of a scan, the user needs to inform some values. The first and most important parameter to be set is the glass calibration (menu option → Glass calibration or CTL+g). The window created by `gui_GlassCalibration` permits to find the parameter that generate a correct indentation curve. The correct indentation curve is the one that does not show indentation in a hard sample. This parameter is stored in two places. Directly in MainExp structure (`MainExp.Glass.Fit` and in the forceVolume object that served for this calibration by the command `MainFC = set(MainFC,'GlassFit',MainExp.Glass.Fit)` which changes the value of `fv.Stiffness.Glass`.

The other parameters are the number of deep, the size of each deep and the Hertz model to fit the indentation curve. Each of these parameters are entered in the GUI and recovered when the stiffness computation is started. There is two button for the stiffness computation, the 'Compute file display' and the 'Compute all files'.

When the user press the 'Compute file display' button, it activates the internal function `pushbuttonCompute_Callback` of `Succellus.m`. The first action is to recover the number of deep, the size of each deep and the Hertz model and store in to the forceVolume object active through the `set` function with the string parameters.

```
# to compute 4 deeps
MainFC = set(MainFC, 'numberIndentationParts', 4);
# each depth has 50nm size
MainFC = set(MainFC, 'indentationDeep', 50);
# and if the sphere is the model chosen
MainFC = set(MainFC, 'HertzModel', 'Sphere');
```

Finally, the computation is done through the `set` function :

```
MainFC=set(MainFC,'Compute','Stiffness');
```

This calls the function `compute_Stiffness`. This function scanns the gride and do comptation for each approach curves.

The first curve analyse is to detect the point of contact between the tip and the sample. This is done with the `compute_PointOfContact` function. As this function returns the indice of the point in the curve where the contact is detected, next this value is converted by `compute_DeepnessFromPoC` into deep and stored in `fv.ImageMatrix.Deep`. At this time, the zero force point (or topography) is stored in `fv.ImageMatrix.Topography`. The next step is to compute the indentation curve, done by the function `compute_Indentation` and the partitionning of the curve, done by `compute_CurvePartition`. The curve partition beginns at the point of contact detected earlier and finish at the end of the curve. These values are then used by `compute_YoungModulus` to return the young modulii computed along this curve and stored in `fv.YoungModulus`. Finnaly, there is a quality curve notation. This detects certain shape of the curves that could alter the trust in the result returned. At the time of redaction, there is only the attractive end detection, through the `detect_AttractiveEnds` function with the FuzzyWalk method.

When the user press the 'Compute all files', it calls the function `gui_compute` through the internal function `AllFileStiffnessCompute_Callback`. This function just make a loop to modify each forceVolume object though the `set` function as explained for the 'Compute file display' button.

2.10 Detect Events

When the user push the "Compute All Files" in the events box, it activates the `pushbutton15_Callback` (l.1137 in Succellus.m). For each experiments, it calls the set function with 'Compute' and 'Event' in parameters (corresponding to l. 176 in set.m) and then calls `compute_event`.

Another way from the Gui to invoke the event detection is to activate the radio button "Events in FC" in the "display" box (l.1133 in Succellus.m) by changing the class variable `FCClass.Compute.Events` from 0 to 1(l.176 in set.m). Then, when plotting the retraction curve, the "plotFC" argument of the class detects if it has to display (and then compute) the event in the curve (l.100 in set.m). If it is the case, it calls `find_events`

In command line it can be done by calling `compute_event` and giving the curveX, curveY gride and a threshold.

The curveX is a vector containing all the x values of the curves. The curveY gride is a 3D gride containing all the curveY recorded during a scan. If the gride is called YGride, the curveY recorded in position x,y in the gride is YGride(x,y,:). The threshold gives the sensitivity to the noise. If 1 is chosen, the detector analyse all points deviating from the noise. If 2 is chosen, all points deviating twice from the noise are analyse. All real positive numbers are accepted. By default, the value 1.5 is chosen.

Each curves are send to the function `find_events` with the curveY, the threshold and the method in parameters. A fuzzy method is used to detect the events.

The fuzzy method first call the function `generate_ConvolutionVectors` which takes in argument only the curveY and returns three vectors, the vertical, angle, V convolution vectors. The convolution vectors contains values along the curveY that reflect the possibility that each positions in the curveY is a vertical, right angle or a V-shape portion respectively. It is then used later to determine the probability that an event exist.

The threshold are then defined. Two possibilities are offered. In the first, the threshold are fix to a certain values and cannot be changed. The second, the threshold are automatically calibrated. In this case, `generate_ConvolutionVectors` computes the three convolution vector from the curveY, but, in addition, returns the threshold values in the "Borne" structure. The treshold values is computed by the quantile value, where 95% of the values are.

In both case, the thresholds are stored in the `Borne` structure are used later for the fuzzy set and organized like this :

`Borne.Vertical` threshold for the vertical fuzzy

`Borne.Angle` threshold for the right angle fuzzy

`Borne.V` threshold for the v shape angle fuzzy

After the generation of convolution vectors, the fuzzy methods calls the `detect_Events` function. It gives in parameters the curveY, the three convolution vectors and the threshold values contained in the "Borne" structure. It returns an event structure containing for each events the curveY portion that defines this event and the indice in the curve.

At the end, we have an event structure containing each events.
For the nth event:

- event{n}.PosX and event{n}.PosY are the position in the scanned gride of the curve containing this event.
- event{n}.x and event{n}.y are the x and y vectors defining this event (i.e. the curve portion that contains this event).
- event{n}.force is the force value of this event.
- event{n}.jumpSlope is the slope of the vertical segment of this event
- event{n}.maxX, event{n}.maxY, event{n}.minX and event{n}.minY are the extremum values of this event.

2.11 Compute Time Lapses

In order to complete a time-lapse computation, the user have to push the button "Compute TimeLapse" in the Time Lapse panel of the Succelus main window (Experiment part). This calls the function `computeTimeLapseButton_Callback` at the line 1001 of the succellus.m file. The function creats MainExp.TimeLapse which will contain the values computed on positive curve, MainExp.TimeLapse-Rand, which will contain values computed on randomly selected curves and MainExp.TimeLapseGlobal which will contain global valuse of the whole scan. The user is prompt about the injection time. This value is stored to inform when, in the experiment, a chemical is added.

For each force-volume files, the function :

Loads event and random gride. If no random gride exist, it generates one with 15 random points. At each depth computed, it loads the young moduli matrix of this depth,and calls the function `compute_RelativeYoungProperties` with the young moduli matrix and the event matrix as argument. The same is done with the Random points with the random matrix in place of the event matrix, and for the global time lapse with three arguments, the young moduli matrix, a matrix filled with ones, and a string 'Global' to inform the `compute_RelativeYoungProperties` function that it treats global values.

The time-lapses values are stored as following :

MainExp.TimeLapse.Deep{deepNbr}.File{fileNbr}.	Events	{eventNbr}
	Event.	mean sem number
MainExp.TimeLapseRand.Deep{deepNbr}.File{fileNbr}.	Events	{eventNbr}
	Event.	mean sem number
MainExp.TimeLapseGlobal.Deep{deepNbr}.File{fileNbr}.	Events	{eventNbr}
	Event.	mean sem number

Then, to access the relative values of the 3^{rd} file at the depth 2, MainExp.TimeLapse.Deep2.File3.Event.mean

Chapter 3

Function Definition

3.1 check_AroundMe

Purpose

Check if a pixel has a non-zero value at a certain distance in a gride.

Syntax

```
eventAround=check_AroundMe(Gride,X,Y,Distance)
```

Description

Given a matrix filled with zeros and non zeros values, test if there is a non zero pixel at a certain distance from the position (X, Y)

Output variables :

eventAround = Number

Returns the number of events the function finds around the selected point.

Input variables :

Gride = Matrix

matrix filled with zeros and non zeros values.

X = Number

Y = Number

such as (X, Y) is a pixel inside the matrix Gride.

Distance = Number

Distance (in pixel) from the given pixel at which the function look around.

3.2 compute_CalibrationConstant

Purpose

Compute the constant used to convert the data from the AFM.

Syntax

```
calib=compute_CalibrationConstant(
Z_SCALE,SensitDeflection,RAMP_SIZE,SensZScan,
NumberPointsPerCurves,ScanRate,SpringConstant)
```

Description

Compute the constant used to convert the data from the AFM.

Output variables :

calib = Struct

Contain the different constant to calibrate the datas.

`calib.FacteurY`

$$\frac{Z\_SCALE * SensitDeflection * SpringConstant}{65536}$$

`calib.FacteurX`

$$\frac{RAMP\_SIZE * SensZScan}{NumberPointsPerCurves}$$

`calib.Vitesse`

$$NumberPointsPerCurves * calib.FacteurX * ScanRate$$

Input variables :

Z_SCALE = Factor used to convert data to indicated units

$$data[nm] = \frac{pixel\,value}{65536} * Z\_SCALE$$

SensitDeflection = 'Sens. Deflection: V'

RAMP_SIZE = ' @4:Ramp size: V'

SensZScan = Sens. Zscan: V'

NumberPointsPerCurves = Number of pixels per scan line

ScanRate = 'Scan rate'

SpringConstant = The spring constant of the cantilever used.

3.3 compute_CurvePartition

Purpose

Cut the indentation curve.

Syntax

```
parts=compute_CurvePartition(Indentation, SegNbr,SegDeep)
```

Description

Output variables :

parts $= (n_0, n_1, ..., n_{SegNb})$

Indices for the selection of the curve parts. The first segment is taken between the n_0 and n_1 points, the second between the $n_1 + 1$ and n_2 points,... the last, between the $n_{SegNbr-1} + 1$ and n_{SegNbr}.

Input variables :

Indentation $= (In_1, In_2, ..., In_i)$

indentation curve coordinate in [nm]

SegNbr = Sn

Number of segment to take out of the indentation curve.

SegDeep = Sd

Deepness of each segments.

3.4 compute_curveSegmentCarac

Purpose

Returns the caracteristic of a segment of curve.

Syntax

```
[slope, constant] =
compute_curveSegmentCarac(curveX,curveY,numberSegment)
```

Description

The function decompose the curve defined by `curveX` and `curveY` in `numberSegment` segments. The slope and constant (such as $y = slope * x + constant$) of each segment are determined. The function returns the result as two vectors. If you ask for one output, the function returns the slope vector.

Output variables :

slope $= (s_1, ..., s_{numberSegment})$

The slope of the respective segments taken on the curve.

constant $= (c_1, ..., c_{numberSegment})$

The constant of the respective segments taken on the curve.

Input variables :

CurveX $= (x_1, ..., x_n)$

Coordinates representing the x value of each points of the curve.

CurveY $= (y_1, ..., y_n)$

Coordinates representing the y value of each points of the curve.

numberSegment = `Integer`

The number of subdivision of the curve you want to perform.

3.5 compute_DeepGride

Purpose

computes the deepness of indentation in the substrate from the detected point of contact to the end in all the scanned area.

Syntax

`deepGride=compute_DeepGride(PoCGride, CurveX, CurveYGride, FitGlass)`

Description

computes the deepness of indentation in the substrate from the detected point of contact to the end in all the scanned area.

calls navigate_GridePosition and compute_DeepnessFromPoC

Output variables :

deepGride $= n * n$ matrix

Input variables :

PoCGride $= n * n$ matrix

Contains the coordinate of the detected point of contact in the curve. Computed by the function compute_PointOfContact

CurveX $= (x_1, x_2, ..., x_p$ matrix

This vector describes the x coordinate of the curve.

CurveYGride $= n * n * p$ matrix

This matrix contains all the curves recorded during the scan of the area.

FitGlass $= = (a, b)$ where $y = ax + b$.

Value of the fit with a force curve taken in a hard sample.

3.6 compute_DeepnessFromPoc

Purpose

computes the deepness of indentation in the substrate from the detected point of contact to the end.

Syntax

```
deep=compute_DeepnessFromPoC(PoC,CurveX,CurveY,FitGlass)
```

Description

computes the deepness of indentation in the substrate from the detected point of contact to the end.

Output variables :

deep = d

The deepness in [nm] calculated between the point of contact and the end of the indentation.

Input variables :

PoC = i

The indice of the point of contact in the curve.

CurveX = $(x_1, x_2, ..., x_i, ..., x_n)$

coordinates in [nm] representing the displacement of the piezzo.

CurveY = $(y_1, y_2, ..., y_i, ..., y_n)$

coordinates in [nm] representing the deflection of the cantilever.

FitGlass = (a, b) where $y = ax + b$.

Value of the fit with a force curve taken in a hard sample.

3.7 compute_event

Purpose

Detects events on a gride coposed of several curves and returns the position of curves with events and all the events.

Syntax

```
[eventGride,allEvents]=compute_event(curveX, curveGride, [threshold])
```

Description

Detects events on a gride coposed of several curves and returns the position of curves with events and a description of all the events. A threshold can be specified.

Output variables :

eventGride = $n * n$ matrix

Represents the position of curves with events detected.
0 means no event detected in the corresponding curve.
1 means at least one event detected in the corresponding curve.

Then if eventGride(x,y)=1, the function has detected an event in the curve curveGride(x,y,:).

allEvents = Cell

- allEvents{nbr}.x
  coordinates in x of the points composing the event
- allEvents{nbr}.y
  coordinates in y of the points composing the event
- allEvents{nbr}.maxX
  X coordinate in pixel in the curve of the end of the event
- allEvents{nbr}.minX
  X coordinate in pixel in the curve of the begining of the event
- allEvents{nbr}.maxY
  Y coordinate in pixel the curve of the top of the event
- allEvents{nbr}.minY
  Y coordinate in pixel the curve of the bottom of the event
- allEvents{nbr}.jumpSlope
  Slope of the jump of contact. Can be a criteria to say if this is really an event or not.
- allEvents{nbr}.force
  Force computed between the top and the bottom. Represents the interraction force between the tip and the surface.

- allEvents{nbrEv}.PosX
  X coordinate of the curve containing the event in the scan gride.
- allEvents{nbrEv}.PosY
  Y coordinate of the curve containing the event in the scan gride.

Input variables :

curveX $= (x_1, x_2, ..., x_i, ..., x_m)$

Coordinates in [nm] of the point of the curve representing the piezzo displacement.

curveGride $= n * n * m$ matrix

Each (n*n*:) vectors are the coordinates in [nm] of the point of the curve representing the cantilever deflection.

threshold = Float

Optionnal value. This is a factor which modulates the effect of noise in the event detection. A value lower than 1 decrease the influence of noise in error detection (then increase the probability of noise detected as event), a value higher than 1 increase the influence of noise in error detection (but real event becomes more susceptible to be detected as noise).

3.8 compute_eventSlope

Purpose

Computes the slope of the jump of contact and the force of events.

Syntax

`events=compute_eventSlope(curveX,events)`

Description

Computes the slope of the jump of contact and the force of events.

Output variables :

events = Cell

- events{nbr}.x
  coordinates in x of the points composing the event
- events{nbr}.y
  coordinates in y of the points composing the event
- events{nbr}.maxX
  X coordinate in pixel in the curve of the end of the event
- events{nbr}.minX
  X coordinate in pixel in the curve of the begining of the event
- events{nbr}.maxY
  Y coordinate in pixel the curve of the top of the event
- events{nbr}.minY
  Y coordinate in pixel the curve of the bottom of the event
- events{nbr}.jumpSlope
  Slope of the jump of contact. Can be a criteria to say if this is really an event or not.
- events{nbr}.force
  Force computed between the top and the bottom. Represents the interraction force between the tip and the surface.
- events{nbrEv}.PosY

Input variables :

curveX = $(x_1, x_2, ..., x_i, ..., x_m)$

Coordinates in [nm] of the point of the curve representing the piezzo displacement.

events = Cell

- events{nbr}.x
  coordinates in x of the points composing the event
- events{nbr}.y
  coordinates in y of the points composing the event

3.9 compute_extractFromMatrix

Purpose

Returns the mean, median of a matrix or in vector form.

Syntax

`valueToReturn[,stdDev[,RestTopo]]=compute_extractFromMatrix(`
`Matrix,toReturn[,Topography,ExcludePercent]`

Description

This function computes the mean or the median of a matrix according to the `toReturn` value. It can also returns the matrix in the form of a vector, rearanging the value in one row. If specified, it can exclude some values according to a `Topography` matrix and a precentage to take from this matrix.

Output variables :

valueToReturn = `Float` or `Vector`

Depends on the `toReturn` value.

stdDev = `Float`

Is the standard deviation of the mean or median computed. `NaN` is returned if a problem occured or if *'Vector'* is set in the variable `toReturn`.

RestTopo = n*n matrix

Filled with `0` and `1`. Represents the values that are or are not taken to compute the mean, median or return the vector, according to the percentage of the topology you choosed.

Input variables :

Matrix = n*n matrix

toReturn = `String`

Determines the value to return. Can be :

'mean'

computes the mean of the matrix.

'median'

computes the median of the matrix.

'vector'

rearange the matrix in a vector.

Topography = n*n matrix

Matrix representing the topography of the scanned area.

ExcludePercent = `Float`

Number of percent of the total height to exclude from the computation. For example, if the total height computed from the `Topography` matrix is 100 and you sets the `ExcludePercent` value to 5, all the values that correspond to a height lower than 5 are not taken in the computation.

3.10 compute_gaussFit

Purpose

Compute a gaussian fit of elements in a vector.

Syntax

```
[x,y,mean,std]=compute_gaussFit(vector,min,max)
```

Description

Compute a gaussian fit of elements borned by min and max in a vector according to the function :

$$y_i = \frac{e^{-\frac{1}{2}*(\frac{x_i - vectMean}{vectStd})^2}}{vectStd * \sqrt{2 * \pi}}$$

Where $vectMean$ and $vectStd$ are respectively the mean value and the standard deviation of the population in the vector.

Output variables :

x $= (x_1, ..., x_i, ..., x_n)$
The x coordinate of the gaussian fit

y $= (y_1, ..., y_i, ..., y_n)$
The y coordinate of the gaussian fit

Input variables :

vector $= (v_1, ..., v_j, ..., v_m)$
The values from where you want to compute the gaussian fit. You don't need to order them.

3.11 compute_GrideMeanPixel

Purpose

Computes the mean value of selected points in a matrix.

Syntax

```
[young, semYoung, numberPoints]=compute_GrideMeanPixel(YoungGride,
PositionGride)
```

Description

Computes the mean, the standard error of mean and the number of values contained in a matrix at selected points. The values are contained in a matrix and the selection of points in this matrix is contained in another matrix filled with 0 and 1, where points containing a non zero value (1) determines the corresponding values in the first matrix to take in the mean and sem computation.

This function calls compute_meansem.m (put reference).

Output variables :

young = Float

Returns the mean of the values taken in the first matrix

semYoung = Float

Returns the standard error of mean of the the values taken in the first matrix

numberPoints = Integer

Returns the numbers of values used to compute the mean and the sem.

Input variables :

YoungGride = $\begin{pmatrix} y_{1,1} & \cdots & y_{1,n} \\ \vdots & \ddots & \vdots \\ y_{n,1} & \cdots & y_{n,n} \end{pmatrix}$

Contains the values to compute the mean, the standard error of mean and determine how many values were taken (NaN are excluded)

PositionGride = $\begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix}$

contains the position of the points you want to compute the mean and the standard error of mean. 0 symbolize non considered points, 1 symbolize points to take.

3.12 compute_Indentation

Purpose

Compute the indentation and force curves.

Syntax

```
[Indentation,Force] =
compute_Indentation(CurveX,CurveY,PoCPos,FitGlass,SpringConst)
```

Description

compute_Indentation compute the indentation and force curves taken the deflection curve.

Output variables :

Indentation $= (I_1, I_2, ..., I_n)$

coordinates in [nm] of the points of the indentation curve.

Force $= (F_1, F_2, ..., F_n)$

coordinates in [Pa] of the points of the force curve.

Input variables :

CurveX $= (x_1, x_2, ..., x_i, ..., x_n)$

coordinates in [nm] representing the displacement of the piezzo.

CurveY $= (y_1, y_2, ..., y_i, ..., y_n)$

coordinates in [nm] representing the deflection of the cantilever.

PoCPos $=$ i

Point of contact between the tip and the substrate.

FitGlass $= (a, b)$ where $y = ax + b$.

Value of the fit with a force curve taken in a hard sample.

SpringConst $=$ s

Spring constante of the cantilever used for the mesurements.

3.13 compute_meanForEachDeep

Purpose

Given a stiffness matrix for all deep, it returns the mean of the matrix for each deep.

Syntax

```
[Mean[,Deviation,[restOfMatrix]]]=compute_meanForEachDeep(
AllStiffMatrix[,TopographyMatrix,removePercent])
```

Description

Given a stiffness matrix for all deep, it returns the mean of the matrix for each deep. You can exclude some values according to a `Topography` matrix and a precentage to take from this matrix. calls compute_extractFromMatrix

Output variables :

Mean = `Vector` of size m

The mean stiffness value for each deep.

Deviation = `Vector` of size m

Optional.

The Deviation value for each deep.

restOfMatrix = $n * n * m$ matrix

Optional.

A matrix filled by 0 and 1. 0 representing the pixels excluded from the mean computation according to the `TopographyMatrix` matrix and to `removePercent` value given in input.

Input variables :

AllStiffMatrix = $n * n * m$ matrix

The 3D matrix containing the values from which you want to compute the mean.

TopographyMatrix = $n * n$ matrix

Optional.

2D matrix containing the topography of the scanned area. It is used with the `removePercent` value to determine which pixel to include or to exclude from the computation.

removePercent = `Integer`

Optional.

Number of high percent you want to exclude from the computation. See `TopographyMatrix` for more detatils.

3.14 compute_MeanSem

Purpose

Compute the mean, the standard error of mean of numbers and the number of values used to compute.

Syntax

`[meanvect, semvect, number]=compute_MeanSem(vecteur)`

Description

Compute the mean, the standard error of mean and the number n of values used to compute.

Output variables :

meanvect = float

returns the mean value of the vector.

$$meanvect = \sum_{k=1}^{n} \frac{v_k}{n}$$

semvect = float

returns the standard error of mean of the vector.

$$sem = \frac{1}{n * \sqrt{n}} * \sum_{k=1}^{n} (v_k - meanvect)^2$$

number = integer

returns the numbers of values in the vector used to compute the mean and the sem. It is different from the length of the vector as it can contain NaN elements.

Input variables :

vecteur = $(v_1, v_2, ..., v_k, ..., v_m)$

contains the set of value you want to compute the mean and sem.

3.15 compute_PointOfContact

Purpose

Detect the point of contact.

Syntax

```
PoC = compute_PointOfContact(CurveX,CurveY,'method',[window])
```

Description

compute_PointOfContac detect the point of contact between the tip and the sample on the given curve. The method of detection can be chosen.

Output variables :

PoC.Position = i

Contain the position of the point of contact on the vectors CurveX and CurveY

PoC.fity = $(f_1, f_2, ..., f_i, ..., f_n)$

Coordinates in [nm] (Y) of the result of the first order fit used to determine the point of contact.

PoC.Errfity = PoC.fity equivalent used to obtain delta

PoC.delta = d

Noise of the portion of the curve used for the fit.

PoC.distDelta = dd

delta+distDelta is the minimal distance from the curve representing the non-contact part.

Input variables :

CurveX = $(x_1, x_2, ..., x_i, ..., x_n)$

coordinates in [nm] of the point of the curve representing the piezzo displacement.

CurveY = $(y_1, y_2, ..., y_i, ..., y_n)$

coordinates in [nm] of the point of the curve representing the cantilever deflection.

'method' = 'CurveFit', 'Manually'

method used to detect the point of contact.

'window' = Integer

Pointer to the window where curves are plotted. Necessary for the method 'Manually'.

3.16 compute_RelativeYoungProperties

Purpose

Computes the properties of the young modulus of selected pixels.

Syntax

```
selectedYoungProp=compute_RelativeYoungProperties(
YoungGride,PositionGride,[KindOfYoungProp,[whereAround]])
```

Description

Computes the properties of the young modulus of selected pixels, relatively to the surrounding pixels or globally, depending on the kind of young modulus proterties you sepcified.

Output variables :

selectedYoungProp = list

- selectedYoungProp{n}.Self
- selectedYoungProp{n}.Around{i}
- selectedYoungProp{n}.Relative{i}

where `i` is the distance and `n` is the number on pixels with events.
if the `KindOfYoungProp` argument is `All Relative`, selectedYoungProp is a 3D matrix where $selectedYoungProp(X, Y, D)$ is the relative young value of the pixel (X, Y) compared to pixels at distance D.

Input variables :

YoungGride = `n*n` matrix

Contains the young modulus of each pixels in the scanned area.

PositionGride = `n*n` matrix

Contains the position where at least one event was detected, symbolized with the number 1. No event detected is marked as 0 in the matrix.

KindOfYoungProp = string (default = 'Relative')

characterize the kind of computation you want to do. Two possibilities :

- `Relative` (default)
  When the position gride are like event gride.
- `Global`
  When you want to compute the young properties on a continuous zone.
- `All Relative`
  When the relative stiffness of each pixel has to be computed.

whereAround = integer (default = 2)

Symbolise the method you want to look around.

2 = horizontally (X)
example at 2 pixels around

```
.....
.....
o.*.o
.....
.....
```

3 = vertically (Y)
example at 2 pixels around

```
..o..
.....
..*..
.....
..o..
```

4 = all around (circle)
example at 2 pixels around

```
..o..
.o.o.
o.*.o
.o.o.
..o..
```

(1 is reserved)

3.17 compute_SimulateFile

Purpose

Importe simulate files generated by Sandor Kasas under ANSYS environment.

Syntax

```
[Indentation,Force,curveParts,YoungModulis]=c
ompute_SimulateFile(fileName)
```

Description

Output variables :

Indentation $= (I_1, I_2, ..., I_n)$

coordinates in [nm] of the points of the indentation curve.

Force $= (F_1, F_2, ..., F_n)$

coordinates in [Pa] of the points of the force curve.

curveParts $= (n_0, n_1, ..., n_{SegNb})$

Indices for the selection of the curve parts. The first segment is taken between the n_0 and n_1 points, the second between the $n_1 + 1$ and n_2 points,... the last, between the $n_{SegNbr-1} + 1$ and n_{SegNbr}.

YoungModulis $= (ym_1, ..., ym_{SegNbr})$

Vector containig the Young moduli at each deepness of the indentation/-force curve. When the curve is not enough deep, it fills the vector with 'NaN' to reach the correct length

Input variables :

fileName = `String`

Is the file name and path to open.

3.18 compute_SlopeChange

Purpose

computes the variation of a curve.

Syntax

`changeInSlope=compute_SlopeChange([curveX,]curveY)`

Description

If only the curveY is send in input, the curve is segmented in several parts. The mean value of each segment is then computed. Then the difference of each adjacent segment mean value represents the variation of the curve. The function retruns the final vector composed by these differences.

If the curveX and the curveY is send in input, the curve is segmented in several parts. The slope of the curve is computed and then the derivative (slope of the slope).

Output variables :

changeInSlope = vector

The size of the resulting vector depends of the size of the input vector(s). If the size of initial vector is n, the size of the resulting vector is $n - \frac{n}{5}$

Input variables :

curveX = $(x_1, ...x_n)$ (Optional)

This optionnal vector defines the x axis.

curveY = $(y_1, ..., y_n)$

This vector defins the y position of each points.

3.19 compute_StiffnessOnDefinedZone

Purpose

Compute the stiffness on a zone defined by the user

Syntax

`MainExp=compute_StiffnessOnDefinedZone(MainExp)`

Description

Compute the stiffness on a zone defined by the user instead of on the whole scan. If no zone were prevoiously defined, the user is promped to select it from the piezzo image of the scan (see section 3.52 and 5.2)

Output variables :

MainExp = structure

The modified MainExp contain the result of the computation on its forceVolume objects (see 5.1 for more details)

Input variables :

MainExp = structure

see 5.1 for more details on the structure of MainExp.

3.20 compute_YoungModulus

Purpose

Compute the Young Modulis

Syntax

`YM = compute_YoungModulus(CuInd,CuFor,CuPart,SegNbr,TipCar,'modèle')`

Description

compute_YoungModulus computes the Young modulis on the selected curve portions, taken the choosen model.

Output variables :

YM $= (ym_1, ..., ym_{SegNbr})$

Vector containig the Young moduli at each deepness of the indentation/-force curve. When the curve is not enough deep, it fills the vector with 'NaN' to reach the correct length

Input variables :

CuInd $= (cI_1, cI_2, ..., cI_i)$

coordinates in [nm] of the points of the indentation curve. cI(1) is the deepest point, cI(i) is the point of contact.

CuFor $= (cF_1, cF_2, ..., cF_i)$

coordinates in [Pa] of the points of the force curve. cF(1) is the deepest point, cF(i) is the point of contact.

CuPart $= (n_0, n_1, ..., n_{SegNbr})$

Indices for the selection of the curve parts. The first segment is taken between the n_0 and n_1 points, the second between the $n_1 + 1$ and n_2 points,... the last, between the $n_{SegNbr-1} + 1$ and n_{SegNbr}. If there is only one point, the function returns a vector of length SegNbr with only 'NaN' inside.

SegNbr = Number of segment choosen when called to compute_CurvePartition. This parameter is useful when the indentation cannot permit to have as segment as you want. The functin fill the the vector with 'NaN' to have the appropriate length.

TipCar = Caracteristic of the tip.

Depending on the hertz model it waits the radius [nm] (Sphere) or the semi-opening angle [rad] (Cone) of the tip.

'modèle' = 'Sphere' or 'Cone'

Modele used for the computation of the Young moduli.

3.21 convert_cellToText

Purpose

Convert a cell structure to a text. Usefull to save variables.

Syntax

```
text=convert_cellToText(cell[,delimiterInsideRow
,[delimiterBetweenRows]])
```

Description

Convert a cell to a text. You can choose the delimiters between the different values inside each elements and between elements inside the cell.

Output variables :

text = `String`

The converted values.

Input variables :

cell = `Cell`

The variable to convert.

delimiterInsideRow = `Char`

Optional.

Character chosen to delimite each value inside an element. If none, '`,`' is the default one.

delimiterBetweenRows = `Char`

Optional.

Character chosen to delimit each element inside the cell. If none, '°' is the default one. You can specify this delimiter only if you specified the previous one (`delimiterInsideRow`)

3.22 convert_matrixToText.tex

Purpose

Convert a matrix structure to a text. Usefull to save variables.

Syntax

`text=convert_matrixToText(matrix,nbCol,nbRow,delimiterInsideRow,`
`delimiterBetweenRows,startCol,startRow)`

Description

Convert a matrix to a text. You can choose the delimiters between the different values inside each elements and between elements inside the cell. Youn can also specify from which column and line you want to start the conversion.

Output variables :

text = `String`

The converted values.

Input variables :

matrix = `Matrix`

The variable to convert.

nbCol = `Integer`

The number of column you want to convert. If you want to convert the whole matrix, enter here the total number of column in the matrix

nbRow = `Integer`

The number of row you want to convert. If you want to convert the whole matrix, enter here the total number of row in the matrix

delimiterInsideRow = `Char`

Optional.

Character chosen to delimite each value inside an element. If none, '`,`' is the default one.

delimiterBetweenRows = `Char`

Optional.

Character chosen to delimit each element inside the cell. If none, '°' is the default one. You can specify this delimiter only if you specified the previous one (`delimiterInsideRow`)

startCol = `Integer`

Optional.

Specifies the column number from which you want to start the convertion.

startRow = `Integer`

Optional.

Specifies the row number from which you want to start the convertion.

3.23 count_numberEventGride

Purpose

Counts the number of events/curves. The result is sorted so that you have the information of the number of curves without events, with one event, two events, ...

Syntax

`NbrEvtVect=count_numberEventGride(eventStruct)`

Description

Returns a vector which contains the number of curves with 0, 1, ..., n events by scanning the `eventStruct` structure.

Elements important in the `eventStruct` structure :

- GridePos = $n * n$ Matrix

  for the size of the scanned area
- each{n}[.PosX & .PosY] = `Integer`

  for their position.

Called by : count_numberEventTimeLapse

Output variables :

NbrEvtVect = `Vector`

NbrEvtVect(1) = nbr of pixels without events.
(2) = nbr of pixels with one event.
...
(nb) = nbr of pixels with (nb-1) events.

Input variables :

eventStruct = `Structure`

Structure containing the matrix of events position and position of each events.

- GridePos = $n * n$ Matrix
  for the size of the scanned area
- each{n}[.PosX & .PosY] = `Integer`
  for their position.

3.24 count_numberEventTimeLapse

Purpose

Function that count the number of event present in an experiment. The result is sorted so that you have the information of the number of curves without events, with one event, two events, ...

Syntax

`Experiment=count_numberEventTimeLapse(Experiment)`

Description

Elements important in the `eventStruct` structure :

- GridePos = $n * n$ Matrix

  for the size of the scanned area
- each{n}[.PosX & .PosY] = `Integer`

  for their position.

Call count_numberEventGride to generate the vector.

Output variables :

Experiment = `Structure`

modified structure.
See section 5.1, page 103. Modifies `Experiment.TimeLapse.NumberEvt`.

Input variables :

Experiment = `Structure`

See section 5.1, page 103

3.25 create_TopoYoungMatrix

Purpose

Creation of the matrix to be display in 3D

Syntax

```
TopoYoung=create_TopoYoungMatrix(
AllYoung,Topography,PiezzoHeight,sizeSegment
[[,Type,borderMin],borderMax]
```

Description

This function creates a matrix that can be plotted in 3D.

Called by : select_3DSlice.m

Output variables :

TopoYoung =

Input variables :

AllYoung = M a $N \times N \times M$ matrix

Where $m_{i,j,k}$ is the k^{th} stiffness of the pixel (i, j).

Topography = T a $N \times N$ matrix

Where $t_{i,j}$ is the topography of the pixel (i, j).

PiezzoHeight = P a $N \times N$ matrix

Where $p_{i,j}$ is the height of the scanner at the end of indentation of the pixel (i, j).

sizeSegment = Float

The sizeSegment represent the size of the segmentation of the curve in nm.

Type = 'Young', 'Deep' or 'None'

This argument inform the function on the target of the limit set with borderMin and borderMax. If the `Type` is set to 'Young', then the borderMin and borderMax are apply on the AllYoung matrix. All pixels which Young values outside of the limit defined are not displayed. And similarly with 'Deep' as argument.

If 'None' is set, then the function does not take care on the borderMin and borderMax argument.

borderMin = Float

Sets the minimum value that is reported in the final matrix.

borderMax = Float

Sets the maximum value that is reported in the final matrix.

3.26 delete_FVobjInList

Purpose

Delete a force volume object in the list.

Syntax

`newExpStruct=delete_FVobjInList(oldExpStruct,FVname,direction)`

Description

Delete the FVname force volume object in the list. The MainExp.ForceVolumeClass is then amputed from this object.

Called by :

- Succellus.m `function DeleteMenu_Callback`

Output variables :

newExpStruct = Struct

The new Experiment structure with the desired forceVolume object amputed.

Input variables :

oldExpStruct = Struct

The Experiment structure from which you want to delet the forceVolume object.

FVname = String

The forceVolume object name to delete. The name is the one in oldExpStruct.name, not the one in the object (FC.name).

3.27 detect_AttractiveEnds

Purpose

Gives a quality note on the shape of the end of the curve.

Syntax

`[Score,FirstFit,LastFit]=detect_AttractiveEnds(CurveX,CurveY,method)`

Description

This functio detects a curvature at the beginning of the curve, when the tip is not in contact with the sample.

Called by :

- compute_Stiffness.m
- plot_FC.m

Output variables :

CurveX = $(x_1, x_2, ...x_n)$

CurveY = $(yav_1, yav_2, ..., yav_n)$

method = String

- 'FuzzyWalk'
  This method uses the fuzzy logic on the first slope, the last slope, the angle and the step to beginn the measurments.

Input variables :

Score = Is the final score, in the range $[0;1]$. A score of zero describes a non attractive and a score of one describes an attractive beginning of the curve.

FirstFit = **Optionnal** Is the fit of the first part of the beginning of the cruve. It was used by the function to compute the slope. This output is used by plot_FC.m to plot in the force curve the position the function detected the first slope.

LastFit = **Optionnal** Is the fit of the second part of the beginning of the cruve. It was used by the function to compute the slope. This output is used by plot_FC.m to plot in the force curve the position the function detected the second slope.

3.28 display_plot3D

Purpose

Plots in 3d the volumes contained in 3D matrix.

Syntax

`display_plot3D(data)`

Description

In this function is specified the colorscale of the YoungModulus

Called by :

- plot_3DYoungView

Output variables :

Nothing = No outputs

Input variables :

data = $x * y * deep$

The 3D matrix containing the information for the colorscale. The pixels that are not displayed have to have the value 0 (zero), NaN doesn't work.

3.29 display_Slice

Purpose

Display a slice of an image.

Syntax

```
figureIndex = display_Slice(
PathX,PathY, Topography, [name, [figureIndex]])
```

Description

Display a slice following a given path through a gride.

Output variables :

figureIndex = Number

Handle of the figure where the plot has been drawn

Input variables :

PathX = $(x_1, x_2, ...x_t)$

such as (x_1, y_1) is the first pixel, ... (x_t, y_t) is the last one.

PathY = $(y_1, y_2, ...y_t)$

such as (x_1, y_1) is the first pixel, ... (x_t, y_t) is the last one.

Topography = $\begin{pmatrix} t_{1,1} & \cdots & t_{1,n} \\ \vdots & \ddots & \vdots \\ t_{n,1} & \cdots & t_{n,n} \end{pmatrix}$

The matrix where information of the topography is stored.

figureIndex =

Handle of the figure where the plot has to be drawn

3.30 export_TimeLapseEvents

Purpose

Export the number of curves with event(s) in a text file.

Syntax

`export_TimeLapseEvents(NbrEventsStruct,baseFileName,InjectionTime)`

Description

Export the number of curves with event(s) in a text file. The first line represents the curves with no events, the second with one, the third with two events and so on untill the maximum of events per curve is reached.

Called by :

- Succellus.m function `ExportNbrEvtMnu_Callback`

Output variables :

None = No output

There is no output as the result of this function is the creation of a file.

Input variables :

NbrEventsStruct = Structure

A structure such that `NbrEventsStruct{n}=nbrOfCurves` is a vector containing the number of curve where `n` event(s) were detected.

For expample, the number returned by `NbrEventsStruct{n}[t]` is the number of curves of the scan number `t` (in the time-lapse) that contains `n` events.

baseFileName = String

This string contain the pathname of the file where the data are saved. By default, it will add the `.csv` extention at the end if there isn't.

InjectionTime = Integer

This number represents the first file after the injection of the buffer (put 1 if there is no injection).

3.31 export_TimeLapseGlobal

Purpose

Exports the global stiffness.

Syntax

`export_TimeLapseGlobal(TimeLapse,baseFileName,InjectionTime)`

Description

Exports the global stiffness contained in the `MainExp.TimeLapseGlobal` in a text file. The first line represents the mean values of each scans, the second, the standard error of mean of each scans, and the following are all values that were taken to give those results.

Output variables :

None = No output

There is no output as the result of this function is the creation of a file.

Input variables :

TimeLapse = Structure

composed like this :

- TimeLapse.Deep{deep}.File{fileNbr}.Event.mean
  representing the mean value of the global stiffness of the file `fileNbr` at the depth `deep`
- TimeLapse.Deep{deep}.File{fileNbr}.Event.sem
  representing the standard error of mean of the mean value of the global stiffness of the file `fileNbr` at the depth `deep`
- TimeLapse.Deep{deep}.File{fileNbr}.Events{evNbre}.self
  collects all the values of all the pixel at the depth `deep` of the file `fileNbr`

baseFileName = String

This string contain the pathname of the file where the data are saved. By default, it will add the `.csv` extention at the end if there isn't.

InjectionTime = Integer

This number represents the first file after the injection of the buffer (put 1 if there is no injection).

3.32 export_TimeLapseRelative

Purpose

Exports the time lapse in a spreadsheet format (.CSV)

Syntax

`varargout=export_TimeLapseRelative(TimeLapse, baseFileName, InjectionTime)`

Description

Exports the time lapse in a spreadsheet format (.CSV).
The first line is the number of the file after the first injection.
The second line mark that what follows is the mean values and separates the several distances.
The third line is the mean values.
The fourth line mark that what follows is the standard deviations and separates the several distances.
The fifth line is the standard deviations.
The sixth line mark that what follows is the number of value used to compute and separates the several distances.
The seventh line are thes values.
The eigth line mark that what follows are all values used to compute and separates the several distances.
The nineth and following lines are all these values.

File → export → TimeLapse → relative Called in Succellus.m in line 1087, 1088.

Output variables :

varargout = None

Specified just for semantic reason.

Input variables :

TimeLapse = Cell

- TimeLapse.Deep{deep}.File{fileNbr}.Relative.mean{dist}
- TimeLapse.Deep{deep}.File{fileNbr}.Relative.sem{dist}
- TimeLapse.Deep{deep}.File{fileNbr}.Relative.number{dist}
- TimeLapse.Deep{deep}.File{fileNbr}.Events{evNbre}.Relative{dist}

baseFileName = String

Contains the base filename of the exported csv with the path. The soft adds in the name the deep information and the csv extention (e.g. baseFileName_Deep_1.csv for the first deep).

InjectionTime = Integer

Indicates the first file after the injection of medium.

3.33 export_TimeLapseStiffnessTomo

Purpose

Export in a text file the values of the stiffness tomography

Syntax

`export_TimeLapseStiffnessTomo(StiffTomoVectStruct,baseFileName,InjectionTime)`

Description

Exports in a text file the mean values of the stiffness tomography.

Output variables :

None = No output

There is no output as the result of this function is the creation of a file.

Input variables :

StiffTomoVectStruct = Structure

`StiffTomoVectStruct.Stdev{fileNbr}(deep)` is the mean stiffness at the depth '`deep`' of the file number '`fileNbr`'

baseFileName = String

This string contain the pathname of the file where the data are saved. By default, it will add the `.csv` extention at the end if there isn't.

InjectionTime = Integer

This number represents the first file after the injection of the buffer (put 1 if there is no injection).

3.34 file_chsuppr

Purpose

Erase some character from a string

Syntax

`ch=file_chsuppr(ch_orig, pos, n)`

Description

Output variables :

ch = String

The original string without character [pos:pos+(n-1)]

Input variables :

ch_orig = String

Original string

pos = Number

Determines the position from where you have to erase.

n = Number

Determines the number of character has to be erased.

3.35 file_clearUpFVNode

Purpose

Returns structure from node.

Syntax

```
cleared=file_clearUpFVNode(rootNode)
```

Description

Returns a structure from a node taken from EAX files. Calls forceVolume.m to create all the force volume object contained in the node.

Output variables :

cleared = Struct

See MainExp in the Variable description chapter to have more information.

Input variables :

rootNode = java node

Is the first node found when parsing the EAX XML file in the function of this toolbox.

3.36 file_clearUpTimeLapseRelativeNode

Purpose

Returns structure from node.

Syntax

`Cleared=file_clearUpTimeLapseRelativeNode(TLNode)`

Description

Returns a structure from a time lapse node taken inside an EAX files.

Output variables :

Cleared = Cell

- Cleared{deep}.File{fileNumber}.Events{numbrEvent}.self
- Cleared{deep}.File{fileNumber}.Event.mean
- Cleared{deep}.File{fileNumber}.Event.sem
- Cleared{deep}.File{fileNumber}.Event.number
- Cleared{deep}.File{fileNumber}.Events{numbrAround}.Around{dist}
- Cleared{deep}.File{fileNumber}.Around.mean{dist}
- Cleared{deep}.File{fileNumber}.Around.sem{dist}
- Cleared{deep}.File{fileNumber}.Around.number{dist}
- Cleared{deep}.File{fileNumber}.Events{numbrRelative}.Relative{dist}
- Cleared{deep}.File{fileNbr}.Relative.mean{dist}
- Cleared{deep}.File{fileNbr}.Relative.sem{dist}
- Cleared{deep}.File{fileNbr}.Relative.number{dist}

Input variables :

TLNode = Java node

Is the node obtained inside the 'TimeLapse' node, corresponding to its child named 'RelativeOnEvent', 'RelativeOnRandom' or 'Global'.

3.37 file_extractHeaderDate

Purpose

Read a text file and extract the date.

Syntax

`date=file_extractHeaderDate(text)`

Description

Read a text where the date is written like '09:00:48 PM Thu Oct 14 2004' and returs a computer comprehesive date.

Output variables :

date = Structure

date.Year = Scalar

date.Month = Scalar

date.Day = Scalar

date.Hour = Scalar

date.Minute = Scalar

date.Second = Scalar

Input variables :

text = String

String containing the date in a form:

`HH:MM:SS month DD YYYY` where H,M,S,D,Y are numbers and month is : 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov' or 'Dec'.

3.38 file_extractNum

Purpose

Extract number from a text.

Syntax

```
Number=file_extractNum(text,'method')
```

Description

Output variables :

Number = Scalar

Input variables :

text = String

Contains the number to extract

'method' = String

Describe from where we wand to scan.

'Reverse' (Default) Scans from the last character

'Forward' Scans from the first character

3.39 file_lectHeader

Purpose

Read the header of nanoscope files

Syntax

`header=file_lectHeader(filepath)`

Description

Given the file path returns all the needed header to let the other functions of the toolbox perform their tasks

Output variables :

header = Structure

This structure contains :

- header.dateTime = Structure
  Contains the time when the file was recorded by the AFM.
  Organization:
  - dateTime.year = Scalar
  - dateTime.month = Scalar
  - dateTime.day = Scalar
  - dateTime.hour = Scalar
  - dateTime.minute = Scalar
  - dateTime.second = Scalar
- header.SensZScan = Scalar
  is the sensitivity of the scan.
- header.OpMode = String
  Contain the operating mode of the AFM.
  For Bioscope files:
  'Force Volume'
  'Image'
- header.NFL
- header.ScanListScanRate
- header.ScanSize
  Size of the scan in [nm].
- header.SensitDeflection
- header.NumberCurvesPerLines
- header.ScanRate
- header.ForceListForwVeloc
- header.ForceListRevVeloc

header.ForceListFVScanRate

header.Z_SCAN_START

header.Z_SCAN_SIZE

header.TTD

header.ImageNumberLines

header.ImageScale

header.ImageSampsPerLine

header.ImageLength

header.ImageTwoOffset

header.ImageOffset
: Indication on where to find the image data in the bioscope file.

header.ForceOffset
: Indication on where to find the ForceVolume data in the bioscope file.

header.NumberPointsPerCurves

header.Z_SCALE

header.HARD_Z_SCALE

header.FV_SCALE

header.RAMP_SIZE

header.SpringConstant
: Contain the spring constant of the cantilever used in [N/m]

Input variables :

filepath = String
: Localisation of the file.

3.40 file_loadXmlFCFile

Purpose

Load AEX files

Syntax

`data=file_loadXmlFCFile(file)`

Description

Load AEX files and returns Experiment structure

Output variables :

data = `Structure`

See MainExp in the Variable description chapter to have more information.

Input variables :

file = `String`

Is the file name and path to open.

3.41 file_makeTimeLapseRelativeNode

Purpose

Creates XML nodes for the file saving.

Syntax

```
[eventTLNode, ]aroundTLNode,
relativeTLNode=file_makeTimeLapseRelativeNode(docNode,deep,deepInTimeLapse)
```

Description

Creates XML TimeLapse nodes for the file saving.

Output variables :

eventTLNode = Java node

aroundTLNode = Java node

relativeTLNode = Java node

Input variables :

docNode = Java node

Main AEX java node

TLDeep = Cell

Cleared = Cell Contains the time lapse

- TLDeep{deep}.File{fileNumber}.Events{numbrEvent}.self
- TLDeep{deep}.File{fileNumber}.Event.mean
- TLDeep{deep}.File{fileNumber}.Event.sem
- TLDeep{deep}.File{fileNumber}.Event.number
- TLDeep{deep}.File{fileNumber}.Events{numbrAround}.Around{dist}
- TLDeep{deep}.File{fileNumber}.Around.mean{dist}
- TLDeep{deep}.File{fileNumber}.Around.sem{dist}
- TLDeep{deep}.File{fileNumber}.Around.number{dist}
- TLDeep{deep}.File{fileNumber}.Events{numbrRelative}.Relative{dist}
- TLDeep{deep}.File{fileNbr}.Relative.mean{dist}
- TLDeep{deep}.File{fileNbr}.Relative.sem{dist}
- TLDeep{deep}.File{fileNbr}.Relative.number{dist}

deepInTimeLapse = Integer

Number of deep computed.

3.42 file_save

Purpose

Save experiment files

Syntax

```
mainExp=file_save(mainExp.xmlFileName)
```

Description

Save experiment files in the AEX file format

Output variables :

mainExp = struct
Contains the Experiments.

Input variables :

mainExp = struct
Contains the Experiments.

xmlFileName = String
Is the file and path name of the file to be saved.

3.43 find_CurveJump

Purpose

Finds jumps in the curve.

Syntax

```
[coord,curve]=find_CurveJump(curve)
```

Description

Finds jumps in the curve by scanning the curve by a window. When the center point of the window is outside the noise, it mark it as a jump.

Output variables :

coord $= (p_1, ..., p_n)$

Marked points have 1 values
Other points are 0

curve $= (c_1, ..., c_n)$

New curve with the markes points having a new value (the mean of the window).

Input variables :

curve $= (c_1, ..., c_n)$

Curve from where you want to detect jumps.

3.44 find_events

Purpose

Detects events in a curve.

Syntax

`[event, newCurve]=find_events(curve, threshold)`

Description

Detects events in a curve. Calls find_CurveJump several times recursively. The function deselect then the curve jump lower than the noise and the lonely ones (an event can't be describe with only one point).

Output variables :

event = Cell

- eventnbrOfEvents.x
  x coordinates composing the event
- eventnbrOfEvents.y
  y coordinates composing the event

newCurve = $nc_1, ..., nc_i, ..., nc_n$

Same as the original curve but without the events.

Input variables :

curve = $c_1, ..., c_i, ..., c_n$

Curve from where you want to detect events.

threshold = Float

Optionnal value. This is a factor which modulates the effect of noise in the event detection. A value lower than 1 decrease the influence of noise in error detection (then increase the probability of noise detected as event), a value higher than 1 increase the influence of noise in error detection (but real event becomes more susceptible to be detected as noise).

3.45 forceVolume

Purpose

Creation of a forceVolume class

Syntax

`fvClass=forceVolume(filepath,method)`

Description

Create a forceVolume class with the appropriate methode according to the file given in argument.

Output variables :

fvClass = forceVolume class

This class contain all variables and functions needed to manimulate the force curves

Input variables :

filepath = String

localisation of the file to be open.

method = String

Specifies which method has to be used to open the file.

'nanoscope File' method used when the file comes directly from the nanoscope software.

'XML' method used when the file was saved using this toolbox.

3.46 generate_EventGride

Purpose

Generates gride filled with 0 and 1.

Syntax

`NewEventGride=generate_EventGride(EventGride,method,number)`

Description

Takes the event gride and returns a gride with dots not corresponding to events in initial gride.

Output variables :

NewEventGride = Matrix

The generated matrix filled with 0 or 1.

Input variables :

EventGride = Matrix

Initial event gride

method = Number

- **1** Random
- **2** Around Vertically
- **3** Around Horizontally
- **4** All around

number = Number

in case of method = 1, sets the number of event you want to generate. If number is set to 0, the function generate as events as it finds in the initial gride.

in case of method =2,3 or 4, sets the distance from each positive pixels we want to generate new pixels.

3.47 generate_grideAroundSeries

Purpose

Creates a series of position gride.

Syntax

`generatedGride=generate_grideAroundSeries(PositionGride,to, whereAround)`

Description

Creates a series of position gride around events. One is position around all events, `n` grides are the postion of one event, and `n` others are the position around each of these `n` events.

Call :

- generate_EventGride.m

Output variables :

generatedGride = list

Organized like this :

generatedGride.AroundAll{dist} :

`dist` grides representing the position around all events at each distance (from `1` to `dist`).

generatedGride.Me{n} :

`n` gride representing the position of each events

generatedGride.AroundMe{n}.distance{dist} :

`n*dist` grides representing the position around each events at each distances (from `1` to `n` and from `1` to `dist`).

Input variables :

PositionGride = `n*n` matrice

Represents the position of the points from where you want to look around. There are symbolized as `1`, and the other points have a null (`0`) value.

to = `dist` integer

Specifies the distances you want to look around (from a distance of 1 to a distance of `dist`). A value of `0` returns an error.

whereAround = integer

Symbolise the method you want to look around.

1 = horizontally (X)
example at 2 pixels around

```
.....
.....
o.*.o
.....
.....
```

2 = vertically (Y)
example at 2 pixels around

```
..o..
.....
..*..
.....
..o..
```

3 = all around (circle)
example at 2 pixels around

```
..o..
.o.o.
o.*.o
.o.o.
..o..
```

3.48 generate_grideOfAxes

Purpose

Creation of a set of axis in a given figure.

Syntax

`axesPosition=generate_grideOfAxes(parentFigure,nbreAxes)`

Description

Create a set of axis in a given figure after destroying all axes present in this figure.

Output variables :

axesPosition = matrix

Contains the pointers for each axes created in the figure.

Input variables :

parentFigure = Number

Is the handle of the figure in which you want to create axes.

nbreAxes = Number

Is the number of figure you want to create in the figure.
Note that the function can create more axis than you expect, because it needs to store in a matrix. For example, a number of axis of 3 returns you 4 axis in a 2x2 matrix.

3.49 generate_Patches

Purpose

Generates patches in a gride at the position where non zero values are in the input matrix, with the given size.

Syntax

`OutputMatrix=generate_Patches(InputMatrix,Method,Size)`

Description

Generates patches in a gride at the position where non zero values are in the input matrix, with the given size.

Call :

- generate_EventGride

WARNING : There is a bug with size higher than 4 pixels, some pixels are not selected. To be fixed.

Output variables :

OuptutMatrix = $N * N$ matrix

Input variables :

InputMatrix = $N * N$ matrix

Method = String

Specifies the patches you want to generate :

- **'X'** To generate horizontal patches
- **'Y'** To generate vertical patches
- **'Surf'** To generate 2D patches

Size = Number

Specifies the size, in pixel, of the patches.

3.50 generate_RelativeTimeLapseVector

Purpose

From a timelapse structure returns vectors (ordered in a structure)

Syntax

`VectorStructure=generate_RelativeTimeLapseVector(Structure)`

Description

From a timelapse structure returns vectors (ordered in a structure). Usefull for exportation.

Output variables :

VectorStructure = Cell

Contains the values in the form of vectors.
These vectors are :

- VectorStructure.Event.mean
- VectorStructure.Event.sem
- VectorStructure.Around.dist{d}.mean
- VectorStructure.Around..sem
- VectorStructure.Relative.dist{d}.mean
- VectorStructure.Around..sem

Input variables :

baseStructure = Cell

Is the structure of MainExp.TimeLapse.deep{p} at a certain p.

- baseStructure.File{n}.Event.mean
- baseStructure.File{n}.Event..sem
- baseStructure.File{n}.Around.mean{d}
- baseStructure.File{n}.Around.sem{d}
- baseStructure.File{n}.Relative.mean{d}
- baseStructure.File{n}.Around.sem{d}

3.51 generate_vectorForceEvent

Purpose

Takes the force value of events in the forceVolum objects and put them in a single vector. Usefull to display histogram of force.

Syntax

`[ForceVector,VectorLength]=generate_vectorForceEvent(MainExp)`

Description

Output variables :

EvForceVector $= \{ef_1, \cdots, ef_i, \cdots, ef_n\}$

ef_i is the force of the i^{th} event

EvLengthVector $= el_1, \cdots, el_i, \cdots, el_n$

el_i is the length of the i^{th} event

Input variables :

MainExp = Structure

3.52 get

Purpose

Function to get some caracteristics of the forceVolume object

Syntax

`Value=get(FVObject,Propertie)`

Description

Output variables :

Value Is the value you asked from the object. It can be a string, a vector, a matrix, ... depending on what you asked.

Input variables :

FVObject = Object

Is the forceVolume object from which you want to extract information.

Properties Properties you can extract:

Compute	All informations about computation
Contrast Exposant	Current value of the exposant of the contrast scroll bar
Contrast Value	Current value of the contrast scroll bar
curveToPlot	Which curve is set to be drown
DeepDisplay	Returns the deepness of the Young Modulus that will be displayed
DeflectMatrix	Matrix NxNxM containing all the deflection force curves
Event Matrix	Matrix NxN containing 1 (one) at the position where an event is detected and 0 (zero) elsewhere
figureIndex	The figure names and their indexes to point to
file	String containing file path and name
FVCurveX	The x vector of the curves
FVCurveYAv	Current advance curve
FVCurveYRe	Current retraction curve
header	All the headers
mainExpIndice	Return his indice in the main experiment container
name	String containing file name
Piezzo Matrix	Piezzo height at the end of the indentation for each pixel in the matrix
PoC Detection Method	Method used to detect the point of contact
PoC Matrix	X coordinate in the deflection force curves of the point of contact
PointTo	Window where next instruction has to be done. eg:contrast,...
posXY	Current x and y position
RetractMatrix	Matrix NxNxM containing all the retraction force curves
Smooth Filter	Return the results of the filter, Low or Low and High frequency
Stiffness Matrix	Return the values of stiffness of the scaned area at the "fv.Display.Stiffness.Deep"
Stiffness Properties	Return all the properties used to compute the stiffness
Topography Matrix	Correction of the piezzo height by the PoC converted in nanometers

3.53 image_Flatten

Purpose

Do a n order flatten on an image matrix

Syntax

`ImageFlattened=image_Flatten(Image,Order)`

Description

Output variables :

ImageFlattened $= N \cdot N$ Matrix

Stored the flatten image.

Input variables :

Image $= N \cdot N$ Matrix

Is the image to be flatten

Order = Number

Degree of the polynome with which the function performs it's flatten. Can be 1, 2, 3, 4 or 5.

3.54 image_ThresholdCorrection

Purpose

Lets you define what is higher than the pixel you clic and what is lower.

Syntax

`calqueMatrix=image_ThresholdCorrection(imageMatrix, [imageHandle, [name, [calqueMatrix]]])`

Description

Creates a matrix filled with 0 and 1 based on the value of the pixels you inserte on the function and where you clic on the gride. The 0 values define the pixels which have a value less or equal than the pixel you selected. The 1 values define the pixels which have a higher value than the pixel you selected. The function displays the image if you doesn't give him a window where it is display (and, in this case, it close this window at the end). When you left clic on the image, it displays the generated matrix and waits for another clic. You can left clic as many times as you want to select the best pixel. When you want to end the function, you have to right clic anywhere in the gride. The function lets you undo one clic, to do so, a middle clic anywhere in the gride restore the last selected pixel.

The function calls the function select_PixelGride (section 3.71, page 91)

Output variables :

calqueMatrix = $N \times N$ matrix

Matrix filled with 0 and 1. The 0 values define the pixels which have a value less or equal than the pixel you selected. The 1 values define the pixels which have a higher value than the pixel you selected.

Input variables :

imageMatrix = $N \times N$ matrix

Matrix used to determine which is higher and which is lower than the selected pixel.

imageHandle = Number

Optionnal

Is the handle of the image matrix (if it is displayed). If not specified, or set as 0, the function display the image matrix in a new figure, and close it when the function terminates.

name = String

Text you want to be display in the figure created by the function.

calqueMatrix $= N \times N$ matrix

Matrix filled with 0 and 1. The 0 values define the pixels which have a value less or equal than the pixel you selected. The 1 values define the pixels which have a higher value than the pixel you selected.

3.55 import_ExperimentYoungEvent

Purpose

Imports young and event from RaftIngFuzzy files.

Syntax

`MainExp=import_ExperimentYoungEvent(MainExp)`

Description

Imports young and event from RaftIngFuzzy files. Function promps the user to indicate the localisation of these files.

Output variables :

MainExp = Cell

Modified MainExp with all the forceVolume object updated with their young and event grides. ForceVolume files are modified through the set function with the argument 'import Event' or 'import Young' followed by the path of files.

Input variables :

MainExp =

Original MainExp.

3.56 import_RaftIngEventGride

Purpose

Imports the position of pixels where at least one event was detected by the RaftIngFuzzy software.

Syntax

`eventGride=import_RaftIngEventGride(FVFileName,grideSize)`

Description

Imports the event postition computed by RaftingFuzzy from the file "FVFileName" with the specified gride size. It returns the events postition in one matrices.

Output variables :

eventGride = `n*n` matrix

matrice composed by 0 or 1, 1 represtents the presence of at least one event, 0 means no event detected at this postition.

Input variables :

FVFileName = string

path and file name from which the young modulus was computed with the RaftIngFuzzy software (eg. /path/name/filename.001). The module converts the file name to the young modulus and the point of contact file name.

grideSize = `n` integer

Specifies the size of the scan gride.

3.57 import_RaftIngYoungGride

Purpose

Imports young modulus computed by the software RaftIngFuzzy

Syntax

`[youngGride, pocGride]=import_RaftIngYoungGride(FVFileName,grideSize)`

Description

Imports the young modulus computed by RaftingFuzzy from the file "FVFileName" with the specified gride size. It returns the young gride and the position of the points of contacts in two matrices.

Output variables :

youngGride = `n*n` matrice

pocGride = `n*n` matrice

Input variables :

FVFileName = string

> path and file name from which the young modulus was computed with the RaftIngFuzzy software (eg. /path/name/filename.001). The module converts the file name to the young modulus and the point of contact file name.

grideSize = `n` integer

> Specifies the size of the scan gride.

3.58 load_bioForceCurve

Purpose

Load the force curves from the bioscope file

Syntax

```
[DEFLECTION_A,DEFLECTION_R] = load_bioForceCurve(file, offset,
NumberCurvesPerLines, NumberPointsPerCurves, Z_SCAN_SIZE,
HARD_Z_SCALE, SensitDeflection)
```

Description

Output variables :

DEFLECTION_A = Matrix

Each points (x,y) of the matrix correspond an approach force curve. So, the point (x,y,i) is the ith point of the curve (x,y).Size of $(NumberCurvesPerLines^2 \cdot NumberPointsPerCurves)$

DEFLECTION_R = Matrix

Each points (x,y) of the matrix correspond a retraction force curve. So, the point (x,y,i) is the i^{th} point of the curve (x,y).Size of $(NumberCurvesPerLines^2 \cdot NumberPointsPerCurves)$

Input variables :

file = String

Filepath of the file from which we want to extract the piezzo image.

offset = Scalar

Tells where are the data in the file.

NumberCurvesPerLines = Scalar

What is the definition used during the scan.

NumberPointsPerCurves = Scalar

Defines the number of points that contains each curves

Z_SCAN_SIZE = Scalar

Used for rescale the data obtained in the file.

HARD_Z_SCALE = Scalar

Used for rescale the data obtained in the file.

SensitDeflection = Scalar

Used for rescale the data obtained in the file.

3.59 load_bioImageGride

Purpose

Load the piezzo image gride from the bioscope file

Syntax

`ImageMatrix = load_bioImageGride(file, offset, NumberCurvesPerLines, ImageNumberLines, ImageSampsPerLine, ImageScale, SensZScan)`

Description

Output variables :

ImageMatrix = Matrix

Each points of the matrix correspond to the altitude of the piezzo at the end of the indentation, in [nm]. Size of $(NumberCurvesPerLines + 1) \cdot (NumberCurvesPerLines + 1)$

Input variables :

file = String

Filepath of the file from which we want to extract the piezzo image.

offset = Scalar

Tells where are the data in the file.

NumberCurvesPerLines = Scalar

What is the definition used during the scan.

ImageNumberLines = Scalar

What is the definition used during the scan.

ImageSampsPerLine = Scalar

What is the definition used during the scan.

ImageScale = Scalar

Used for rescale the data obtained in the file.

SensZScan = Scalar

Used for rescale the data obtained in the file.

3.60 move_FVobjInList

Purpose

Moves a force volume object in the list.

Syntax

`newExpStruct=move_FVobjInList(oldExpStruct,FVname,direction)`

Description

Moves the FVname force volume object in the list.

Output variables :

newExpStruct = Struct

Input variables :

oldExpStruct =

FVname = String

> The forceVolume object name to move. The name is the one in oldExpStruct.name, not the one in the object (FC.name).

direction = String

> Direction where you want to move your object.
> Possible values :
>
> - 'next'
>   Moves the object to the next position (n+1).
> - 'prev'
>   Moves the object to the previous position (n-1).
> - 'last'
>   Moves the object to the last position.
> - 'first'
>   Moves the object to the first position.

3.61 navigate_GridePosition

Purpose

Change the position values

Syntax

```
[Pos,IsEnd]=navigate_GridePosition(PosX,PosY,NbrPixelPerLines,
'direction',[DisplayWin])
```

Description

Changes the position values and tells if we are at the end of the gride

Output variables :

Pos.X, Pos.Y, Pos.Abs New position in the gride

IsEnd Switching variable

=1 if it reached the end of the gride
=0 otherwise

Input variables :

PosX, PosY = x, y

Position in the gride

NbrPixelPerLines = n

Number of pixel per lines. The gride is a square, then it's also the number of pixel per column.

'direction' string

representing the direction where you want to go.

'first' to go to the pixel (1,1)
'next' to go to the next pixel
'prev' to go to the previous pixel
'up' to go to the next line, but the same colume
'down' to go to th previous line, but the same column

DisplayWin Optionnal variable.

Tells if the function has to display a warning window when it has reached an extremity of the gride.

= 0 (default) to not display
= 1 to display

3.62 plot_EventsInFC

Purpose

Plots the curve and the detected events in the same plot.

Syntax

```
figureIndex=plot_EventsInFC(curveX, curveY, events, [name, [figureIndex]])
```

Description

Plots the curve and the detected events in the same plot.

Output variables :

figureIndex = Float

Input variables :

curveX = $(x_1, , ...x_n)$

curveY = $(y_1, ..., y_n)$

events = Cell

See compute_event for a complete description.

name = String

figureIndex = Float

3.63 plot_ExpSeries

Purpose

Display a series of grides contained in an experiment

Syntax

`MainExp=plot_ExpSeries(MainExp,GrideToDisplay)`

Description

This function extracts image matrix contained in all the forceVolume classes in the experiment and display them as a gride in a single window. The images to display are choosen in MainExp

Output variables :

MainExp = struct

Conain the experiment

Input variables :

MainExp = struct

Conain the experiment

GrideToDisplay = String

'Cache' displays the cache (if existed) used to compute the stiffness

'Piezo' displays the piezo height images

'Stiffness' display the stiffness maps

'Clusters' display the clusters detected

3.64 plot_FC

Purpose

Plot the Force Curves

Syntax

`figureIndex=plot_FC(CurveX,CurveYAv,CurveYRe,[name,[figureIndex,[PoCDetection]]])`

Description

Plot the approach and/or the retraction force curve (FC). If we want to plot only one FC, the other needs to be specified as `0`. If we want to detect the point of contact, we have to specifie the name (can be empty string), and the figure index (can be `0` if it doesn't exist) too.

Output variables :

figureIndex =

Is the handle of the window.

Input variables :

CurveX $= (x_1, x_2, ...x_n)$

CurveYAv $= (yav_1, yav_2, ..., yav_n)$ or 0

CurveYRe $= (yre_1, yre_2, ..., yre_n)$ or 0

name = string

figureIndex = number

PoCDetection = 0 or string

accepted values are 0 for no detection or a string accepted by the compute_PointOfContact (page 30) function.

3.65 plot_Gride

Purpose

Plots the matrix

Syntax

```
figureIndex=plot_Gride(matrixToPlot, [text, [Contraste, [figureIndex]]])
```

Description

Plots the matrix in 'copper' color with, optionnaly a text in the window, a given contrast, and targeted in the selected window.

Output variables :

figureIndex = Number

Is the index to hold the correct window.

Input variables :

matrixToPlot $= \begin{pmatrix} m_{1,1} & \cdots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,n} \end{pmatrix}$

Where $m_{i,j}$ are the individual values of the matrix

name = String

Optionnal Variable
Is the text you want to be displayed in the head of the window

Contraste = Number

Optionnal Variable
Sets the contrast of the displayed picture.

figureIndex = Number

Optionnal Variable
Is the index to hold the correct window.

3.66 plot_grideOfPlot

Purpose

Plots several grides in one figure.

Syntax

`figureHandle=plot_grideOfPlot(figureHandle, colorMatrix, [dotMatrix], [text]])`

Description

Plots several color matrices in one figure. In these matrices can be plotted also dots, which are defined in the dotMatrices. The optional value 'text' is used to write in the figure header.

This function use the generate_grideOfAxes function (page 67)

Output variables :

figureHandle = Number

Handle of the figure.

Input variables :

figureHandle = Number

Handle of the figure. If not valide, or figure was closed, the function opens a new one (which handle is return at the end of the function).

colorMatrix = $(n \times n \times m)$

Is the collection of m matrix of size $n \times n$ to be color ploted.

dotMatrix = $(n \times n \times m)$

Is the collection of m matrix of size $n \times n$ used to plot dots in the color plot. The dots are defined with 1, and no dots with 0

text = String

Text you want to see on the top of the figure.

3.67 plot_Indentation

Purpose

Plots the indentation vs force curve

Syntax

```
figureIndex=plot_Indentation(Indentation,Force,CurvePartition,[name,
[figureIndex]])
```

Description

Plots the indentation vs force curve and the curves partition selected for the stiffness computation.

Output variables :

figureIndex = Number

Is the index to hold the correct window.

Input variables :

Indentation = $(I_1, I_2, ..., I_i)$

coordinates in [nm] of the points of the indentation curve. I(1) is the deepest point, I(i) is the point of contact.

Force = $(F_1, F_2, ..., F_i)$

coordinates in [Pa] of the points of the force curve. F(1) is the deepest point, F(i) is the point of contact.

CurvePartition = $(n_0, n_1, ..., n_{SegNbr})$

Indices for the selection of the curve parts. The first segment is taken between the n(0) and n(1) points, the second between the n(1)+1 and n(2) points,... the last, between the n(SegNbr-1)+1 and n(SegNbr).

name = String

Text to appears in the top of the window.

figureIndex = Number

Is the index to hold the correct window.

3.68 plot_Stiffness

Purpose

Plots the stiffness matrix

Syntax

```
figureIndex=plot_Stiffness(matrixToPlot, [deep, [text, [Contraste,
[figureIndex]]]])
```

Description

Plots the matrix in 'jet' color with, optionnaly a text in the window, a given contrast, and targeted to the selected window if already exist.

Output variables :

figureIndex = Number

Is the handle of the correct window.

Input variables :

matrixToPlot = M a $N \times N \times M$ matrix

Where $m_{i,j,k}$ is the k^{th} stiffness of the pixel (i, j).

deep = Number (default = 1)

Optional Variable.
Sets the deepness of the Young modulus you want to display.

text = String

Optionnal Variable
Is the text you want to be displayed in the head of the window

Contraste = Number

Optionnal Variable
Sets the contrast of the displayed picture.

figureIndex = Number

Optionnal Variable
Is the index to hold the correct window.

3.69 plot_TimeLapse

Purpose

Plots the timeLapse

Syntax

`figureHandle=plot_TimeLapse(Structure, [mainTitle, [figureHandle, [comment]]])`

Description

Plots the time lapse in three windows.
One for the global stiffness, one for the relative stiffness on events at each distances computed, and one fore the relative stiffness on random points at each distances computed.

Output variables :

figureHandle = Float

Input variables :

Structure =Cell

mainTitle = String

figureHandle = Float

comment = String

3.70 select_Path

Purpose

Select a path between two pixels

Syntax

`ListOfPixel=select_Path(Window,Size)`

Description

Gives handle to the user The function compute then the pixels defining the path between these two pixels.

This function calls select_PixelGride (page 91)to select the two pixels.

Output variables :

ListOfPixel = Structure

is composed by two vectors which define the coordinate in the gride of the pixels in the path.

ListOfPixel.X = $(x_1, x_2, ...x_t)$

ListOfPixel.Y = $(y_1, y_2, ...y_t)$

such as (x_1, y_1) is the first pixel, ... (x_t, y_t) is the last one.

Input variables :

Window = Number

Is the handle of the figure object. (usefull to know in which window you want to select the path)

Size = Number

Is the size of the matrix displayed on the window. It is used to detect when a clic is made outside of the matrix.

3.71 select_PixelGride

Purpose

Lets the user select a pixel in a given gride.

Syntax

`[PosX,PosY,[button]]=select_PixelGride(Window,Size)`

Description

Lets the user select a pixel in a given gride and return the position (x, y)

Output variables :

PosX $= x$

PosY $= y$

such as (x, y) is the selected pixel position in the gride.

button $= n$

Optionnal
Returns the button with which we clic. In this way, you can make different behaviour depending the button you clic on the matrix.

Window = Number

Handle of the window where the user has to select the pixel

Size = Number

Size of the gride displayed in the selected window. It is used to detect when the user clic outside of the gride, in such a case, the function still wait for the pixel selection.

3.72 set

Purpose

Function to modify the forceVolume object

Syntax

```
FVObject=set(FVObject,Propertie)
```

Description

Output variables :

FVObject = Object

Is the modified forceVolume object

Input variables :

FVObject = Object

Is the initial forceVolume object we want to modify.

Properties Properties you can change and how:

One Parameter	
'CorrectCurveDrop'	modify the end of the curve if it drops down to zero
'plotFC'	Plots the force curve.
'plotPiezzoImage'	Plots the piezzo image Matrix.
'plotIndent'	Plots the indentation/-force curve.
'plotStiffness'	Plots the stiffness matrix.
'plotTopography'	Plots the topography.
'plotDeepGride'	Plots the deep of indentation.

Two Parameters

'navigateFC'	{'next', 'prev', 'up', 'down' of 'reload'}	Navigates through the matrix and puts in memory the corresponding force curves
'Compute'	{'Start', 'Stiffness', 'Event', 'Both', 'Smooth Filter'}	Starts the computation of the selected in the whole matrix.
'numberIndentationParts'	`Number`	Sets the number of parts needed for the stiffness computation.
'indentationDeep'	`Number`	Sets the deep of each parts in the indentation curve.
'HertzModel'	{'Sphere' of 'Cone'}	Sets the model used for the Stiffness computation.
'import Event'	{path}	Imports event gride (look in the path specified).
'import Young'	{path}	Imports young gride (look in the path specified).
'PointTo'	{'Young Moduli Image', 'Stiffness', 'PiezzoImage', 'Topography' or 'Deep Image'}	Set the pointer to the selected window to target the command like contrast.
'Contrast Exposant'	`Number`	Sets the contrast exposant, use to have the correct value from the slider change. Value affected : `fv.Contrast.Exp`
'mainExpIndice	`Number`	Value affected : `fv.MainExpIndice`
'Stiffness Deep'	`Number`	Changes the stiffness displayed when several deep has been computed. Value affected : `fv.Stiffness.Deep`
'GlassFit'	`vector(1,2)`	Changes the value of the slope of the curve in hard sample. Value affected : `fv.Stiffness.Glass`
'Automat'	{'On' or 'Off'}	This Value switches off all navigation warning window. Usefull when doing automated computation on several files. Value affected : `fv.Parameters.Automate`

Three Parameters

'FC'	{'Av' or 'Re'}	{'On' or 'Off'}	Sets the curves to be plotted. Value affected : `fv.CurveToPlot.Av` resp. `fv.CurveToPlot.Re`
'Compute'	{'Stiffness' or 'Event'}	{'On' or 'Off'}	Sets what you want to automatically compute. Value affected : `fv.Compute.Event` resp. `fv.Compute.Stiffness`
'PoC'	{'Display' or 'DetectionMethod'}	{'On' or 'Off'} or {'Curve Fit' or 'Slope Change'}	Value affected : `fv.Display.PoC` resp. `fv.PoC.DetectionMethod`
'Indentation'	'Display'	{'On' or 'Off'}	Value affected : `fv.Display.Indentation`
'Contrast'	{'Stiffness' or 'PiezzoImage'}	`Number`	Value affected are resp. : `fv.Contrast.Stiffness`, `.Piezzo`, `.Corrected` or `.DeepGride`
"Smooth Filter'	{'Method' or 'Size'}	resp. `String` with {'X', 'Y' or 'Surf'} or `Integer`	Values affected are resp.: `MainFC.Parameters.SmoothFilter.Method` or `MainFC.Parameters.SmoothFilter.Size`

Chapter 4

Modules Description

4.1 Gui

Purpose

Main module. Displays the graphical user interface (Gui.m and Gui.fig are necessary).

Button and how they work

Date / Time This is a text field where the date and time when the force curves are recored is displayed. The display is updated when the forceVolume object is change by the popup menu "Force Volume files in memory" in the Experiment pannel (line 602).

ForceVolume - Navigate

Up

Upbutton (line 161)

Passes the 'up' parameters to the function navigate_FC through the set function.
Finish by the internal function refreshAll (line 645).

Down

downbutton (line 172)

Passes the 'down' parameters to the function navigate_FC through the set function.
Finish by the internal function refreshAll (line 645).

Prev

prevbutton (line 138)

Passes the 'prev' parameters to the function navigate_FC through the set function.
Finish by the internal function refreshAll (line 645).

Next

nextbutton (line 150)

Passes the 'next' parameters to the function navigate_FC through the set function.
Finish by the internal function refreshAll (line 645).

ForceVolume - Stiffness

parts

editNumberParts (line 343)

Edits the number of parts you need for the stiffness study. Passes the parameters 'numberIndentationParts' with the number to the set function. The result is the update of the MainFC.Stiffness.SegmentNumber variable.

nm

editDeepIndent (line 369)

Edits the deep of each parts you need for the stiffness study. Passes the parameters 'indentationDeep' with the number to the set function. The result is the update of the MainFC.Stiffness.Deep variable.

Hertz Model

popuHertzModel (line 395)

Popup to choose which model you want to fit the indentation curve.
Passes the parameters 'HertzModel' with'Sphere' or 'Cone' to the set function. The result is the update of the MainFC.Stiffness.HertzModel variable.

Compute File Displayed

pushbuttonCompute (line 429)

Updates first the values MainFC.Stiffness.SegmentNumber, MainFC.Stiffness.Deep and MainFC.Stiffness.HertzModel with the displayed values.
Transfers the Glass.Fit value of MainExp to MainFC through the set function (with the 'GlassFit' and the value for parameter).
The computation is done by the 'Compute','Stiffness' parameter of the set function. Finally the result is copied from MainFC to the corresponding MainExp.ForceVolumeClass.

Compute all files

AllFileStiffnessCompute (line 731)

Sets the value of MainExp.Compute.All to 1 and calls the gui_compute function.

ForceVolume - Young Moduli Deep

Prev

PrevYM (line 717)

Display the deeper young moduli.
It first gets the current deep, and then increase the displayed deep if it is less than the number of computed deep.

Next

NextYM (line 703)

Display the less deep young moduli.
It first gets the current deep, and then decrease the displayed deep. You cannot set a deep less than one.

ForceVolume - Parameters

PoC detection method

PoCDetChange (line 775)

Changes the point of contact detection method.
Sets the MainFC.PoC.DetectionMethod to the wanted value through the set function, with the 'PoC', 'DetectionMethod' and the value (a string).

ForceVolume - Display

Forward Force Curve

DisplayAvFC (line 270)

Display (marked) or not the forward force curve. Update the values of MainFC.CurveToPlot.Av througn the set function with the 'FC','Av' and 'On' or 'Off' argument.
Finally plots the results with the argument 'PlotFC' of the set function.

Reverse Force Curve

displayRevFC (line 286)

Display (marked) or not the reverse force curve. Update the values of MainFC.CurveToPlot.Re througn the set function with the 'FC','Re' and 'On' or 'Off' argument.
Finally plots the results with the argument 'PlotFC' of the set function.

Point Of Contact

checkboxPOC (line 323)

Display (marked) or not the point of contact detected with the choosen method. If the user want to turn off, it first check if the indentation force curve is displayed (). If it is the case, it displays an error explaining that the software needs the point of contact detection in order to display the indentation curve. You first need to turn of the indentation display.
If all is ok, the MainFC.Display.PoC value is set to 1 or 0 to display or not the point of contact through the set function with the 'PoC', 'Display' and 'On' or 'Off' argument.

Indentation Curve

checkboxIndDisplay (line 301)

Display (marked) or not the indentation curve. Changes the MainFC.Display.Indentation boolean value through the set function with the 'Indentation','Display' and 'On' or 'Off' argument.

Piezzo Height Image

DisplayPiezzoImg (line 254)

Display (marked) or not the height of the piezzo at the end of the indentation through the set function with the 'plotPiezzoImg' argument, which displays and returns the MainFC object with the handle of the window (via the plot_Gride function).

Young Moduli Image

checkboxDisplayYM (line 458)

Display (marked) or not the young modulus through the set function with the 'plotStiffness' argument, which displays and returns the MainFC object with the handle of the window (via the plot_Stiffness function).

Topography Image

checkboxDisplayTopography (line 474)

Display (marked) or not the topographical image through the set function with the 'plotTopography' argument, which displays and returns the MainFC object with the handle of the window (via the plot_Gride function).

Indentation Deep Image

checkboxDisplayDeep (line 489)

Display (marked) or not the deep of the indentation through the set function with the 'plotDeepGride' argument, which displays and returns the MainFC object with the handle of the window (via the plot_Gride function).

ForceVolume - Images

Popup

popupImageSelection (line 546)

Changes the window pointer (MainFC.figureIndex.Pointer) to the choosen value via the set function with the 'PointTo' and the choosen window for argument.

Display Slice

ButtonSlice (line 810)

Display the slice of the image displayed in the window pointed through a path. First takes the window handle and then calls the function select_Path and display_Slice.

Slider

sliderContrastValue (line 504)

Changes the contrast of the image displayed in the window pointed.

Exponent

editContrastExponent (line 586)

Changes the exponent of the slice. The slice can have a value from 1 to 10, so the exponent permit to increase the value.

4.2 gui_LoadExperiment

Purpose

Loads AEX files resulting in a MainExp structure in memory.

Variables needed:

directories = list of directories used. First try to load Directories.ini, if it doesnt exist, it is created.

Variables created:

directories = list of directories.

MainFC = See Variable description

MainExp = See Variable description

MainExp.Glass.Fit = $(a; b)$

where $y = ax + b$

Value of the slope in hard sample. Load it from the first experiment. If it doesn't exist, create the null vector.

MainExp.Glass.Sensitivity = n

Is the exposant value of the steps when calibrating the glass.

4.3 gui_LoadForceVolume

Purpose

Loads DI force volume file.

Variables needed:

directories = list of directories used.

First try to load Directories.ini, if it doesnt exist, it is created.

MainExp = Optional

See Variable description (section 5.1, page 103). If it doesn't exist, a new experiment is created

Variables created or modified:

directories = list of directories.

MainFC = See Variable description

MainExp.NumberExperiment = Number

Is incremented by one. Stores the number of force volume object stored.

MainExp = See Variable description

The forceVolume object is placed at the end of the structure `MainExp.ForceVolumeClasse`.

MainExp.name = Structure

This structure stores the names of the force volume files in memory in `MainExp.ForceVolumeClasse`. The order is the same (i.e `MainExp.name{n}` is the name of the `MainExp.ForceVolumeClasse{n}` forceVolume object.

MainFC.mainExpIndice = Number

The indice indicates where in the `MainExp.ForceVolumeClasse` structure the forceVolume object is stored.

FVPanel = gui panel

Set it to visible.

MainExp.Glass.Fit = $(a; b)$

where $y = ax + b$

Value of the slope in hard sample.
Is created if the force volume file is the first file of a new experiment, otherwise it remains unchanged. The default value is $(0; 0)$

MainExp.Glass.Sensitivity = n

Is the exponent value of the steps when calibrating the glass.
Is created if the force volume file is the first file of a new experiment, otherwise it remains unchanged. The default value is 1

Chapter 5

Variables Description

5.1 MainExp

MainExp.Auteur = string

Name of the autor of the experiment.

MainExp.Description =string

Description of the experiment.

MainExp.ForceVolumeClasse = struct

Where are stored the forceVolume object, as structure.
To point the object n :

`MainExp.ForceVolumeClasse{n}`

MainExp.Figure.GrideOfPlot = n

Handle of the figure in which the series of topography and events of each ForceVolume object are display.

MainExp.Glass.Fit = (a,b)

Value `y=ax+b` of the first order fit vith curves on hard sample.

MainExp.Glass.Sensitivity = n

Sensitivity of the correction used for the calibration of the fit with curves on hard sample.

MainExp.name = string

Names of the files include in the experiment.

MainExp.NumberExperiment = n

Number of files include in the experiment.

MainExp.Parameters = Struct

Several computation parameters shared by all forceVolume experiments.

MainExp.Parameters.Event = Struct

Parameters concerning the event computation.

MainExp.Parameters.Event.AngleConv

MainExp.Parameters.Event.DetectNoise

MainExp.Parameters.Event.VerticalConv

MainExp.Parameters.Event.VShape

MainExp.TimeLapse = struct

Contains the result computed by the computeTimeLapseButton_Callback function of Succellus.m, called when clicking the ”Compute TimeLapse” button

.Deep{deepNb}.File{fileNb} = struct

This structure contains the values for the file `fileNb` at the deepness `deepNb`

- **.EventsevtNb** : struct
  The characteristic of the event `evtNb` of the file `fileNb` at the deepness `deepNb` is described here :
  - **.self** : number
    is the stiffness of this event
  - **.Arounddist** : number
    is the stiffness of each pixels aroud this event at the distance `dist`
  - **.Relativedist** : number
    is the relative stiffness of this event at the distance `dist`
- **.Event** : struct The global characteristic of the events of the file `fileNb` at the deepness `deepNb` is described here :
  - **.mean** : number
    is the mean stiffness of the events of this file
  - **.sem** : number
    is the standard error of mean of the stiffness of the events of this file
  - **.number** : number
    is the number of event taken in consideration to compute the two latter numbers.
- **.Arounddist** : struct
  - **.mean** : number
    is the mean stiffness around the events of this file at the distance `dist`
  - **.sem** : number
    is the standard error of mean of the stiffness around the events of this file at the distance `dist`
  - **.number** : number
    is the number of values taken in consideration to compute the two latter numbers.
- **.Relative** : struct
  - **.mean** : number
    is the mean relative stiffness of the events of this file
  - **.sem** : number
    is the standard error of mean of the relative stiffness of the events of this file
  - **.number** : number
    is the number of event taken in consideration to compute the two latter numbers.

.Vector{deepNbr} :

Is the vector for easy plot

- **.Event** :
  Structure containing the following elements :

        - **.mean**
        - **.sem**
        - **.number**
    - **.Around.dist{pixelDist}** :
      Structure containing the following elements :
        - **.mean**
        - **.sem**
        - **.number**
    - **.Relative.dist{pixelDist}** :
      Structure containing the following elements :
        - **.mean**
        - **.sem**
        - **.number**

For example, to access the relative stiffness of the event number 4 of the file 10 at two pixel distance and first deep :

`MainExp.TimeLapse.Deep{1}.File{10}.Events{4}.Relative{2}`

MainExp.TimeLapseRand

MainExp.TimeLapseGlobal

5.2 MainFc

MainFC.calib.FacteurX = number

Contain the calibration constant to create the CurveX.
Created in @forceVolume/calibrationConstant.m

MainFC.calib.FacteurY = number

Contain the calibration constant to calibrate the CurveYAv and Re.
Created in @forceVolume/calibrationConstant.m

MainFC.calib.Vitesse = number

Contain the calibration constant.
Created in @forceVolume/calibrationConstant.m

MainFC.Compute.Events = Number

=1 to detect the events.
=0 to not compute it.

P.S. as the event detection is not yet implemented, the value has no effects on the computation.

MainFC.Compute.Stiffness = Number

=1 to compute the stiffness.
=0 to not compute it.

MainFC.Current.CurvePartition = Vector

Contain the indices of each partition selected on the force/indentation curve. Used to compute the several stiffness.

MainFC.Current.PoC.delta = Number

MainFC.Current.PoC.distDelta = Number

MainFC.Current.PoC.Errfity = Vector

MainFC.Current.Force = Vector

Contain the force curve of the current gride position.
Asked by textttcompute_CurvePartition.

MainFC.Current.Indentation = Vector

Contain the indentation curve of the current gride position.
Asked by textttcompute_CurvePartition.

MainFC.Current.PoC.Position = Number

MainFC.CurveToPlot.Av = Number

=1 to display the approach curve.
=0 to not display it.

MainFC.CurveToPlot.Re = Number

=1 to display the retration curve.
=0 to not display it.

MainFC.PoC.DetectionMethod = String

Specifies the way to detect the point of contact.
Values are :
`'Mannual'`,
`'SlopeChange'`
`'CurveFit'`

MainFC.Display.Indentation = Number

Indicates if the user wants to display the indentation curve
Values are :
`= 1` to display
`= 0` to not display

MainFC.Display.PoC = Number

Indicates if the user wants to display the point of contact
Values are :
`= 1` to display
`= 0` to not display

MainFC.Event.GridePos = Matrix

Contains the position of the detected events.

MainFC.Event.GrideRand = Matrix

Contains the random generated gride, to statistical studies (to compare with the event gride).

MainFC.Events = Cell

Contain the description of each events.

`MainFC.Events{nbr}.x` contains the x coordinate of the event number `nbr`.

`MainFC.Events{nbr}.y` contains the y coordinate of the event number `nbr`.

`MainFC.Events{nbr}.maxY` contains the higher y value of the event number `nbr`.

`MainFC.Events{nbr}.minY` contains the lower y value of the event number `nbr`.

`MainFC.Events{nbr}.maxX` contains the x value corresponding to the higher y value of the event number `nbr`.

`MainFC.Events{nbr}.minX` contains the x value corresponding to the lower y value of the event number `nbr`.

`MainFC.Events{nbr}.jumpSlope` contains the slope of the jump during the unbouding event ($\frac{maxY-minY}{maxX-minX}$)

MainFC.figureIndex = Struct

Contain the indexes of the displayed figures
They are :

`MainFC.figureIndex.DeepGride` links to the window where the deepness of indentation is displayed.

`MainFC.figureIndex.FV` links to the windows where force-curves are plotted.

`MainFC.figureIndex.PiezzoImage` links to the window where the height of the piezzo is displayed.

`MainFC.figureIndex.Pointer` Specifies the window where the command should be send.

`MainFC.figureIndex.Indentation` links to the window where the force/ indentation curves are plotted.

`MainFC.figureIndex.Stiffness` links to the window where the stiffness of the sample is displayed.

`MainFC.figureIndex.Topography` links to the window where the topography is displayed.

MainFC.file = string

contain the link to the bioscope file.

MainFC.ForceMatrix = Not used

I tried this to store all the force curves. But as they have all different length.... Perhaps sould I try to store the whole curve, without thinking about point of contact...

MainFC.FVCurveX = Vector

Contains the X values of the curve in memory.

MainFC.FVCurveYAv = Vector

Contains the Y values of the approche curve in memory.

MainFC.FVCurveYRe = Vector

Contains the Y values of the retraction curve in memory.

MainFC.FVDeflectionAvMatrix = 3D matrix

Contain the approche force curves.
It's size is (NxNxM)
where N is equal to MainFC.header.NumberCurvesPerLines,
and M is equal to MainFC.header.NumberPointsPerCurves

MainFC.FVDeflectionReMatrix = 3D matrix

Contain the retraction force curves.
It's size is (NxNxM)
where N is equal to MainFC.header.NumberCurvesPerLines,
and M is equal to MainFC.header.NumberPointsPerCurves

MainFC.FV.IndentMatrix = Not used

I tried this to store all the indentation curves. But as they have all different length.... Perhaps sould I try to store the whole curve, without thinking about point of contact...

MainFC.gridePos.Abs =1

Numbre containing the absolute position in the gride of the curve in memory

MainFC.gridePos.X = Number

Numbre containing the X position in the gride of the curve in memory

MainFC.gridePos.Y = Number

Numbre containing the Y position in the gride of the curve in memory

MainFC.header = struct

contain the headers

MainFC.ImageMatrix.Corrected = 2D matrix

Contain the total height of the sample at each points of the scanned area.

MainFC.ImageMatrix.Deep = 2D matrix

Contain the deep of indentation in the sample at each points of the scanned area.

MainFC.ImageMatrix.Piezzo = 2D matrix

Contain the height of the piezzo at the end of the indentation each points of the scanned area.

MainFC.ImageMatrix.PoC = 2D matrix

Contain the indice on the curve of the point of contact between the tip and the sample each points of the scanned area.

MainFC.ImageMatrix.SmoothFilterHighFrequency = 4D matrix

$SFHL = (PosX, PosY, Deep, Size)$ represents the value at the given position $(PosX, PosY)$ in the deepth $Deep$ with patch size $Size$.

MainFC.ImageMatrix.SmoothFilterLowFrequency = 4D matrix

$SFLL = (PosX, PosY, Deep, Size)$ represents the value at the given position $(PosX, PosY)$ in the deepth $Deep$ with patch size $Size$.

MainFC.mainExpIndice = number

contain the position of the file in the MainExp structure.

MainFC.name = string

contain the name of the file.

MainFC.Parameters = Structure

Contain all parameters used for the computation. (see compute_* for more details)

MainFC.Parameters.Deep =[]

MainFC.Parameters.delta =[]

MainFC.Parameters.distDelta =[]

MainFC.Parameters.force =[]

MainFC.Parameters.Glass =[]

MainFC.Parameters.SegmentNumber =[]

MainFC.Parameters.Stiffness.HertzModel =[]

MainFC.Parameters.PoC.DetectionMethod =[]

MainFC.Parameters.SmoothFilter.Size = Integer

Represents the maximum size of the patches to apply when doing the smooth filter.

Use in `set` function with parameters `'compute','Smooth Filter'`
Set in `set` function with parameters `'Smooth Filter','Size`

MainFC.Parameters.SmoothFilter.Method = String

Represents the method to apply when doing the smooth filtering of the young modulus. **'X'** to do one dimentional patches horizontally, **'Y'** to do one dimentional patches horizontally, and **'Surf'** to do two dimentional patches.

Use in `set` function with parameters `'compute','Smooth Filter'`
Set in `set` function with parameters `'Smooth Filter','Size`

MainFC.path = string

contain the path of the bioscope file.

MainFC.PoC.Current = Number

Contain the indice of the detected point of contact on the curve currently in memory. So if `MainFC.PoC.Current = i`, the point defined by `MainFC.FVCurveX(i),MainFC.FVCurveYAv(i)` is the point of contact.

MainFC.PoC.force = Number

Used to force the point of contact detection, even if it was already computed.
`= 1` (default) force detection
`= 0` doesn't force
PS. This is a future feature, it has no effect now.

MainFC.Pointer = String

Specifies on which figure we want to target specialized instructions (like setting the contraste,...).
Values are:

`Stiffness`

`PiezzoImage`

MainFC.Stiffness.Deep = Number

Specifies the deepness in [nm] of the indentation curve we want to compute each stiffness.
default = `50`

MainFC.Stiffness.Glass = (a, b)

Where $y = ax + b$
Specifies the value of the fit with a force curve taken in a hard sample.
This value can be changed by `@forceVolume/set.m`
This value is used by the function `plot_Indentation`
`MainFC.Parameters.Glass` is update by this value for computation. Only the updated value is used for computation of young moduli and is stored in the saved file. So `MainFC.Stiffness.Glass` is a temporary item to fine calibrate.

MainFC.Stiffness.HertzModel = String

Specifies which Hertz model we want to use to compute the stiffness.
Values are:
`'Sphere'` (default)
`'Cone'`

MainFC.Stiffness.SegmentNumber = Number

Specifies the number of stiffness (each one having a deep of `MainFC.Stiffness.Deep` [nm]) we want to compute.

MainFC.Stiffness.SemiAngle = Number

Contain the semi-opening angle [rad], used if the tip is modelized as a cone (i.e. `MainFC.Stiffness.HertzModel='Sphere'`)
default = $(45/2) * pi/180$

MainFC.Stiffness.TipCar.Radius = Number

Contain the radius of the sphere, used if the tip is modelized as a sphere (i.e. `MainFC.Stiffness.HertzModel='Sphere'`)
default = `40`

MainFC.test =0

Internal switcher to make test (if set to 1)

MainFC.YoungModulus = 2D Matrix

Contain the Young moduli

Chapter 6

AEX
Afm Exchange XML file Format

```
<!DOCTYPE ForceCurveAnalyse [
  <!ELEMENT ForceCurveAnalyse (date, auteur, description, NumberForceVolumeFiles,
      ForceVolumeNames, TimeLapse+)>
    <!-- date specifie la date de creation du fichier -->
    <!ELEMENT date (#PCDATA)>

    <!ELEMENT auteur (#PCDATA)>
    <!-- "description" contient un court texte decrivant l'experience realisee -->
    <!ELEMENT description (#PCDATA)>
    <!NumberForceVolumeFiles (#PCDATA)>
    <!ForceVolumeNames (#PCDATA)>
 <!ELEMENT TimeLapse (RelativeOnEvent+,
   <TimeLapse FileInjection (#PCDATA)>
      <RelativeOnEvent>
         <ListOfEvents File (#PCDATA)>
            <Events DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Around DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            </Around>
         </ListOfEvents>
         <ListOfEvents File (#PCDATA)>
            <Events DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Around DeepNbr (#PCDATA)>
               <data Distance (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Relative DeepNbr (#PCDATA)>
               <data Distance (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
      <RelativeOnRandom>
         <ListOfEvents File (#PCDATA)>
            <Events DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Around DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            </Around>
         </ListOfEvents>
         <ListOfEvents File (#PCDATA)>
            <Events DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Around DeepNbr (#PCDATA)>
               <data Distance (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
            <Relative DeepNbr (#PCDATA)>
               <data Distance (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
      <Global>
         <ListOfEvents File (#PCDATA)>
            <Events DeepNbr (#PCDATA)>
                <!ELEMENT description (#PCDATA)>
]>
```

```
<!ELEMENT forceVolumeFile (description,header,image,forceCurve)>

  <!-- date spécifie la date de création du fichier -->
  <!ATTLIST forceVolumeFile date CDATA #IMPLIED>

  <!ATTLIST forceVolumeFile auteur DATA #IMPLIED>
  <!-- "description" contient un court texte décrivant l'expérience réalisée -->

  <!ELEMENT description (#PCDATA)>
  <!-- L'élément "Header" contient les entêtes traités par le programme -->

  <!ELEMENT header EMPTY>
    <!ATTLIST header SensZScan CDATA #REQUIRED >
    <!ATTLIST header OpMode CDATA #REQUIRED >
    <!ATTLIST header ScanSize CDATA #REQUIRED >
    <!ATTLIST header NFL CDATA #REQUIRED >
    <!ATTLIST header ScanListScanRate CDATA #REQUIRED >
    <!ATTLIST header SensitDeflection CDATA #REQUIRED >
    <!ATTLIST header ScanRate CDATA #REQUIRED >
    <!ATTLIST header ForceListForwVeloc CDATA #REQUIRED >
    <!ATTLIST header ForceListRevVeloc CDATA #REQUIRED >
    <!ATTLIST header NumberCurvesPerLines CDATA #REQUIRED >
    <!ATTLIST header ForceListFVScanRate CDATA #REQUIRED >
    <!ATTLIST header Z_SCAN_START CDATA #REQUIRED >
    <!ATTLIST header Z_SCAN_SIZE CDATA #REQUIRED >
    <!ATTLIST header TTD CDATA #REQUIRED >
    <!ATTLIST header ImageOffset CDATA #REQUIRED >
    <!ATTLIST header ImageLength CDATA #REQUIRED >
    <!ATTLIST header ImageSampsPerLine CDATA #REQUIRED >
    <!ATTLIST header ImageNumberLines CDATA #REQUIRED >
    <!ATTLIST header ImageScale CDATA #REQUIRED >
    <!ATTLIST header ForceOffset CDATA #REQUIRED >
    <!ATTLIST header NumberPointsPerCurves CDATA #REQUIRED >
    <!ATTLIST header SpringConstant CDATA #REQUIRED >
    <!ATTLIST header Z_SCALE CDATA #REQUIRED >
    <!ATTLIST header HARD_Z_SCALE CDATA #REQUIRED >
    <!ATTLIST header FV_SCALE CDATA #REQUIRED >
    <!ATTLIST header RAMP_SIZE CDATA #REQUIRED >
  <!-- L'élément "image" contient les "altitudes" mesurées du piezzo à la fin de l'
      indentation -->

  <!ELEMENT image (#PCDATA)>
    <!ATTLIST image contrast CDATA #IMPLIED>
  <!-- "pointOfContact" contient les position (en pixel) des points de contactes détectés
       sur les courbe de force -->

```

```
 <!ELEMENT pointOfContact (#PCDATA)>
  <!ATTLIST pointOfContact method CDATA #REQUIRED>
  <!ATTLIST pointOfContact contrast CDATA #IMPLIED>


<!-- topography contient les informations sur la topographie de la zone scannée -->
<!ELEMENT topography (#PCDATA)>

<!ELEMENT event (data)>
  <!-- 'curveID' désigne le numéro de la courbe auquel l'évènement est attaché -->
  <!ATTLIST event curveID IDREF #REQUIRED>
  <!-- "position" se réfère à la position de l'évènement sur la courbe curveNb, en nm
      -->
  <!ATTLIST event position CDATA #REQUIRED>
  <!--- "grade" comporte des grades (verticalité, angle obtu, angle droit, PoC, Moyenne
      ) décrivant l'évènement détecté -->
  <!ATTLIST event gradeMean CDATA #REQUIRED>
  <!ATTLIST event gradeVert CDATA #REQUIRED>
  <!ATTLIST event gradeVAnge CDATA #REQUIRED>
  <!ATTLIST event gradeRAngle CDATA #REQUIRED>
  <!ATTLIST event gradeVPoc CDATA #REQUIRED>
  <!-- "data" contient les points composant la courbe de rétraction faisant parti de l'
      évènement détecté-->
  <!ELEMENT data (#PCDATA)>

<!ELEMENT Stiffness (data*)>
  <!-- information sur la profondeur prise en compte pour le calcul d'une dureté -->
  <!ATTLIST Stiffness deep CDATA #REQUIRED>
  <!-- information sur le nombre de dureté par pixel prises (chacunes d'elles reflète
      la dureté sur une profondeur 'deep') -->
  <!ATTLIST Stiffness number CDATA #REQUIRED>
  <!-- information sur la valeur de contraste lors du dernier affichage de la fenêtre
      de dureté -->
  <!ATTLIST Stiffness contraste CDATA #IMPLIED>
  <!-- Information numérique sur la pente d'une courbe prise sur une surface dure -->
  <!ATTLIST Stiffness glass CDATA #IMPLIED>


  <!-- contient les données, autant de fois que la valeur number -->
  <!ELEMENT data (#PCDATA)>
<!ELEMENT ForceCurves (CurveYAv*,CurveX)>
<!ELEMENT CurveYAv (#PCDATA)>
<!ATTLIST CurveYAv curveID ID #REQUIRED>
<!ELEMENT CurveYRe (#PCDATA)>
<!
<!-- Une seule courbe X est nécessaire puisqu'elle est identique pour toutes les
    courbes-->
<!ELEMENT CurveX (#PCDATA)>
```

Bibliography

Abrami, L., Velluz, M. C., Hong, Y., Ohishi, K., Mehlert, A., Ferguson, M., Kinoshita, T., and Gisou van der Goot, F. (2002). The glycan core of GPI-anchored proteins modulates aerolysin binding but is not sufficient: the polypeptide moiety is required for the toxin-receptor interaction. *FEBS Lett*, 512(1-3):249–54.

Allen, S., Chen, X., Davies, J., Davies, M., Dawkes, A., Edwards, J., Roberts, C., Sefton, J., Tendler, S., and Williams, P. (1997). Detection of antigen-antibody binding events with the atomic force microscope. *Biochemistry*, 36(24):7457–63.

Allen, S., Davies, J., Davies, M., Dawkes, A., Roberts, C., Tendler, S., and Williams, P. (1999). The influence of epitope availability on atomic-force microscope studies of antigen-antibody interactions. *Biochem J*, 341 (Pt 1):173–8.

Anderson, R. G. and Jacobson, K. (2002). A role for lipid shells in targeting proteins to caveolae, rafts, and other lipid domains. *Science*, 296(5574):1821–5.

Arnoldi, M., Fritz, M., Bauerlein, E., Radmacher, M., Sackmann, E., and Boulbitch, A. (2000). Bacterial turgor pressure can be measured by atomic force microscopy. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, 62(1 Pt B):1034–44.

Ashkin, A. and Dziedzic, J. M. (1987). Optical trapping and manipulation of viruses and bacteria. *Science*, 235(4795):1517–20.

Barman, S. and Nayak, D. P. (2000). Analysis of the transmembrane domain of influenza virus neuraminidase, a type II transmembrane glycoprotein, for apical sorting and raft association. *J Virol*, 74(14):6538–45.

Bavari, S., Bosio, C. M., Wiegand, E., Ruthel, G., Will, A. B., Geisbert, T. W., Hevey, M., Schmaljohn, C., Schmaljohn, A., and Aman, M. J. (2002). Lipid raft microdomains: a gateway for compartmentalized trafficking of Ebola and Marburg viruses. *J Exp Med*, 195(5):593–602.

Binnig, G., Quate, C., and Gerber, C. (1986). Atomic force microscope. *Physical review letters*, 56(9):930–933.

Binnig, G., Rohrer, H., Gerber, C., and Weibel, E. (1982). Tunneling through a controllable vacuum gap. *Applied Physics Letters*, 40(2):178–180.

Brandao, M. M., Fontes, A., Barjas-Castro, M. L., Barbosa, L. C., Costa, F. F., Cesar, C. L., and Saad, S. T. (2003). Optical tweezers for measuring red blood cell elasticity: application to the study of drug response in sickle cell disease. *Eur J Haematol*, 70(4):207–11.

Bretscher, M. S. (1973). Membrane structure: some general principles. *Science*, 181(100):622–9.

Brown, D. (1993). The tyrosine kinase connection: how GPI-anchored proteins activate T cells. *Curr Opin Immunol*, 5(3):349–54.

Brown, D. A. and London, E. (1998a). Functions of lipid rafts in biological membranes. *Annual Review of Cell and Developmental Biology*, 14:111–136.

Brown, D. A. and London, E. (1998b). Structure and origin of ordered lipid domains in biological membranes. *J Membr Biol*, 164(2):103–14.

Brugger, B., Graham, C., Leibrecht, I., Mombelli, E., Jen, A., Wieland, F., and Morris, R. (2004). The membrane domains occupied by glycosylphosphatidylinositol - anchored prion protein and Thy-1 differ in lipid composition. *J Biol Chem*, 279(9):7530–6.

Burnham, N. A. and Colton, R. J. (1989). Measuring the nanomechanical properties and surface forces of materials using an atomic force microscope. *Journal of Vacuum Science and Technology A: Vacuum, Surfaces, and Films*, 7(4):2906–2913.

Butt, H. J. and Jaschke, M. (1995). Calculation of thermal noise in atomic force microscopy. *Nanotechnology*, 6(1):1–7.

Casey, P. J. (1995). Protein lipidation in cell signaling. *Science*, 268(5208):221–5.

Dickenson, R. P., Hutton, W. C., and Stott, J. R. (1981). The mechanical properties of bone in osteoporosis. *J Bone Joint Surg Br*, 63-B(2):233–8.

Dietrich, C., Yang, B., Fujiwara, T., Kusumi, A., and Jacobson, K. (2002). Relationship of lipid rafts to transient confinement zones detected by single particle tracking. *Biophys J*, 82(1 Pt 1):274–84.

Digital Instrument (2000). *Scanning Probe Microscopy, Training Notebook.*

Domke, J. and Radmacher, M. (1998). Measuring the elastic properties of thin polymer films with the atomic force microscope. *Langmuir*, 14(12):3320–3325.

Douglass, A. and Vale, R. (2005). Single-molecule microscopy reveals plasma membrane microdomains created by protein-protein networks that exclude or trap signaling molecules in T cells. *Cell*, 121(6):937–50.

Drab, M., Verkade, P., Elger, M., Kasper, M., Lohn, M., Lauterbach, B., Menne, J., Lindschau, C., Mende, F., Luft, F. C., Schedl, A., Haller, H., and Kurzchalia, T. V. (2001). Loss of caveolae, vascular dysfunction, and pulmonary defects in caveolin-1 gene-disrupted mice. *Science*, 293(5539):2449–52.

Drake, B., Prater, C., Weisenhorn, A., Gould, S., Albrecht, T., Quate, C., Cannell, D., Hansma, H., and Hansma, P. (1989). Imaging crystals, polymers, and processes in water with the atomic force microscope. *Science*, 243(4898):1586–9.

Drygin, Y. F., Bordunova, O. A., Gallyamov, M. O., and Yaminsky, I. V. (1998). Atomic force microscopy examination of tobacco mosaic virus and virion RNA. *FEBS Lett*, 425(2):217–21.

Dufrene, Y. F. (2000). Direct characterization of the physicochemical properties of fungal spores using functionalized AFM probes. *Biophys J*, 78(6):3286–91.

Dulinska, I., Targosz, M., Strojny, W., Lekka, M., Czuba, P., Balwierz, W., and Szymonski, M. (2006). Stiffness of normal and pathological erythrocytes studied by means of atomic force microscopy. *J Biochem Biophys Methods*, 66(1-3):1–11.

Ehehalt, R., Keller, P., Haass, C., Thiele, C., and Simons, K. (2003). Amyloidogenic processing of the Alzheimer beta-amyloid precursor protein depends on lipid rafts. *J Cell Biol*, 160(1):113–23.

Ferguson-Pell, M., Hagisawa, S., and Masiello, R. D. (1994). A skin indentation system using a pneumatic bellows. *J Rehabil Res Dev*, 31(1):15–9.

Field, K. A., Holowka, D., and Baird, B. (1995). Fc epsilon RI-mediated recruitment of p53/56lyn to detergent-resistant membrane domains accompanies cellular signaling. *Proc Natl Acad Sci U S A*, 92(20):9201–5.

Fivaz, M., Vilbois, F., Thurnheer, S., Pasquali, C., Abrami, L., Bickel, P. E., Parton, R. G., and van der Goot, F. G. (2002). Differential sorting and fate of endocytosed GPI-anchored proteins. *EMBO J*, 21(15):3989–4000.

Freigang, J., Proba, K., Leder, L., Diederichs, K., Sonderegger, P., and Welte, W. (2000). The crystal structure of the ligand binding module of axonin-1/TAG-1 suggests a zipper mechanism for neural cell adhesion. *Cell*, 101(4):425–33.

Galbiati, F., Engelman, J. A., Volonte, D., Zhang, X. L., Minetti, C., Li, M., Hou, Jr, H., Kneitz, B., Edelmann, W., and Lisanti, M. P. (2001). Caveolin-3 null mice show a loss of caveolae, changes in the microdomain distribution of the dystrophin-glycoprotein complex, and t-tubule abnormalities. *J Biol Chem*, 276(24):21425–33.

Glogauer, M. and Ferrier, J. (1998). A new method for application of force to cells via ferric oxide beads. *Pflugers Arch*, 435(2):320–7.

Gould, S., Marti, O., Drake, B., Hellemans, L., Bracker, C. E., Hansma, P. K., Keder, N. L., Eddy, M. M., and Stucky, G. D. (1988). Molecular resolution images of amino acid crystals with the atomic force microscope. *Nature*, 332:332–334.

Gould, S. A. C., Drake, B., Prater, C. B., Weisenhorn, A. L., Manne, S., Kelderman, G. L., Butt, H. J., Hansma, H., Hansma, P. K., Magonov, S., and Cantow, H. J. (1990). The atomic force microscope: a tool for science and industry. *Ultramicroscopy*, 33:93–98.

Grafstrom, S., Ackermann, J., Hagen, T., Neumann, R., and Probst, O. (1994). Analysis of lateral force effects on the topography in scanning force microscopy. *The 1993 international conference on scanning tunneling microscopy*, 12(3):1559–1564.

Guirland, C., Suzuki, S., Kojima, M., Lu, B., and Zheng, J. (2004). Lipid rafts mediate chemotropic guidance of nerve growth cones. *Neuron*, 42(1):51–62.

Hansma, H. G., Sinsheimer, R. L., Groppe, J., Bruice, T. C., Elings, V., Gurley, G., Bezanilla, M., Mastrangelo, I. A., Hough, P. V., and Hansma, P. K. (1993). Recent advances in atomic force microscopy of DNA. *Scanning*, 15(5):296–9.

Hansma, P., Elings, V., Marti, O., and Bracker, C. (1988). Scanning tunneling microscopy and atomic force microscopy: application to biology and technology. *Science*, 242(4876):209–16.

Hansma, P. K., Cleveland, J. P., Radmacher, M., Walters, D. A., Hillner, P. E., Bezanilla, M., Fritz, M., Vie, D., Hansma, H. G., Prater, C. B., Massie, J., Fukunaga, L., Gurley, J., and Elings, V. (1994). Tapping mode atomic force microscopy in liquids. *Applied Physics Letters*, 64(13):1738–1740.

Harder, T., Scheiffele, P., Verkade, P., and Simons, K. (1998). Lipid domain structure of the plasma membrane revealed by patching of membrane components. *J Cell Biol*, 141(4):929–42.

Heisenberg, W. (1958). *Physics and Philosophy: The Revolution in Modern Science*. Harper and Brother Publishers.

Herreros, J., Ng, T., and Schiavo, G. (2001). Lipid rafts act as specialized domains for tetanus toxin binding and internalization into neurons. *Mol Biol Cell*, 12(10):2947–60.

Hertz, H. (1882). Über die Berührung fester elastischer Körper. *Journal für die reine und angewandte Mathematik*, (92):156 – 171.

Hochmuth, R. M. (2000). Micropipette aspiration of living cells. *J Biomech*, 33(1):15–22.

Kane, R. L., McMahon, T. A., Wagner, R. L., and Abelmann, W. H. (1976). Ventricular elastic modulus as a function of age in the Syrian golden hamster. *Circ Res*, 38(2):74–80.

Kasas, S., Gotzos, V., and Celio, M. (1993). Observation of living cells using the atomic force microscope. *Biophys J*, 64(2):539–44.

Kasas, S., Riederer, B., Catsicas, S., Cappella, B., and Dietler, G. (2000a). Fuzzy logic algorithm to extract specific interaction forces from atomic force microscopy data. *Review of Scientific Instruments*, 71(5):2082 – 2086.

Kasas, S., Thomson, N. H., Smith, B. L., Hansma, H. G., Zhu, X., Guthold, M., Bustamante, C., Kool, E. T., Kashlev, M., and Hansma, P. K. (1997). Escherichia coli RNA polymerase activity observed using atomic force microscopy. *Biochemistry*, 36(3):461–8.

Kasas, S., Wang, X., Hirling, H., Catsicas, S., Haeberli, C., Dietler, G., and Thomson, N. (2000b). Setup for observing living cells using a commercial atomic force microscope. *Review of Scientific Instruments*, 71(11):4338–4340.

Kasas, S., Wang, X., Hirling, H., Marsault, R., Huni, B., Yersin, A., Regazzi, R., Grenningloh, G., Riederer, B., Forro, L., Dietler, G., and Catsicas, S. (2005). Superficial and deep changes of cellular mechanical properties following cytoskeleton disassembly. *Cell Motil Cytoskeleton*, 62(2):124–32.

Konofagou, E. E., D'hooge, J., and Ophir, J. (2002). Myocardial elastography–a feasibility study in vivo. *Ultrasound Med Biol*, 28(4):475–82.

Kucerka, N., Pencer, J., Nieh, M. P., and Katsaras, J. (2007). Influence of cholesterol on the bilayer properties of monounsaturated phosphatidylcholine unilamellar vesicles. *Eur Phys J E Soft Matter*, 23(3):247–54.

Kusumi, A., Koyama-Honda, I., and Suzuki, K. (2004). Molecular dynamics and interactions for creation of stimulation-induced stabilized rafts from small unstable steady-state rafts. *Traffic*, 5(4):213–30.

Lafont, F., Abrami, L., and van der Goot, F. G. (2004). Bacterial subversion of lipid rafts. *Curr Opin Microbiol*, 7(1):4–10.

Lalli, G., Bohnert, S., Deinhardt, K., Verastegui, C., and Schiavo, G. (2003). The journey of tetanus and botulinum neurotoxins in neurons. *Trends Microbiol*, 11(9):431–7.

Lam, W. A., Rosenbluth, M. J., and Fletcher, D. A. (2007). Chemotherapy exposure increases leukemia cell stiffness. *Blood*, 109(8):3505–8.

Laney, D. E., Garcia, R. A., Parsons, S. M., and Hansma, H. G. (1997). Changes in the elastic properties of cholinergic synaptic vesicles as measured by atomic force microscopy. *Biophys J*, 72(2 Pt 1):806–13.

Ledesma, M. D., Brugger, B., Bunning, C., Wieland, F. T., and Dotti, C. G. (1999). Maturation of the axonal plasma membrane requires upregulation of sphingomyelin synthesis and formation of protein-lipid complexes. *EMBO J*, 18(7):1761–71.

Ledesma, M. D., Simons, K., and Dotti, C. G. (1998). Neuronal polarity: essential role of protein-lipid complexes in axonal sorting. *Proc Natl Acad Sci U S A*, 95(7):3966–71.

Lekka, M., Laidler, P., Gil, D., Lekki, J., Stachura, Z., and Hrynkiewicz, A. Z. (1999). Elasticity of normal and cancerous human bladder cells studied by scanning force microscopy. *Eur Biophys J*, 28(4):312–6.

Li, X. M., Momsen, M. M., Smaby, J. M., Brockman, H. L., and Brown, R. E. (2001). Cholesterol decreases the interfacial elasticity and detergent solubility of sphingomyelins. *Biochemistry*, 40(20):5954–63.

Lyyra, T., Kiviranta, I., Vaatainen, U., Helminen, H. J., and Jurvelin, J. S. (1999). In vivo characterization of indentation stiffness of articular cartilage in the normal human knee. *J Biomed Mater Res*, 48(4):482–7.

Manduca, A., Oliphant, T. E., Dresner, M. A., Mahowald, J. L., Kruse, S. A., Amromin, E., Felmlee, J. P., Greenleaf, J. F., and Ehman, R. L. (2001). Magnetic resonance elastography: non-invasive mapping of tissue elasticity. *Med Image Anal*, 5(4):237–54.

Marti, O., Ribi, H. O., Drake, B., Albrecht, T. R., Quate, C. F., and Hansma, P. K. (1988). Atomic force microscopy of an organic monolayer. *Science*, 239(4835):50–2.

Martin, Y., Williams, C. C., and Wickramasinghe, H. K. (1987). Atomic force microscope–force mapping and profiling on a sub 100-[a-ring] scale. *Journal of Applied Physics*, 61(10):4723–4729.

Mate, C. M., McClelland, G. M., Erlandsson, R., and Chiang, S. (1987). Atomic-scale friction of a tungsten tip on a graphite surface. *Physical Review Letters*, 59(17):1942–1945.

Mayor, S. and Rao, M. (2004). Rafts: scale-dependent, active lipid organization at the cell surface. *Traffic*, 5(4):231–40.

Meyer, G. and Amer, N. M. (1988). Novel optical approach to atomic force microscopy. *Applied Physics Letters*, 53(12):1045–1047.

Mills, J. P., Qie, L., Dao, M., Lim, C. T., and Suresh, S. (2004). Nonlinear elastic and viscoelastic deformation of the human red blood cell with optical tweezers. *Mech Chem Biosyst*, 1(3):169–80.

Morgenthaler, F., Knott, G., Floyd Sarria, J., Wang, X., Staple, J., Catsicas, S., and Hirling, H. (2003). Morphological and molecular heterogeneity in release sites of single neurons. *Eur J Neurosci*, 17(7):1365–74.

Morone, N., Fujiwara, T., Murase, K., Kasai, R. S., Ike, H., Yuasa, S., Usukura, J., and Kusumi, A. (2006). Three-dimensional reconstruction of the membrane skeleton at the plasma membrane interface by electron tomography. *J Cell Biol*, 174(6):851–62.

Mou, J., Sheng, S., Ho, R., and Shao, Z. (1996). Chaperonins GroEL and GroES: views from atomic force microscopy. *Biophys J*, 71(4):2213–21.

Muller, P. and Herrmann, A. (2002). Rapid Transbilayer Movement of Spin-Labeled Steroids in Human Erythrocytes and in Liposomes. *Biophys. J.*, 82(3):1418–1428.

Nguyen, D. H. and Hildreth, J. E. (2000). Evidence for budding of human immunodeficiency virus type 1 selectively from glycolipid-enriched membrane lipid rafts. *J Virol*, 74(7):3264–72.

Niethammer, P., Delling, M., Sytnyk, V., Dityatev, A., Fukami, K., and Schachner, M. (2002). Cosignaling of NCAM via lipid rafts and the FGF receptor is required for neuritogenesis. *J Cell Biol*, 157(3):521–32.

Nohe, A., Keating, E., Fivaz, M., van der Goot, F. G., and Petersen, N. O. (2006). Dynamics of GPI-anchored proteins on the surface of living cells. *Nanomedicine*, 2(1):1–7.

Ophir, J., Alam, S. K., Garra, B., Kallel, F., Konofagou, E., Krouskop, T., and Varghese, T. (1999). Elastography: ultrasonic estimation and imaging of the elastic properties of tissues. *Proc Inst Mech Eng [H]*, 213(3):203–33.

Owicki, J. C. and McConnell, H. M. (1980). Lateral diffusion in inhomogeneous membranes. Model membranes containing cholesterol. *Biophys J*, 30(3):383–97.

Putman, C. A., van der Werf, K. O., de Grooth, B. G., van Hulst, N. F., Greve, J., and Hansma, P. K. (1992). New imaging mode in atomic-force microscopy based on the error signal. *Scanning Probe Microscopies*, 1639(1):198–204.

Putman, C. A. J., der Werf, K. O. V., Grooth, B. G. D., Hulst, N. F. V., and Greve, J. (1994). Tapping mode atomic force microscopy in liquid. *Applied Physics Letters*, 64(18):2454–2456.

Radmacher, M. (1997). Measuring the elastic properties of biological samples with the AFM. *IEEE Eng Med Biol Mag*, 16(2):47–57.

Radmacher, M. (2002). Measuring the elastic properties of living cells by the atomic force microscope. *Methods Cell Biol*, 68:67–90.

Radmacher, M., Fritz, M., Kacher, C. M., Cleveland, J. P., and Hansma, P. K. (1996). Measuring the viscoelastic properties of human platelets with the atomic force microscope. *Biophys J*, 70(1):556–67.

Ramstedt, B. and Slotte, J. P. (2002). Membrane properties of sphingomyelins. *FEBS Lett*, 531(1):33–7.

Ren, X., Ostermeyer, A. G., Ramcharan, L. T., Zeng, Y., Lublin, D. M., and Brown, D. A. (2004). Conformational defects slow Golgi exit, block oligomerization, and reduce raft affinity of caveolin-1 mutant proteins. *Mol Biol Cell*, 15(10):4556–67.

Rotsch, C. and Radmacher, M. (2000). Drug-induced changes of cytoskeletal structure and mechanics in fibroblasts: an atomic force microscopy study. *Biophys J*, 78(1):520–35.

Sabharanjak, S., Sharma, P., Parton, R. G., and Mayor, S. (2002). GPI-anchored proteins are delivered to recycling endosomes via a distinct cdc42-regulated, clathrin-independent pinocytic pathway. *Dev Cell*, 2(4):411–23.

Samori, B., Siligardi, G., Quagliariello, C., Weisenhorn, A. L., Vesenka, J., and Bustamante, C. J. (1993). Chirality of DNA supercoiling assigned by scanning force microscopy. *Proc Natl Acad Sci U S A*, 90(8):3598–601.

Scheiffele, P. (2003). Cell-cell signaling during synapse formation in the CNS. *Annu Rev Neurosci*, 26:485–508.

Scheiffele, P., Roth, M. G., and Simons, K. (1997). Interaction of influenza virus haemagglutinin with sphingolipid-cholesterol membrane domains via its transmembrane domain. *EMBO J*, 16(18):5501–8.

Sharma, P., Varma, R., Sarasij, R. C., Ira, Gousset, K., Krishnamoorthy, G., Rao, M., and Mayor, S. (2004). Nanoscale organization of multiple GPI-anchored proteins in living cell membranes. *Cell*, 116(4):577–89.

Sheets, E. D., Holowka, D., and Baird, B. (1999a). Critical role for cholesterol in Lyn-mediated tyrosine phosphorylation of FcepsilonRI and their association with detergent-resistant membranes. *J Cell Biol*, 145(4):877–87.

Sheets, E. D., Holowka, D., and Baird, B. (1999b). Membrane organization in immunoglobulin E receptor signaling. *Curr Opin Chem Biol*, 3(1):95–9.

Sheets, E. D., Lee, G. M., Simson, R., and Jacobson, K. (1997). Transient confinement of a glycosylphosphatidylinositol-anchored protein in the plasma membrane. *Biochemistry*, 36(41):12449–58.

Shogomori, H. and Futerman, A. H. (2001). Cholesterol depletion by methyl-beta-cyclodextrin blocks cholera toxin transport from endosomes to the Golgi apparatus in hippocampal neurons. *J Neurochem*, 78(5):991–9.

Silver, F. H., Bradica, G., and Tria, A. (2001). Viscoelastic behavior of osteoarthritic cartilage. *Connect Tissue Res*, 42(3):223–33.

Simons, K. and Ikonen, E. (1997). Functional rafts in cell membranes. *Nature*, 387(6633):569–72.

Simons, K. and Toomre, D. (2000). Lipid rafts and signal transduction. *Nat Rev Mol Cell Biol*, 1(1):31–9.

Simons, K. and van Meer, G. (1988). Lipid sorting in epithelial cells. *Biochemistry*, 27(17):6197–202.

Simons, K. and Vaz, W. L. (2004). Model systems, lipid rafts, and cell membranes. *Annu Rev Biophys Biomol Struct*, 33:269–95.

Simson, R., Sheets, E. D., and Jacobson, K. (1995). Detection of temporary lateral confinement of membrane proteins using single-particle tracking analysis. *Biophys J*, 69(3):989–93.

Singer, S. J. and Nicolson, G. L. (1972). The fluid mosaic model of the structure of cell membranes. *Science*, 175(23):720–31.

Skibbens, J. E., Roth, M. G., and Matlin, K. S. (1989). Differential extractability of influenza virus hemagglutinin during intracellular transport in polarized epithelial cells and nonpolar fibroblasts. *J Cell Biol*, 108(3):821–32.

Smith, L. M., Rubenstein, J. L., Parce, J. W., and McConnell, H. M. (1980). Lateral diffusion of M-13 coat protein in mixtures of phosphatidylcholine and cholesterol. *Biochemistry*, 19(25):5907–11.

Smith, S. B., Finzi, L., and Bustamante, C. (1992). Direct mechanical measurements of the elasticity of single DNA molecules by using magnetic beads. *Science*, 258(5085):1122–6.

Sneddon, I. N. (1965). The relation between load and penetration in the axisymmetric boussinesq problem for a punch of arbitrary profile. *International Journal of Engineering Science*, 3:47–57.

Sooksawate, T. and Simmonds, M. A. (2001). Effects of membrane cholesterol on the sensitivity of the GABA(A) receptor to GABA in acutely dissociated rat hippocampal neurones. *Neuropharmacology*, 40(2):178–84.

Steck, T. L., Ye, J., and Lange, Y. (2002). Probing red cell membrane cholesterol movement with cyclodextrin. *Biophys J*, 83(4):2118–25.

Suresh, S. (2007). Biomechanics and biophysics of cancer cells. *Acta Biomater*, 3(4):413–38.

Suresh, S., Spatz, J., Mills, J. P., Micoulet, A., Dao, M., Lim, C. T., Beil, M., and Seufferlein, T. (2005). Connections between single-cell biomechanics and human disease states: gastrointestinal cancer and malaria. *Acta Biomater*, 1(1):15–30.

Tansey, M. G., Baloh, R. H., Milbrandt, J., and Johnson, Jr, E. M. (2000). GFRalpha-mediated localization of RET to lipid rafts is required for effective downstream signaling, differentiation, and neuronal survival. *Neuron*, 25(3):611–23.

Tao, N. J., Lindsay, S. M., and Lees, S. (1992). Measuring the microelastic properties of biological-material. *Biophysical journal*, 63(4):1165 – 1169.

Taraboulos, A., Scott, M., Semenov, A., Avrahami, D., Laszlo, L., and Prusiner, S. B. (1995). Cholesterol depletion and modification of COOH-terminal targeting sequence of the prion protein inhibit formation of the scrapie isoform. *J Cell Biol*, 129(1):121–32.

Tokumasu, F., Jin, A. J., Feigenson, G. W., and Dvorak, J. A. (2003). Nanoscopic lipid domain dynamics revealed by atomic force microscopy. *Biophys J*, 84(4):2609–18.

Touhami, A., Nysten, B., and Dufrene, Y. (2003). Nanoscale mapping of the elasticity of microbial cells by atomic force microscopy. *Langmuir*, 19(11):4539–4543.

van der Merwe, P. A. and Barclay, A. N. (1994). Transient intercellular adhesion: the importance of weak protein-protein interactions. *Trends Biochem Sci*, 19(9):354–8.

Vie, V., Van Mau, N., Lesniewska, E., Goudonnet, J., Heitz, F., and Le Grimellec, C. (1998). Distribution of ganglioside gm1 between two-component, two-phase phosphatidylcholine monolayers. *Langmuir*, 14(16):4574–4583.

Wang, N. and Ingber, D. E. (1995). Probing transmembrane mechanical coupling and cytomechanics using magnetic twisting cytometry. *Biochem Cell Biol*, 73(7-8):327–35.

Weihs, T. P., Nawaz, Z., Jarvis, S. P., and Pethica, J. B. (1991). Limits of imaging resolution for atomic force microscopy of molecules. *Applied Physics Letters*, 59(27):3536–3538.

Weisenhorn, A. L., Hansma, P. K., Albrecht, T. R., and Quate, C. F. (1989). Forces in atomic force microscopy in air and water. *Applied Physics Letters*, 54(26):2651–2653.

Weisenhorn, A. L., Khorsandi, M., Kasas, S., Gotzos, V., and Butt, H. J. (1993). Deformation and height anomaly of soft surfaces studied with an afm. *Nanotechnology*, 4(2):106–113.

Williams, M. (2002). Optical Tweezers: Measuring Piconewton Forces. *Biophysical Textbook Online.*

Yeagle, P. L. (1985). Cholesterol and the cell membrane. *Biochimica et Biophysica Acta (BBA) - Reviews on Biomembranes*, 822(3-4):267–87.

Yersin, A., Hirling, H., Steiner, P., Magnin, S., Regazzi, R., Huni, B., Huguenot, P., De los Rios, P., Dietler, G., Catsicas, S., and Kasas, S. (2003). Interactions between synaptic vesicle fusion proteins explored by atomic force microscopy. *Proc Natl Acad Sci U S A*, 100(15):8736–41.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zenhausern, F., Adrian, M., Emch, R., Taborelli, M., Jobin, M., and Descouts, P. (1992). Scanning force microscopy and cryo-electron microscopy of tobacco mosaic virus as a test specimen. *Ultramicroscopy*, 42-44:1168–1172.

Charles Roduit
ch. des retraites 4
CH-1004 Lausanne

Tel : +41(0)21 550 0456
Mobile : +41(0)76 234 7677
e-mail : charles.roduit@a3.epfl.ch

Driver license Cat B
Origin : CH (Leytron VS)
Born the 24.09.1976

Charles Roduit

Education and Scientific Formation

2003-2007	**Phd in Neurosciences** Science de la Vie, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
1997-2002	**Master degree in Biology** University of Lausanne, Switzerland.
1995-1997	**Physics** EPFL, Switzerland.
1995	**Swiss General Certificate of Education** section Economy, Switzerland

Language

French	Mother tongue
English	Fluent
German	Basic

Personal Interest

Traveling	Switzerland, France, Italy, Turkey, USA, Canada
Sport	Snowboard, Ski, Hiking, Cycling, Horseback
Divers	Reading, Astronomy, Computing, Photography, ...

Other Activities

2007	Sales + Support of the AFM Data Processing Software, which was developed during my thesis, Veeco (Digital Instrument)
2000	Webmaster, under contract with Swissmatbéton, Lausanne (VD), Switzerland
1998 - 1999	Startup creation, Computer advise and sales, Lausanne (VD), Switzerland
1997 - 1998	Snowboard teacher, Swiss Ski Scool, Ovronnaz (VS), Switzerland

Computer Skills

Operating system	Linux, Mac OSX, MS-Windows
Office	OpenOffice.org, Gnumeric, MS-Office, Latex
Graphics	The Gimp, Inscape, Blender, Scribus
Programming	Python, Matlab, bash, R

Teaching Experences

2005 - 2007	Organizing and coaching practical works AFM 3 days , 3rd year student, Biology (Unil) , "Study of neuronal membrane elastical properties*, drugs effects on rafts.*"
2004 - 2005	Coaching TP "*Biologie cellulaire I*", 1rst year student, Life Science (EPFL)
2004 - 2005	Teaching (2x1h) + Practical Work "*Introduction to AFM*", 1rst year student, Life Science (EPFL)

Postgrade Formation

2007	*Biophotonics*, Doctoral course, EPFL
2007	*Biomedical approach for drug evaluation*, Doctoral course, EPFL
2007	*Python advanced*, IT domain course, EPFL
2007	*Python basic*, IT domain course, EPFL
2007	*ADN Biophysic*, Doctoral course, EPFL
2005	Introduction to neuroscience, Doctoral course, EPFL

Research Activities

2003 – 2007	**PhD work** in Neurosciences, Laboratoire de Neurobiologie Cellulaire (LNC), Science de la Vie, Ecole Polytechnique Fédérale de Lausanne, Switzerland. *Membrane elastic heterogeneity studied at nanometrical scale on living cells.* Head : Dr. S. Kasas and Pr. S. Catsicas.
2002	**Master work** biology, University of Lausanne, Switzerland. Subject : *Effect of a putative pheromone on the Bacillus subtilis W23 tarA transcription: analyse of the function of the factor sigma ECF (Extra Cytoplasmic Function),* σ^X; Head : Dr. C. Mauël.
2002	**Certificat** of Experimental Microbiology, Université de Lausanne. Subject: *Study of the gene product yqgA of Bacillus subtilis.* Head : Dr. P. Margot.
2001	**Certificat** of Gene Signaling and Regulation, University of Lausanne. Subject: *Expression of the defense gene during interaction between insects and Arabidopsis thaliana;* Head : Dr. P. Reymond, Pr. EE. Farmer.
2005	*Introduction to cellular neurobiology*, Doctoral course, EPFL

International Conferences

01/2007	C. Roduit, S. Kasas, F. Lafont et S. Catsicas. "Relative Stiffness of Rafts Microdomains revealed by Atomic Force Microscopy." (Posters) *Annual Linz Winter Workshop . Advances in Single-Molecule Research for Biology and* Nanoscience. (A)
01/2007	C. Roduit, A. Yersin, F. Lafont, G. Dietler, S. Catsicas et S. Kasas. "Stiffness Tomography by Atomic Force Microscopy" (Posters) *Annual Linz Winter Workshop . Advances in Single-Molecule Research for Biology and* Nanoscience. (A)
09/2006	C. Roduit, A. Yersin, F. Lafont, G. Dietler, S. Catsicas et S. Kasas. "Stiffness Tomography by Atomic Force Microscopy" (Poster) *Neuroscience meeting, Diablerets* (CH) .
09/2005	C. Roduit, S. Kasas, F. Lafont et S. Catsicas. "Relative Stiffness of Rafts Microdomains revealed by Atomic Force Microscopy." (Poster) N*euroscience meeting,* Diablerets (CH).
12/2003	C. Roduit, A. Yersin, G. Dietler, F. Lafont, S. Catsicas et S. Kasas. "New Tool for Exploring Cell Surface Properties Using Atomic Force Microscopy" (Poster) *Veeco User meeting,* EPFL (CH)

Publications

2006	Yersin, A., Hirling, H., Kasas, S., Roduit, C., Kulangara, K., Dietler, G., Lafont, F., Catsicas, S. and Steiner, P. (2006) *Elastic properties of the cell surface and trafficking of single AMPA receptors in living hippocampal neurons* Biophys J, 92(12):4482-4489.
2007	Roduit, C., van der Goot, F. G., De Los Rios, P., Yersin, A., Steiner, P., Dietler, G., Catsicas, S., Lafont, F., and Kasas, S. (2007) *Elastic membrane heterogeneity of living cells revealed by stiff nanoscale membrane domains.* Biophys J, 94(4):-------.

Patent and Licenses

2007	Stiffness tomography by atomic force microscopy, S. Kasass, C. Roduit, F. Lafont, S. Catsicas. Date International Application Number : PCT/US07/64727
2007	Sale of an exclusive license of the Force-Volume and Tomography Software to Veeco.