Niklas Valjakka

Ohutkalvojen tunnistaminen atomivoimamikroskoopilla määritetyn kimmokertoimen avulla

Fysiikan pro gradu -tutkielma

5. kesäkuuta 2023

Jyväskylän yliopisto

Matemaattis-luonnontieteellinen tiedekunta

Tekijä: Niklas Valjakka

Yhteystiedot: nipevalj@jyu.fi

Ohjaajat: Kai Arstila ja Timo Sajavaara

Työn nimi: Ohutkalvojen tunnistaminen atomivoimamikroskoopilla määritetyn kimmokertoimen avulla

Title in English: Identification of thin film layers using Atomic Force Microscope -determined Young's Modulus

Työ: Pro gradu -tutkielma

Opintosuunta: Fysiikka

Sivumäärä: 49+47

Tiivistelmä: Tässä työssä tutkin aiemmin vähän tutkittua menetelmää tutkia monikerroksisia ohutkalvoja. Menetelmä perustuu AFM:n avulla näytteestä mitattuun voimakäyrätietoon. Mittaukset koostuivat näytteiden valmistamisesta, sekä HIM:llä ja AFM:llä tapahtuneista kuvauksista.

Työssäni tulen osoittamaan, että ohutkalvoista on mahdollista erottaa eri materiaalikerroksia menetelmän avulla. Menetelmässä on kuitenkin useita rajoituksia, jotka estävät menetelmän käyttöönoton käytännön sovelluksissa.

Avainsanat: Youngin moduuli, Atomivoimamikroskopia, Voimakäyrät, Heliumionimikroskopia

Abstract: In this work, I investigate a previously little studied method to study multilayer thin films. The method is based on force curve data measured from a sample using AFM. The measurements consisted of sample preparation, and imaging with HIM and AFM.

In my work, I will show that it is possible to distinguish different material layers in thin films using this method. However, the method has several limitations that prevent its use in practical applications.

Keywords: Young's Modulus, Atomic Force Microscopy, Force Curves, Helium Ion Microscopy

Lyhenne- ja termiluettelo

Kimmokerroin	Tunnetaan myös nimellä Youngin moduuli (englanniksi Young's Modulus). Kuvaa jännityksen ja venymän suhdetta.
ALD	Atomikerroskasvatus (englanniksi Atomic Layer Deposition).
HIM	Heliumionimikroskooppi (englanniksi Helium Ion Microscope).
FIB	Keskitetty ionisuihku (englanniksi Focused Ion Beam). Tapa työstää materiaaleja sputteroimalla käyttäen kiihdytettyjä ioneja keskitetysti.
Työstö	FIB:n avulla näytteeseen tehty vaurio.
AFM	Atomivoimamikroskooppi (englanniksi Atomic Force Microscope).
Voimakäyrä	Voima vs. etäisyys -kuvaaja, joka kuvaa AFM-kärjen ja näytteen keskinäistä vuorovaikutusta.
PF	Peak Force eli huippuvoima, joka toimii takaisinkytkentänä AFM:n Peak Force QNM -kuvantamismenetelmässä.
PFS	Suurin PF:n arvo. Käyttäjän asettama.
Kärki	AFM:ssä tipin osa, joka tunnustelee näytteen pintaa.

Sisältö

1 JOHDANTO 1

2 AINEEN MEKAANISET OMINAISUUDET 3

2.1 Kimmokertoimen määritelmä 3

2.2 Kimmokertoimen mittaus 3

3 ATOMIVOIMAMIKROSKOPIA JA VOIMAKÄYRÄT 6

3.1 Atomikerroskasvatus nanorakenteiden suojana 10

3.2 Aiempia atomivoimamikroskoopilla tehtyjä mittauksia 11

4 HELIUMIONIMIKROSKOOPPI 15

4.1 Heliumionimikroskopian historiaa 15

4.2 Heliumionimikroskoopin toimintaperiaate 17

5 TUTKITTAVAT MATERIAALIT JA MITTAUKSET 21

5.1 Näytteiden valmistus 21

5.2 Mittausten kulku 22

6 MITTAUSTULOSTEN ANALYSOINTI 27

7 POHDINTALUKU 35

8 YHTEENVETO 39

LÄHTEET 41

LIITTEET 45

A Tutkielmaani liittyvät ohjelmakoodit 45

1 Johdanto

Maailma on täynnä erilaisia mineraaleja ja materiaaleja. Miljoonia vuosia sitten ihmiskunnan yleisimmät työkalut valmistettiin kivestä, puusta ja luusta. Sivilisaation kehittyessä ja tiedon karttuessa työkaluihin käytetyt materiaalit ovat merkittävästi monipuolistuneet. Jotta työkaluja voitaisiin käyttää mahdollisimman tehokkaasti, ne tulee valmistaa materiaaleista, joiden ominaisuudet vastaavat parhaiten työkalun käyttötarkoitusta. Käytetyn materiaalin lisäksi myös kappaleen muodolla on väliä. Materiaalien tunnistaminen, näiden ominaisuuksien luotettava mittaaminen sekä muodon tarkastelu on yksi olennaisimmista materiaalitutkimuksen tavoitteista.

Materiaalitutkimukseen liittyvät karakterisontimenetelmät keskittyvät usein erilaisten näytteiden koostumusten ja rakenteiden selvittämiseen. Tutkimusalassa tarkasteltavien näytteiden koot voivat vaihdella metallisista esineistä mikroskooppisiin kappaleisiin. Samoin näytteestä voidaan tutkia jotain mekaanista, sähköistä tai termistä ominaisuutta. Yksi materiaalifysiikan tutkimuskohde on erilaiset monikerroksiset ohutkalvot, joissa yhden kerroksen paksuus vaihtelee yksittäisistä kymmeniin nanometreihin, sekä näiden käyttötarkoituksesta riippuvat olennaiset ominaisuudet.

Viime vuosikymmeninä teknologiakehityksessä ovat ohutkalvot nousseet merkittävään asemaan. Tämä on seurausta ohutkalvojen käytöstä muun muassa kappaleita suojaavina päällysteinä ja eristävinä kerroksina erilaisissa elektroniikkalaitteissa. Lisäksi ohutkalvoja käytetään aurinkopaneeleissa, joiden kehittäminen on tällä hetkellä ajankohtaista uusiutuvan energian ollessa ilmastonmuutoksen ja vihreän siirtymän takia merkittävässä valokeilassa. Ohutkalvojen yleistyessä tulee tarve tutkia ohuita kerroksia mahdollisimman tarkasti ja monipuolisesti sopivuuden määrittämiseksi eri sovelluksille. Tätä tarvetta on käsitelty aiheeseen liittyvässä kirjallisuudessa, jossa on perehdytty moniin ohutkalvojen analysointimenetelmiin [1].

Käyttäen tavanomaisia kuvantamismenetelmiä, kuten esimerkiksi pyyhkäisyelektronimikroskooppia (Scanning electron microscope, SEM), ohutkalvojen paksuuksia olisi mahdollista tarkastella kuvaamalla ohutkalvoa näytteen reunasta. Hyvin ohuiden kalvojen (paksuus korkeintaan muutamia kymmeniä nanometrejä) kanssa ongelmaksi muodostuvat kuitenkin

reunaefektit, jotka peittävät alleen paljon informaatiota. Hyvin ohuiden kalvojen tarkastelussa suora kuvantaminen ei siis ole aina mahdollista, vaan tarvitaan jokin toinen menetelmä.

Tässä tutkielmassa esittelen uuden tavan tunnistaa ohutkalvon materiaalikerroksia. Tutkimuksessa näytteen pintaan tehdään keskitetyn ionisuihkun (Focused Ion Beam, FIB) avulla lähes pinnan suuntainen työstö. Tämän jälkeen työstön pintaa tarkastellaan keräämällä tältä kohdalta atomivoimamikroskoopilla voimakäyrätietoa. Menetelmän tarkoitus on laajentaa tutkittavaa alaa työstön avulla ja tunnistaa materiaalikerroksia voimakäyristä saatavien ominaisuuksien avulla. Tämän avulla voitaisiin erottaa myös hyvin ohuita kerroksia, jotka muuten jäisivät reunaefektien alle.

Tutkimuksen tavoitteena on kehittää ohutkalvojen karakterisointimenetelmä, jonka avulla myös muutamien kymmenien nanometrien kerrokset olisivat erotettavissa. Kyseisen menetelmän soveltuvuutta ohutkalvojen karakterisointiin ei ole tutkittu aiemmin, joten tutkielmassani otan kantaa myös siihen, onko menetelmä aidosti hyödyllinen. Arvioin menetelmän hyödylliseksi, jos saadut tulokset ovat tarkkuudessaan verrattavia muihin analysointimenetelmiin.

Koska kyseessä on aiemmin vähän tutkittu tapa ylipäänsä tarkastella materiaaleja, keskityn tutkielmassani esittelemään kyseisen mittausmenetelmän, käyden läpi mittauksiin liittyvän teorian, menetelmät ja laitteiston. Tämän lisäksi arvioidaan saatuja tuloksia ja pyritään selvittämään, onko menetelmä käytännöllinen kuvaamaan materiaalivaihteluita ohutkalvoissa. Tutkimuksessani keskityn mittaamieni tulosten kvantitatiiviseen analyysiin, pyrkimällä laskemaan oikeisiin suureisiin perustuvia arvoja. Erinäisiä ongelmia liittyy kuitenkin tulosten vertailuun. Näihin haasteisiin palaan tarkemmin myöhemmin tuloksia käsiteltäessä.

Tutkimukseen kuului mittausten lisäksi mittausohjelmistolla kerätyn mittaustulosten tarkastelua. Analyysi tapahtui tätä tutkielmaa varten kehittelemälläni Python-ohjelmalla (LIITE 1). Ohjelmani tarkoitus on lukea binääridataa, piirtää voimakäyriä tästä datasta ja laskea voimakäyrästä saatavia ominaisuuksia. Arvioin ohjelmani luotettavuutta tutkielmassa vertaamalla ohjelmani piirtämiä voimakäyriä mittausohjelmiston tuottamiin kuvaajiin. Oman ohjelman käyttö antaa mahdollisuuden tutkia ominaisuuksia, jotka eivät näy atomivoimamikroskoopin valmistajan mittausohjelmassa.

2 Aineen mekaaniset ominaisuudet

Topografian lisäksi atomivoimamikroskoopilla voidaan mitata voimakäyriä. Tämän vuoksi on hyvä tarkentaa, millaisia ominaisuuksia voimakäyristä voidaan määrittää. Tässä luvussa esittelen tutkimukseni kannalta keskeisen suureen, materiaalin kimmokertoimen. Kyseinen suure on määritettävissä näytteestä otetusta voimakäyrästä ja on jokaiselle aineelle ominainen.

2.1 Kimmokertoimen määritelmä

Kimmokerroin kuvastaa kiinteän materiaalin kuormituksen suhdetta aiheutettuun venymään. Mitä suurempi kimmokerroin, sitä suurempi voima tarvitaan venyttämään tai puristamaan materiaalia tietyn verran. Kimmokerroin voidaan esittää muodossa

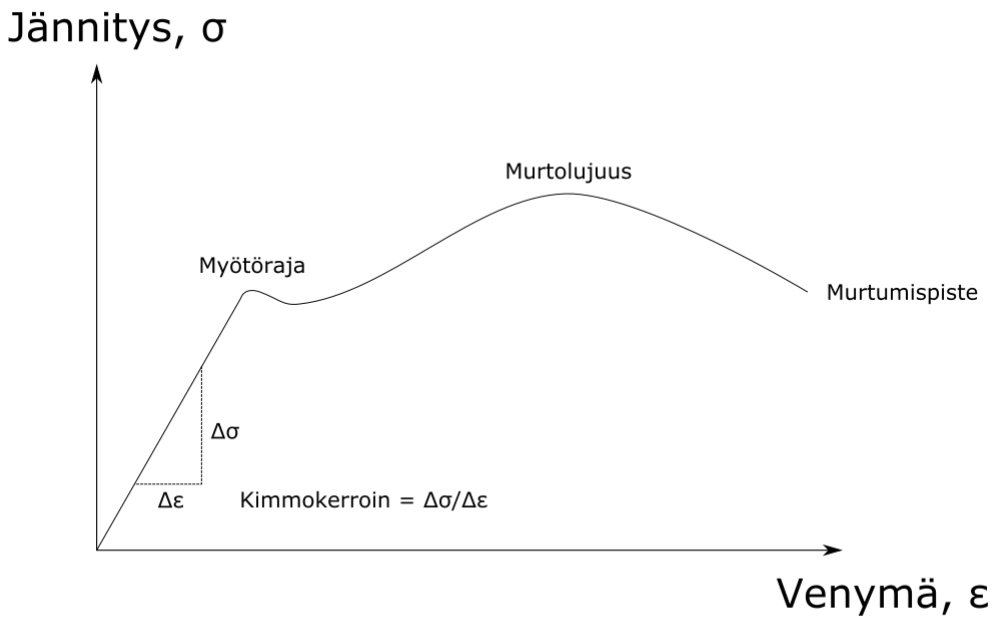
$$E = \frac{\sigma}{\varepsilon} = \frac{F/A}{\Delta L/L_0} = \frac{FL_0}{A\Delta L}, \tag{2.1}$$

missä E on kappaleen kimmokerroin, σ on kappaleeseen kohdistuva jännitys, ε on kappaleen venymä, F on kappaletta venyttävä voima, A on vaikutuspinta-ala, ΔL on kappaleen pituuden muutos ja L_0 on alkuperäinen pituus. Kimmokertoimen arvot ovat etenkin metalleilla erittäin suuria (useimmilla jopa > 10 GPa), joten yksikkönä on totuttu käyttämään gigapascalia (GPa) [2].

2.2 Kimmokertoimen mittaus

Kimmokertoimen mittaus tapahtuu tavallisesti kohdistamalla materiaaliin ensin tietty jännitys, jonka jälkeen mitataan tästä aiheutunut venymä. Tuloksena tulisi saada kuvan 1 mukainen jännitys–venymäkuvaaja. Kuvasta havaitaan, että pienemmillä jännityksen arvoilla kuvaaja on lineaarinen, kuten yhtälö (2.1) antaa olettaa. Kimmokerroin saadaan tämän lineaarisen osan kulmakertoimesta.

Tarkasteltaessa metalleille tyypillistä jännitys–venymäkuvaajaa, havaitaan että näiden suu-



Kuva 1. Taipuisille materiaaleille tyypillinen jännitys-venymäkuvaaja. Alemmilla venymän arvoilla suureiden suhde on lineaarinen yhtälön (2.1) mukaan.

reiden suhde on lineaarinen ainoastaan tiettyyn pisteeseen asti. Jos kappaleeseen kohdistuva jännitys ylittää kynnyksen, jota kutsutaan myötörajaksi, niin kuvaaja lakkaa olemasta lineaarinen, eikä jännityksen ja venymän suhde ole enää vakio.

Myötöraja ilmaisee lineaarisuuden päättymisen lisäksi myös rajan elastisen ja plastisen väänтymän välillä. Venymän ollessa myötörajan alapuolella, vääntymät ovat elastisia eli palautuvia: näyte palaa alkuperäiseen muotoonsa jännityksen laskiessa nollaan. Sen sijaan myötörajan yläpuolella muodonmuutokset ovat plastisia eli palautumattomia. Tällä alueella materiaali ei palaudu alkuperäiseen muotoonsa, vaan siihen jää pysyviä muodonmuutoksia.

Tutkittaessa näytteen kimmokerrointa on siis huomioitava, että näytteeseen aiheutetut venymät ovat tarpeeksi pienet, jotta myötöraja ei ylity. Edellisten huomioiden perusteella myötörajan ylittäminen ei aiheuta ainoastaan epätarkkuutta kimmokertoimen määrittämisessä, vaan voi myös vaikeuttaa mittausten toistettavuutta, jos tutkittavan alueen topografia muut-

tuu merkittävästi. Myös tässä tutkimuksessa tämä vaikutti käytettyjen voimien suuruuteen: jännittävä voima tuli valita siten, että venymä ei kasva liian suureksi.

Toinen tärkeä tekijä kimmokerrointa mitattaessa on ympäristön lämpötila. Kokeellisesti on havaittu, että kimmokertoimen arvo laskee lämpötilan kasvaessa [3]. Metallisidosten heikkeneminen korkeissa lämpötiloissa havaitaan myös arkisissa tilanteissa, esimerkiksi metalleja taottaessa. Tässä tutkimuksessa kaikki tehdyt mittaukset suoritettiin laboratorio-olosuhteissa, joten oletan lämpötilasta johtuvan poikkeaman olevan merkityksetön.

Viimeinen tärkeä seikka on huomioida, että kaikki materiaalit eivät noudata kuvan 1 mukaista jännitys-venymäkuvaajaa. Yksi tässä tutkimuksessa käytetty materiaali, polymetyylimetakrylaatti (PMMA), on hauras muovimateriaali, joka kestää merkittävää jännitystä mutta venyy hyvin vähän. PMMA:n tapauksessa kuvaaja koostuu ainoastaan lineaarisesta osasta, joka päättyy murtumispisteeseen pienemmällä venymän arvolla (verrattuna taipuvan materiaalin murtumispisteeseen). Kuten aiemmin todettu, käytettävät voimat tulee harkita tarkkaan, jotta näyte ei vaurioidu ja näin mittausten toistaminen vaikeudu.

3 Atomivoimamikroskopia ja voimakäyrät

Atomivoimamikroskopia (AFM) on skannausmikroskopiaan (Scanning Probe Microscopy, SPM) pohjautuva tutkimusmenetelmä, jonka avulla kerätään pääasiassa topografiatietoa näytteiden pinnoilta.

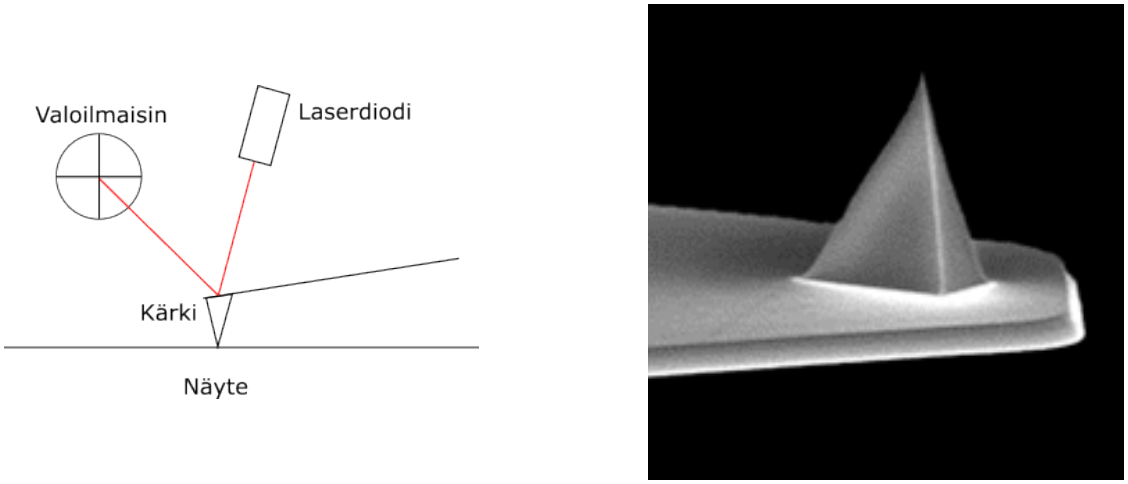
Atomivoimamikroskoopin toimintaperiaate perustuu näytteen pinnan atomikuoren tunnusteluun terävän kärjen avulla. Jotta kärki kykenisi tunnustelemaan pinnan atomikerrosta, on mittauskärjen oltava hyvin terävä, kärjen säteen ollessa vain nanometrejä. Kärki on kiinnitetty taipuisaan palkkiin, jonka toiseen päähän (samaan jonka toisella puolella kärki sijaitsee) kohdistetaan heijastuva lasersäde. AFM:n toimintaperiaate sekä kuva kärjestä näkyvät kuvassa 2.

Lähestyessään pintaa materiaalin ja näytteen atomit vuorovaikuttavat Van der Waals -voimien kautta heikossa vuorovaikutuksessa, jolloin palkki ”nytkähtää” kohti pintaa. Tämän jälkeen kärkeä työnnetään materiaalin pintaa vasten, kunnes kärjen ja pinnan välinen voima saavuttaa asetetun maksimivoiman arvon. Tämän jälkeen kärkeä nostetaan pinnasta takaisin aloituskorkeuteen. Mittauskärjen liikettä seurataan palkkiin kohdistuvan säteen liikkeeseen perustuen fotoilmaisimessa.

Käyttämällä mittauskärkeä lukuisissa lähekkäin sijaitsevissa pisteissä näytteen pinnalla, ja toistamalla edellä kuvailtu toimenpide, saadaan näytteestä kerättyä tietoa pinnan topografiasta tietyllä alalla.

Nykyään atomivoimamikroskoopit ovat kehittyneet keräämään myös muuta kuin korkeustietoa. Työssäni käytin Brukerin Dimension Icon -atomivoimamikroskooppia, joka kykenee keräämään korkeustiedon lisäksi näytteen pinnalta voimakäyriä mittaamalla kärkeen kohdistuvan voiman määrää.

Voimakäyrät kuvaavat kärjen ja näytteen välistä vuorovaikutusta lähellä näytteen pintaa ja kärjen koskettaessa näytettä. Voimakäyrien muoto perustuu mittauskärjen ja näytteen pinnan välisiin vuorovaikutuksiin. Lähestyttäessä näytettä kärjen ja pinnan atomien väliset vuorovaikutukset aiheutuvat näiden välisistä Van der Waals -vuorovaikutuksista. Kun kärkeä ale-



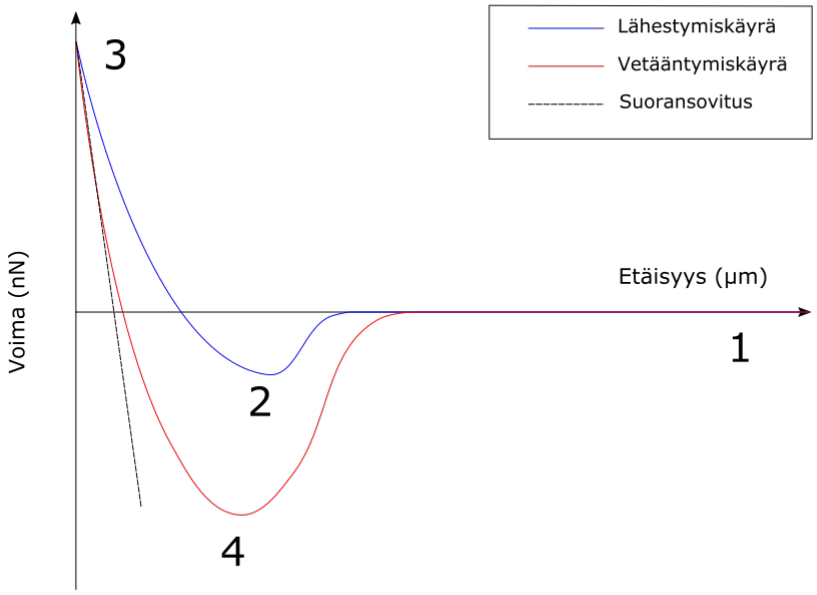
Kuva 2. Vasemmalla periaaatekuvaa AFM:n toiminnasta. Kärjen liikettä näytteen päällä seurataan diodilta tulevan lasersäteen ja detektorin avulla. Oikealla kuva ScanAsyst Air -kärjestä [4].

taan irrottamaan näytteestä, kappaleiden välinen adheesio aiheuttaa sen, että kärjen irrottamiseksi näytteestä tarvitaan voimaa, mikä näkyy kuvaajassa laskeumana etääntymiskäyrällä. Esimerkkikuva tavanomaisesta voimakäyrästä on kuvaajassa 3

Kärjen ja näytteen välinen vuorovaikutus on seuraavanlainen: aluksi kärki ja näyte ovat suhteellisen kaukana toisistaan, jolloin näiden välillä ei ole merkittävästi voimaa (VdW-voimat ovat kääntäen verrannollisia etäisyyden kuudenteen potenssiin). Tietyllä välimatkalla näytteen pinta-atomit ja kärjen huipun atomit ovat sen verran lähellä, että palkki, johon kärki on kiinnitetty, alkaa taipua.

Kun kärki on näytteen pinnalla, alkaa palkki taipua, kun sitä lasketaan yhä lähemmäs näytettä. Näytteen pinta alkaa hylkiä kärkeä, jolloin kärkeen kohdistuva voima ja jousen venymä alkavat nopeasti kasvaa. Kärki pureutuu alas niin kauan, kunnes palkin taipuma saavuttaa mittausohjelmaan asetetun maksimiarvon. Tämän jälkeen kuvauskärkeä aletaan nostaa näytteestä. Pinta hylkii tässä vaiheessa vielä kärkeä, mutta kun kärkeä on nostettu tiettyyn pisteeseen, se alkaakin vetää kärkeä puoleensa. Kärki irtoaa näytteestä lopulta erottavan voiman saavuttaessa adheesiovoiman arvon, jolloin jousen taipuma menee nopeasti nollaan. Tämän jälkeen palkki nostetaan takaisin määrättyyn aloituskorkeuteen.

Kuinka voimakäyrät siis auttaisivat erottamaan materiaaleja keskenään? Tämän asian selvit-



Kuva 3. Esimerkkikuva tyypillisestä voimakäyrästä. Kuvaan on merkitty kohdat, jossa kärki on etäällä näytteestä (1), jossa lähellä näytettä kärki painuu kohti näytteen pintaa (2), jossa kärki on tunkeutunut maksimisyvyyteensä ja sitä aletaan vetää pois (3), ja jossa kärki lopulta irtoaa näytteestä (4). Lisäksi kuvaajaan on merkitty vetääntymiskäyrän vasempaan laitaan sovitettu suora.

tämiseksi tulee tietää, miten tutkittava materiaali ja valitun tipin kärki vaikuttavat voimakäyrän muotoon. Tämän tutkielman osalta tyydyn keskittymään tutkimuskysymykseni kannalta olennaiseen asiaan, eli kärjen ja näytteen kontaktiosioon.

Näytteen jäykkyyttä kuvataan suureella k_s. Näytteen jäykkyys on ominaisuus, joka liittyy voima–etäisyys -kuvaajan kulmakertoimeen seuraavasti: Mittauskärjen ja näytteen välistä etäisyyttä voidaan kuvata yhtälöllä [5]

$$D = Z_p + Z_c + \delta, \tag{3.1}$$

missä D on kärjen ja näytteen välinen etäisyys, Z_p on laitteiston skannerin korkeusarvo, Z_c

on palkin mittauspään poikkeama normaaliarvostaan ja δ on deformaation arvo, joka kertoo kuinka syvälle kärki on painautunut näytteeseen.

Kontaktissa kärki on kiinni näytteessä, jolloin $D = 0$. Jos oletetaan systeemin olevan tasapainossa, kärki painaa näytettä samalla voimalla kuin näyte taivuttaa palkkia, jolloin Newtonin toisen lain mukaan $k_c Z_c = k_s \delta$, missä k_c on palkin jousivakio. Sijoittamalla nämä yhtälöön (3.1) saadaan

$$k_c Z_c = -\frac{k_c k_s}{k_c + k_s} Z_p = k_{eff} Z_p \tag{3.2}$$

Tästä yhtälöstä voidaan päätellä näytteen jäykkyyden vaikutus kuvaajaan: Jos systeemi on kunnollisesti kalibroitu, niin laserin poikkeama voidaan muuttaa palkkia taivuttavaksi voimaksi $F_c = k_c Z_c$. Näin ollen kontaktiosiossa voimakäyrä on yhtälön (3.2) mukaan lineaarinen, ja käyrän kulmakerroin on efektiivinen jäykkyys k_{eff}.

Huomattavaa on, että jos $k_c >> k_s$, niin $k_{eff} \approx k_s$, eli kulmakerroin kuvaa näytteen jäykkyyttä. Tämän takia voimakäyrän kulmakertoimen määrittäminen on yksi tapa tarkastella näytteen mekaanisia ominaisuuksia.

Toisaalta jos $k_s >> k_c$, niin $k_{eff} \approx k_c$, jolloin kulmakerroin kuvaa enemmän käytetyn palkin ominaisuuksia. Siksi on oltava tarkkana, että palkin jousivakio on tarpeeksi suuri näytteeseen verrattuna. Toisaalta liian kova palkki voi vaurioittaa näytettä, joten pehmeitä näytteitä tarkasteltaessa tulee välttää liian jäykkiä palkkeja. Edellä todettujen ongelmien välttämiseksi on tärkeää valita sopiva palkkityyppi.

Miten näytteen jäykkyys kytkeytyy kimmokertoimeen? Tämä selviää yhtälöstä

$$k_s = 2a \left(\frac{1 - \nu_s^2}{E_s} + \frac{1 - \nu_t^2}{E_t} \right)^{-1}, \tag{3.3}$$

missä suure a on kärjen ja näytteen välinen kontaktisäde, ja ν sekä E ovat vastaavasti Poissonin luku ja kimmokerroin. Yhtälössä (3.3) alaindeksi s tarkoittaa näytteen suuretta ja alaindeksi t kärjen suuretta. Tavanomaisesti kärjen kimmokerroin on huomattavasti suurempi kuin näytteen, jolloin yhtälö (3.3) yksinkertaistuu muotoon

$$k_{\mathrm{s}} = 2a\frac{E_{\mathrm{s}}}{1-\nu_{\mathrm{s}}^2}, \tag{3.4}$$

jossa siis havaitaan, että kärjen ominaisuudet eivät vaikuta mitattuun jäykkyyteen. On kuitenkin havaittava, että tässä työssä tutkitut ohutkalvonäytteet omaavat huomattavan suuria kimmokertoimen arvoja ($E_{\mathrm{s}} > 10$ GPa), jotka ovat lähes samaa kokoluokkaa piinitridikärjen kimmokertoimen ($E_{\mathrm{t}} = 140$ GPa) kanssa. Tämän takia yhtälön (3.4) käyttäminen aiheuttaa jonkin verran epätarkkuutta näytteen jäykkyyden mittauksessa.

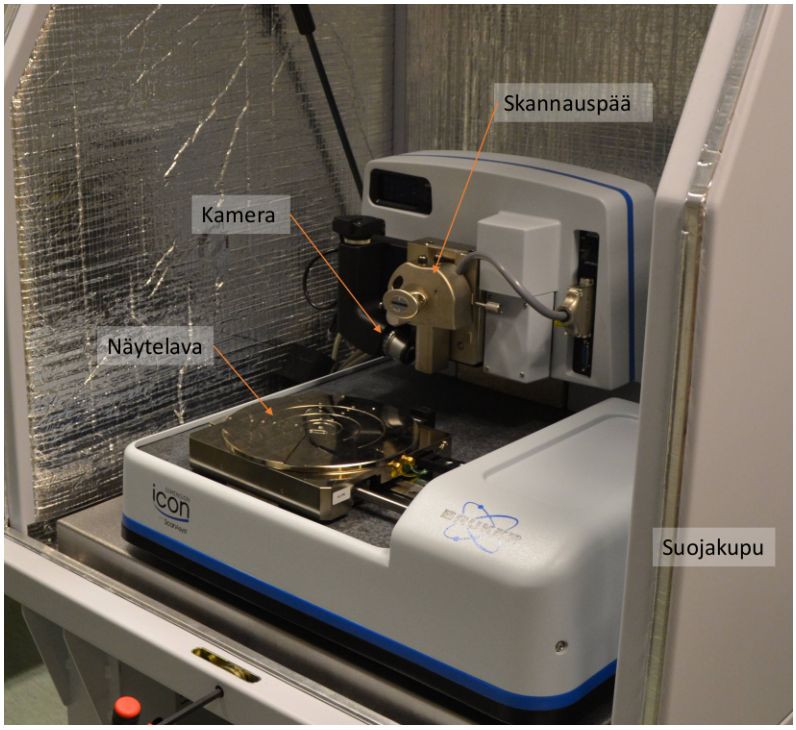
Yhtälöistä (3.3) ja (3.4) nähdään, että kulmakertoimen arvo olisi lähes suoraan verrannollinen näytteen kimmokertoimeen, joka on materiaalille ominainen mekaaninen ominaisuus. Näin ollen mittaamalla voimakäyrien kulmakertoimia vetääntymiskäyrän kontaktiosan kohdilta, materiaaleja tulisi kyetä erottamaan toisistaan. Jotta tuloksia voitaisiin verrata toisiinsa, tulee mitattujen voimakäyrien olla mahdollisimman tarkkoja.

Tarkkojen voimakäyrien saamiseksi on tärkeää tietää eri mittausparametrien vaikutukset, sekä olla selvillä mitä voimakäyrästä halutaan määrittää. Tässä tutkimuksessa käyn läpi tiettyjen mittausparametrien, kuten Peak Force Setpointin ja Peak Force Amplituden arvojen vaikutusta kulmakertoimen arvoihin, sekä pyrin määrittämään optimaaliset mittausparametrit AFM-mittausten osalta.

Mittaukset tapahtuivat pääasiassa siten, että kalvot kuvattiin ensin HIMillä, jossa työstön avulla myös kalvon sisemmät kerrokset ja muut pinnanalaiset rakenteet tuotiin työstämällä esille. Tämän jälkeen näyte vietiin AFM:lle, jossa voimakäyriä keräämällä saatiin eroteltua materiaaleja kulmakertoimia tarkastelemalla. Ennen aiempaan kirjallisuuteen etenemistä täytyy käydä läpi tarkemmin nanorakenteiden tutkimista.

3.1 Atomikerroskasvatus nanorakenteiden suojana

Nanorakenteiden poikkileikkausten tunnustelu tällä menetelmällä on hieman haastavaa, sillä HIM-työstön seurauksena rakenteet usein tuhoutuvat, tai sitten työstettävistä rakenteista vyöryy ainetta takana olevien rakenteiden päälle. Näin ollen työstämisen takia voidaan menettää kriittistä informaatiota rakenteista ja niiden poikkileikkauksista.



Kuva 4. Kuva Jyväskylän Nanotiedekeskuksen Dimension Icon -atomivoimamikroskoopista.

Edellä mainittuja vaurioita on mahdollista välttää tekemällä näytteisiin pinnoitus ennen työstämistä. Tällainen pinnoittaminen pitää huolen siitä, että työstettävää materiaalia poistuu mahdollisimman tasaisesti, eikä työstö turmele esiin tuotuja rakenteita ja poikkileikkauksia.

Pinnoittamiseen on tarjolla useita eri menetelmiä kuten sputterointi tai työstöhöyrystys. Tässä kaikista sopivin menetelmä on atomikerroskasvatus (ALD) [6]. Menetelmän avulla pinnoitusta voidaan tehdä tasaisesti näytteeseen, aina pienimpiä onkaloita myöten. Koska tässä kyseessä on nanorakenteista, tämä ominaisuus on hyvin tärkeä. Pieni paksuus takaa myös sen, että pinnoitus on mahdollista tehdä ALD:llä suhteellisen nopeasti, huolimatta menetelmän hitaudesta muihin kasvatusmenetelmiin verrattuna.

3.2 Aiempia atomivoimamikroskoopilla tehtyjä mittauksia

AFM:ää hyödynnetään usein nanoskaalan rakenteiden karakterisoinnissa. Yhdessä tutkimuksessa nanoputkien mekaanisia ominaisuuksia pyrittiin määrittämään AFM:n avulla [7]. Tässä tutkimuksessa käsiteltiin nanoindentaatiota ja DMT-mallia, jotka liittyvät olennaisesti myös

tähän tutkielmaan. Työssä mainitaan lisäksi ”Buecklen sääntö”, jonka mukaan AFM mittaa luotettavasti ainoastaan siinä tapauksessa, että tunkeutumissyvyys on vähemmän kuin 10 prosenttia kerroksen syvyydestä. Tässä tutkimuksessa kuitenkin osoitettiin, että kyseinen sääntö ei aina välttämättä pidä paikkaansa. Lisäksi artikkeli nostaa esiin myös huomion epätarkkuudesta, jos tunkeutumissyvyys on samaa kokoluokkaa pinnan epätasaisuuden kanssa.

Nanoindentaatioon on keskitytty paljon, kuten esimerkiksi kollageenisäikeiden kimmokertoimia tutkittaessa [8]. Kyseisessä tutkimuksessa määritettiin onnistuneesti kollageenisäikeiden kimmokertoimia nanoindentaation avulla, ja tuotiin esille terävän kärjen käytännöllisyys nanometrien kokoisia kappaleita tutkittaessa. Saatu kimmokertoimien arvot olivat kuitenkin laajalta alueelta 3,7 – 11,5 GPa, ja työssä esiintyi merkittäviä virhelähteitä, johtuen esimerkiksi käytetystä referenssipalkista.

Keskitetyn ionisuihkun ja AFM:n käytöstä kappaleen mekaanisten kertoimien ominaisuuksien selvittämiseksi on tutkittu aiemminkin, esimerkiksi tässä rotan viiksikarvaa tutkittiin leikkaamalla se ionisuihkun avulla ja tutkimalla paljastunutta poikkileikkausta AFM:llä [9]. Tutkimuksen avulla onnistuttiin saamaan uutta tietoa rotan viiksen koostumuksesta ja pinnan Youngin moduulien hajonnasta. Tuloksista saatiin myös selville, että rotan viikset ovat pehmeämpiä sen keskeltä kuin ulommilta reunoilta.

Tässä työssä näytteen tutkiminen tapahtui Brukerin kehittämän Quantitative NanoMechanics (QNM) kuvantamismenetelmää, joka sallii näytteen nanoskaalan ominaisuuksien, kuten kimmokertoimen tai adheesion mittaamiseen. Menetelmää on käytetty paljon juuri kimmokertoimen määrittämiseen näytteissä [10, 11, 12]. Tutkituissa materiaaleissa on merkittävää, että niiden kimmokertoimet ovat olleet luokkaa $0.1 - 10$ GPa, eli verrattain pienet verrattuna yleisiin ALD-ohutkalvomateriaaleihin.

Valitettavasti kärjen kulumisesta QNM-kuvantamismoodin aikana ei ole kovin paljon aiempia tutkimuksia. Etenkin kontaktimoodiin perustuvassa AFM:ssä kärjen kunto on merkittävä huoli [13], ja sen seuraamiseen on pyritty määrittämään karakterisointimenetelmiä [14]. Kummassakin tutkimuksessa havaittiin käytetyn kärjen TEM-kuvista merkittävää kulumista, mikä johtuu kärjen liikuttelusta näytteen pinnalla. Tämän seurauksena mittauksen resoluutio kärsi pahoin, eikä näytteen pintaa mitattu niin tarkkaan kuin kulumattoman kärjen kanssa.

Samoin kärjen liikkeen suuntainen voima kasvoi kärjen ja näytteen kosketuspinta-alan kasvaessa.

Myös voimakäyriin on perehdytty merkittävästi kirjallisuudessa, ja voimakäyrää on pyritty kuvaamaan tietyillä malleilla. Näistä malleista yksinkertaisin on Hertzin malli [5], jossa lävistävän kärjen muoto oletetaan puolipalloksi. Mallissa voimaa kuvataan tunkeutumissyvyyden funktiona

$$F = \frac{4}{3}\frac{E_\mathrm{s}}{1-\nu_\mathrm{s}^2}\sqrt{R\delta^3}, \tag{3.5}$$

jossa E_s ja ν_s ovat samat kuin yhtälössä (3.3). Lisäksi R on kärjen säde ja δ on tunkeutumissyvyys. Hertzin mallin tapauksessa on oletettava, että kärkeen vaikuttava adheesiovoima on olematon, eli toisin sanoen $F_\mathrm{Adh} << F_\mathrm{PFS}$. Tämän tutkielman näytteiden osalta tätä oletusta ei voi tehdä, sillä adheesiovoimat tulevat olemaan hyvin merkittäviä, kuten myöhemmin huomataan. Näin ollen ilmiön tutkimiseen tarvitaan tarkempi malli.

Hertzin mallia tarkkaan seuraava DMT-malli ottaa huomioon adheesiovoiman lisäämällä yhtälöön (3.5) adheesiovoimatermin F_Adh, jolloin voimaksi saadaan

$$F = \frac{4}{3}\frac{E_\mathrm{s}}{1-\nu_\mathrm{s}^2}\sqrt{R\delta^3} - F_\mathrm{Adh}. \tag{3.6}$$

DMT-malli toimii parhaiten tapauksissa, joissa kärjen säde on pieni ja tutkittava näyte on erittäin jäykkä [5]. Tässä työssä tutkittavat näytteet täyttävät juuri nämä vaatimukset, ja siksi tulen palaamaan DTM-malliin vielä myöhemmin tutkielmassa.

Muita malleja ovat JKR-malli [10], joka on toinen vaihtoehto DMT-mallille, ja Sneddonin malli [15], joka Hertzin mallista poiketen olettaa kärjen kartion muotoiseksi. Kumpikin näistä malleista soveltuu parhaiten vähäisen jäykkyyden omaaviin näytteisiin, joiden tunkeutumissyvyydet olisivat huomattavat. Tämän työn näytteet ovat kuitenkin sellaisia, että edellä oleva ehto ei täyty, eivätkä nämä mallit ole sopivia kuvaamaan mittauksia.

DMT-mallin ja QNM:n käyttöä materiaalien karakterisoinneista on tehty tutkittaessa asfalttia [16]. Tutkimuksen tuloksena saatiin selville, että DMT-mallin mukaan saadut maksimiarvot

liki kaksinkertaistuivat sekoitetuissa näytteissä kontrollinäytteeseen verrattuna.

Merkittävä asia, johon tässäkin tutkimuksessa törmätään, on alempien kerrosten vaikutus ylempien kerrosten voimamittauksiin. Yhdessä julkaisussa [15] Sneddonin mallia parannettiin ottamaan huomioon näytteen alla oleva materiaali yhtälössä

$$F = \frac{8E_{\mathrm{s}} \tan\theta\,\delta^2}{3\pi} \times \left\{ 1 + 1.7795 \frac{2\tan\theta}{\pi^2} \frac{\delta}{h} + 16\,(1.7795)^2 \tan^2\theta \frac{\delta^2}{h^2} + O\left(\frac{\delta^3}{h^3}\right) \right\}, \tag{3.7}$$

missä F, E ja δ ovat samat kuin yhtälössä (3.5), θ on kartiomaisen kärjen puolikulma ja h on näytteen korkeus. Poissonin vakion v arvoksi oletetaan $0,5$. Yhtälössä on otettu huomioon kerrosten vaikutus termillä δ/h, eli indentaation ja näytteen paksuuden suhteella (Termin mennessä nollaan yhtälö palautuu alkuperäiseen Sneddonin malliin). Tämän yhtälön on todettu vastaavan paremmin tehtyjä mittauksia, kun alla olevan näytteen paksuus on ollut pienimmillään. Kyseinen tutkimus on osoitus siitä, että alla olevan materiaalin vaikutusta on mahdollista kuvata yhtälöllä (3.7).

4 Heliumionimikroskooppi

Tavallisten elektronisuihkuilla toimivien mikroskooppien rinnalle on viime vuosikymmeninä ilmestynyt myös ioneihin perustuvia mikroskooppeja. Yksi näistä mikroskoopeista on heliumionimikroskooppi (HIM), josta ALIS Corporation kehitti ensimmäisenä prototyypin 2000-luvulla [17]. Kuvassa 5 on esitelty Jyväskylän yliopiston HIM-laitetta.

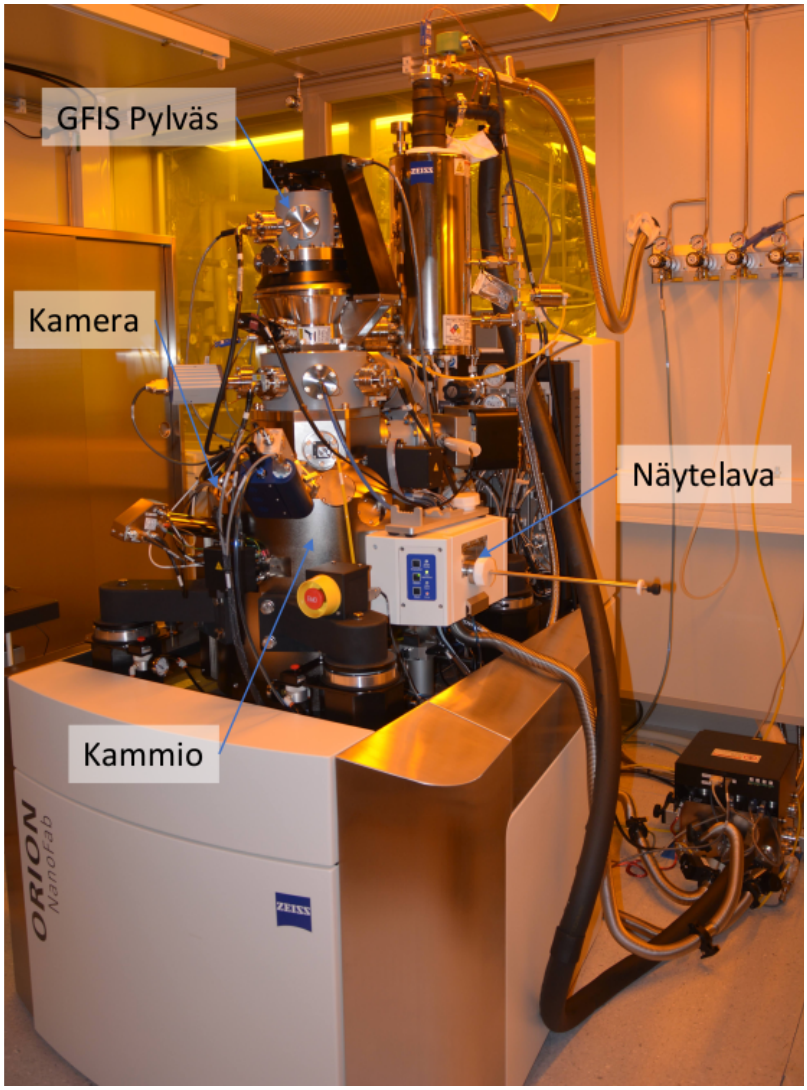
4.1 Heliumionimikroskopian historiaa

Ennen HIMin keksimistä käytettiin biologisten näytteiden ja ohutkalvojen kuvantamiseen elektronimikroskooppia (SEM), joka itsessään kykenee 1–20 nm:n resoluutioon [18]. SEMissä oli kuitenkin muutamia merkittäviä heikkouksia, kuten esimerkiksi kyvyttömyys kuvata eristäviä materiaaleja, jolloin kuvantamisessa jouduttiin turvautumaan mm. näytteen päällystämiseen johtavalla kalvolla.

Mikro- ja nanokokoluokan työstämisiä ja tutkimista on toteutettu myös ennen HIMiä yhdistetyllä FIB/SEM-laitteistolla, joka mahdollisti materiaalin työstämisen lisäksi nopean tavan kuvata työstön jälkeä. Tällaisella instrumentilla on pyritty tutkimaan muun muassa nanokuviointia. Jäädytettyjä biologisia näytteitä on esimerkiksi pommitettu keskitetyllä ionisuihkulla ja kuvattu heti perään SEMillä [19]. Tietysti HIMillä vastaava pystytään tekemään vielä paremmalla resoluutiolla ja tarkemmalla työstöllä.

Työstäminen ionisuihkulla on itsessään paljon tutkittu aihe viime vuosikymmeninä, ja siksi FIB-työstöjä on tutkittu useita keinoja, myös SPMää käyttäen. Esimerkiksi hiilinanoputkia on työstetty FIB:llä, jonka jälkeen näytettä tutkittiin Raman spektroskopialla [20]. Tuloksista huomataan, että hiilinanoputkia työstämällä FIB levittää amorfista hiiltä ympäri näytettä. Tämä saattanee johtua siitä, että työstö oli tehty käyttäen galliumionisuihkua, joka muodostuu merkittävästi raskaammista ioneista kuin helium. Tämä on tärkeä huomioida myös heliumilla työstettäessä.

SEMillä näytteitä on mahdoton työstää kuvausten lomassa, sillä elektronin massa ei riitä sputteroimaan näytteessä olevia atomeja irti. Jos näytteitä halutaan tutkia pinnan alta, joudu-



Kuva 5. Kuva Jyväskylän Nanotiedekeskuksen ORION NanoFab-heliumionimikroskoopista.

taan työstettävät näytteet siirtämään toiselle laitteelle ja takaisin, mikäli tarkoitus on kuvata näytettä pinnan alapuolelta.

Myös työstämiseen käytetty gallium ei ole täysin ongelmaton, sillä hyvin raskaana ionina se helposti sputteroi kohdemateriaalia, ja myös saastuttaa kohteen itsellään. Tarvitaan kevyempää ionityyppiä, joka ajaisi saman asian kuin gallium, tuhoamatta kuitenkaan näytettä yhtä paljon. Tarvitaan neonia.

Bevellien, eli näytteen pinnan suuntaan viistosti tehtyjen työstöjen, käytöstä on näyttöä yhdessä tutkimuksessa, jossa pyrittiin karakterisoimaan hiili-ionisuihkulla työstettyä kraatteria

SIMS:n ja AFM:n avulla [21]. Kuten tässä työssä, sisäisiä rakenteita tuotiin esille työstämisen avulla, ja AFMllä saatiin tietoa työstön tasaisuudesta ja syvyydestä. Tutkimuksessa karakteroistiin Irganox 1010 -näytettä, johon oli upotettu neljä reilun kolmen nanometrin paksuisia Irganox 3114 -kerrosta eri syvyyksille. Tutkimuksen tulokset antavat osoittaa, että AFMää voidaan käyttää työstön tutkimisessa ainakin työstetyn pinnan muotoa selvitettäessä.

4.2 Heliumionimikroskoopin toimintaperiaate

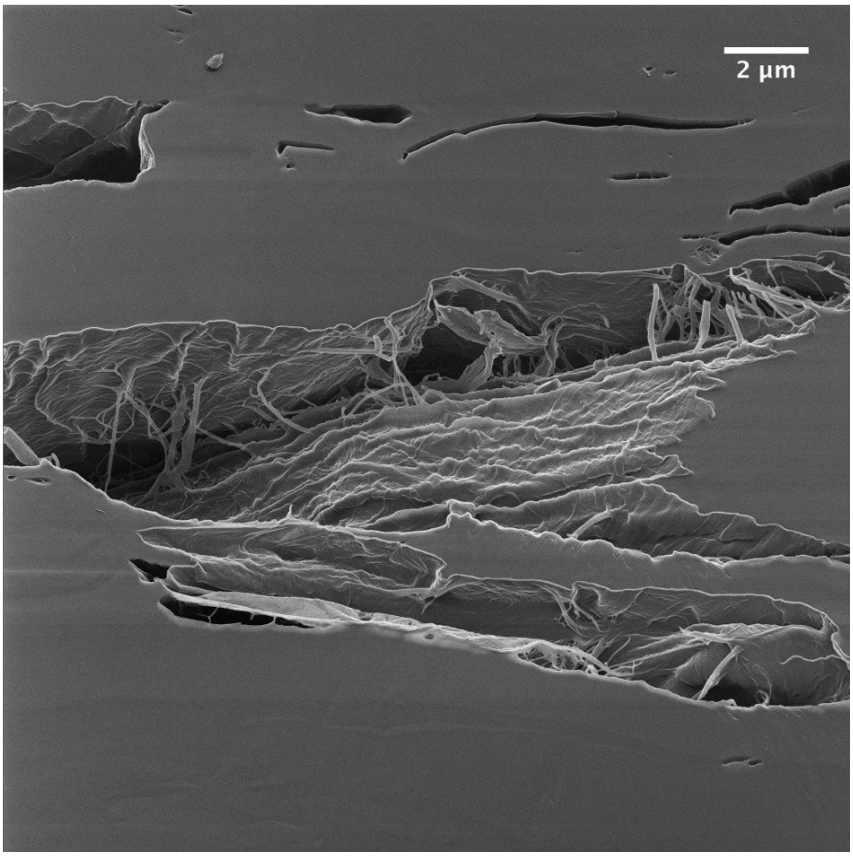
HIMin toimintaperiaate muistuttaa paljolti SEMiä. Aluksi heliumkaasua ionisoidaan, jotta kaasua olisi mahdollista kiihdyttää sähkökentän avulla. Heliumin ionisointi tapahtuu terävän volframikärjen avulla, jonka päässä kolme atomia on tiukasti yhdessä, muodostaen trimeriksi kutsutun muodostelman. Kolmen atomin yhdistelmä on valittu, koska sen on osoitettu olevan kestävämpi kuin yhden atomin monomeeri tai kahden atomin dimeeri.

Kun ionit on saatu kiihdytettyä, tulee ne seuraavaksi ohjata näytteen pinnalle haluttuun kohtaan. Avuksi tulee linsseistä ja poikkeuttimista muodostettu systeemi, joiden avulla ionisuihkua liikettä voidaan ohjata. Aluksi suihku kohdistetaan pieneen reikään, jonka tarkoitus on päästää läpi ainoastaan pieni, keskimmäinen osa säteestä. Reiän tarkoitus on parantaa kuvan laatua estämällä säteen reunaosien pääsy näytteeseen. Toimenpide vähentää optisen aberraation määrää. Reiän jälkeiset linssit kohdistavat säteen näytteen pinnalle.

Kun ionisuihku osuu näytteeseen, se irrottaa törmäyksessä pinnan atomien kanssa sekundaarielektroneja aivan kuten elektronisuihkukin. Sekundaaarielektronit ohjataan HIMissä ET-ilmaisimelle, jossa ne havaitaan fotomonistinputken avulla, ja ne toimivat pääasiallisina kuvan muodostajina HIM-kuvantamisessa. Koska ET-ilmaisimella voidaan mitata ainoastaan sekundaarielektronien intensiteettejä, ovat HIMillä otetut kuvat tavanomaisesti mustavalkoisia.

Heliumionimikroskoopin keskeinen hyöty on laaja syvyysterävyys. Lisäksi heliumionimikroskoopin avulla voidaan kuvata eristeitä käyttäen elektronitykkiä näytteen varauksen neutralointiin. Pienen pyyhkäisykokonsa ansiosta HIMillä on myös erittäin hyvä erotuskyky.

Kuvassa 6 on esitelty HIMillä saatua kuvaa Broad Ion Beam (BIB) -tekniikalla työstetystä

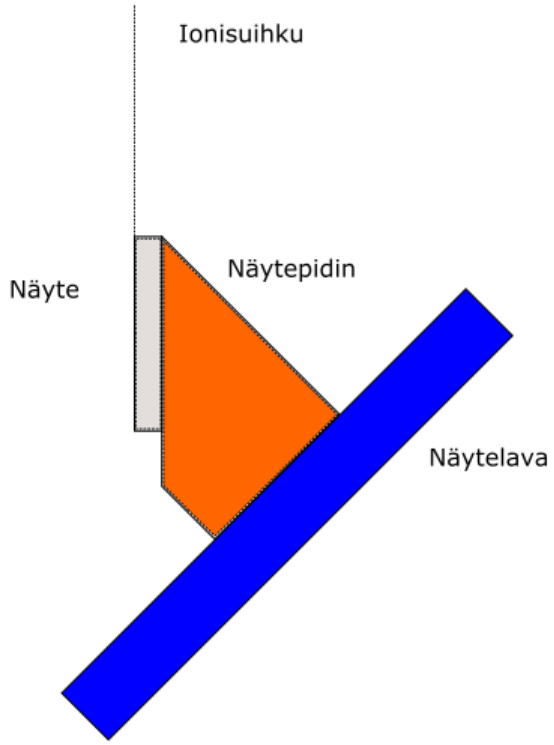


Kuva 6. HIMillä kuvattua, BIB-tekniikalla työstettyä nanoselluloosapaperia. Näytteen pinnalla, josta työstö on tehty, havaitaan hyvin vähän yksityiskohtia, verrattuna onkaloihin näytteessä.

näytteestä (BIB ei ole samanlainen työstötapa kuin FIB). Kuvasta nähdään, että HIMillä saadaan paljon tietoa kuvattavan alueen ollessa epätasainen: työstössä paljastuneessa aukossa näkyy tarkemmin yksityiskohtia kuin tasaisella pinnalla.

Tässä työssä tärkein HIMin ominaisuus on kyky työstää tutkittavaa näytettä pommittamalla sitä neonioneilla, jotka sputteroivat materiaalia pois tieltään. Näin tutkijalla on mahdollisuus saada esiin myös syvempiä kerroksia näytteestä. Tämä on myös syy, minkä takia HIM on valittu työhön käytettäväksi kuvausmenetelmäksi: se on erinomainen tapa tuoda esiin pinnanalaisia kerroksia ohutkalvosta ja samalla kuvata näytettä tarkalla resoluutiolla.

Millainen työstön kannattaa siis olla, jotta alempia kerroksia voitaisiin tutkia? Kuten aiemmin todettu, erittäin ohuet kalvot peittyvät herkästi reunaefektien alle. Työstämällä näytettä



Kuva 7. HIM-työstö näytteen pinnan suuntaisesti.

pienellä kulmalla lähes pinnansuuntaisesti voidaan mitattavaa aluetta kasvattaa sitä enemmän mitä loivemmalla kulmalla työstetään. Uusi tutkittava pituus on

$$L_{\text{eff}} = \frac{L}{\sin \alpha}, \tag{4.1}$$

missä L on ohutkalvon paksuus ja α on työstökulma. Työstämällä esimerkiksi kahden asteen kulmassa kasvaa tutkittavan osion pituus lähes 30-kertaiseksi, mikä on merkittävä suurennos. HIMin avulla voidaan siis kuvata sellaisiakin kalvoja, joita ei esimerkiksi SEMin avulla saataisi kunnolla näkyviin.

Jotta näytettä olisi mahdollista työstää pienessä kulmassa, tulee näytteen pinta saada samansuuntaiseksi ionisuihkun kanssa. Tässä nousee kuitenkin esille ongelma HIMin toimintaperiaatteen kanssa: Näytelavaa ei ole mahdollista kallistaa 90 asteen kulmaan. Tämän takia tasaisen näytepitimen päälle asetettu näyte on mahdotonta asettaa sopivaan kulmaan säteeseen nähden.

Kuvassa 7 on esitetty tapa, jolla ratkaisen ongelman tässä tapauksessa. Käyttämällä kaltevaa näytepidintä, näyte on mahdollista saada HIMin sisään valmiiksi vinossa asennossa. Näin ollen näytelavaa siirtämällä näyte saadaan yhdensuuntaiseksi ionisuihkun kanssa, vaikka lava ei itse kierrykään. Näyte on myös mahdollista saada vaakatasoon kiertämällä lavaa toiseen suuntaan. Tämä mahdollistaa näytteen kuvantamisen sekä ennen työstöä että työstön jälkeen.

HIMillä tehdyistä työstöistä oletetaan pystyvän saamaan hyvin tasaisia, jotta näytteen pintaa olisi mahdollista tarkastella AFM:llä ilman että pinnan topografia vaikuttaisi tuloksiin. Tämä tarkoittaa, että tutkittavat näytteet tulee valmistaa mahdollisimman tasaisiksi. Lisäksi FIB-tekniikalla työstetyt ovat vain muutaman mikrometrin levyisiä. Jotta työstöjä olisi mahdollista kuvata myöhemmin HIMillä ja AFM:llä, tulee työstöt merkitä tarpeeksi hyvin niiden paikantamiseksi. Nämä asiat ovat tärkeä ottaa huomioon tutkittavia näytteitä ja mittauksia suunniteltaessa.

5 Tutkittavat materiaalit ja mittaukset

Työn mittauslaitteisto koostui Jyväskylän Yliopiston Nanotiedekeskuksessa sijaitsevista Zeiss ORION NanoFab -heliumionimikroskoopista, jolla näytteitä työstettiin ja kuvattiin, sekä Brukerin Dimension Icon -atomivoimamikroskoopista, jolla suoritettiin voimakäyrien mittaus ja kuvantaminen. Näiden lisäksi tehdyistä työstöistä on otettu valokuvat optisella mikroskoopilla Nanotiedekeskuksen puhdastilassa.

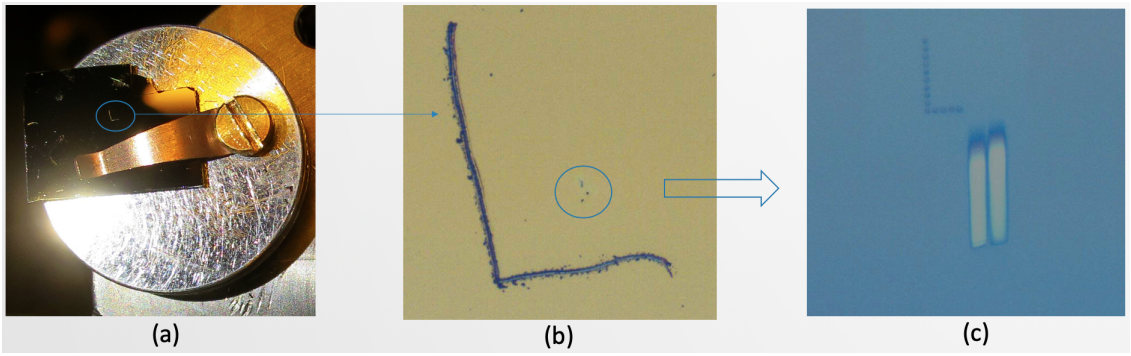
5.1 Näytteiden valmistus

Tutkittavat näytteet olivat pääasiassa höyrystämällä kasvatettuja ohutkalvoja, joiden koostumukset on esitelty taulukossa 1. Näiden lisäksi näytteisiin kuului spinnaamalla valmistettuja PMMA resistejä sekä muutama ALD-menetelmällä kasvatettu ohutkalvo. Toisin kuin edellä olevat metallit, näiden kimmokerroin on huomattavasti pienempi kuin metalleilla, ja ne sopivat juuri suuremman jousivakion omaaviin (> 10 N/m) käytettävissä oleviin kärkiin.

Resistit spinnattiin piisubstraattien päälle. Resistikerroksista tehtiin 100 nm:n, 200 nm:n ja 300 nm:n paksuisia, jotta alla olevan substraatin vaikutusta mittaustuloksiin voitaisiin testata. Lopulta 100 nm:n näyte jäi kuitenkin käyttämättä, koska arvioin sen olevan liian ohut.

Taulukko 1. Metallikerroskalvojen koostumuksia. Jokaisen kerroksen paksuus on noin 50 nm, paitsi PMMA:n jonka paksuus on 200 nm.

1. Kerros	2. Kerros	3. Kerros	4. Kerros	Substraatti
Cu	Al	Cu	Al	Si
Al	Cu	Al	Cu	Si
Cu	Ti	Al	Nb	Si
Al	Nb	Ti	Cu	Si
Al_2O_3	PMMA	Al_2O_3	PMMA	Si



Kuva 8. Kuvia työstöprosessista. (a) Näyte näytepitimessä, kuvaan merkitty L:n muotoinen naarmu. (b) Kyseinen naarmu lähempää kuvattuna optisella mikroskoopilla. Kuvaan on ympyröity työstön paikka. (c) HIMillä työstetty näyte kuvattuna optisella mikroskoopilla.

5.2 Mittausten kulku

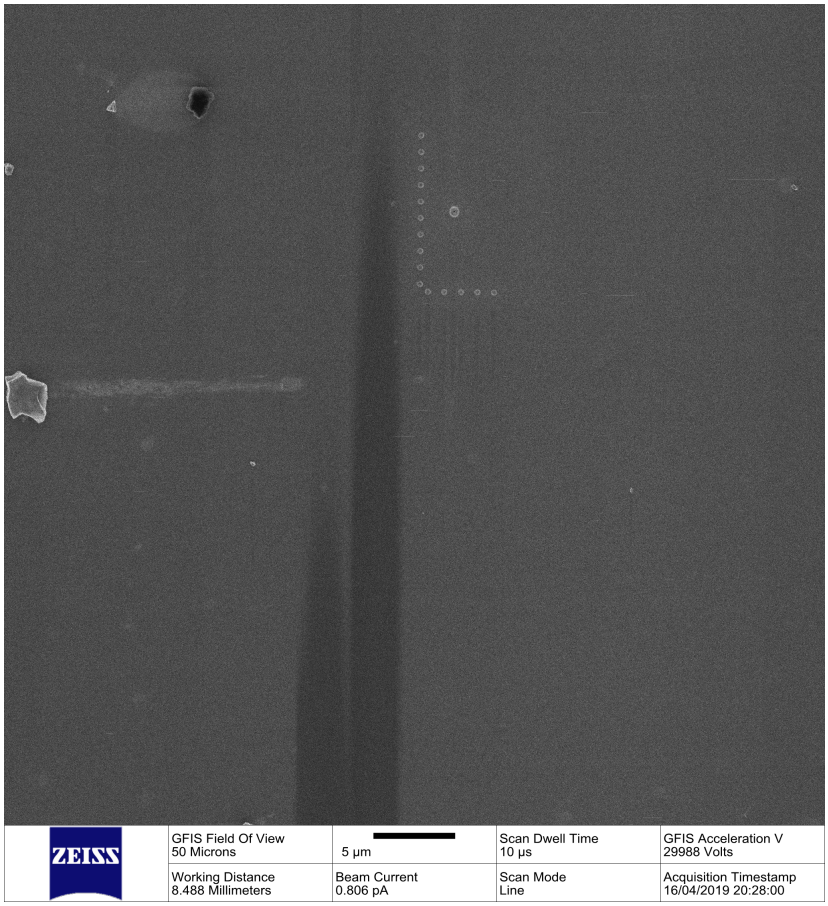
Työstöjen teko tapahtui tekemällä aluksi näytteisiin pieni L:n muotoinen naarmu timanttikynällä, joka näkyisi paljain silmin sekä optisessa mikroskoopissa. Tämä naarmu toimisi merkkinä tehdyn työstön paikantamiselle myöhemmin.

Toimenpiteen jälkeen ohutkalvonäytteet asetettiin näytepitimissä HIMin sisään, jossa tehdyt naarmut kuvattiin. Näytelava asetettiin kuvan 7 mukaisella tavalla pinnan suuntaisten työstöjen tekemiseksi.

Kun HIMin kuvat oli optimoitu mahdollisimman tarkasti, suunnitellun työstökohdan ympärille tehtiin vielä pienempi L:n muotoinen merkintä korkeaenergisen heliumionisäteen avulla. Tämän toimenpiteen tarkoitus on aiheuttaa kohoumia, jotka olisivat nähtävissä sekä optisella mikroskoopilla että AFM:llä, ja auttaisivat tehdyn työstön paikantamisessa. Tämän jälkeen suoritettiin itse työstö, jossa näytettä pommitettiin korkeaenergisellä neonionisäteellä, työstäen näytettä muutaman mikron alalta.

Kun työstöt oli tehty ja kuvattu, näytteet siirrettiin atomivoimamikroskoopille, jossa tehtiin mittaukset käyttäen Peak Force QNM -menetelmää. Mittauksen tarkoitus oli kerätä voimakäyriä näytteistä, yleensä työstöjen alueelta tai näiden läheltä.

AFM-tutkimusten tarkoituksena oli selvittää optimaaliset mittausparametrit, jotka takaisivat mahdollisimman tarkat mittaustulokset. Tätä varten mittauksissa pyrittiin varioimaan para-

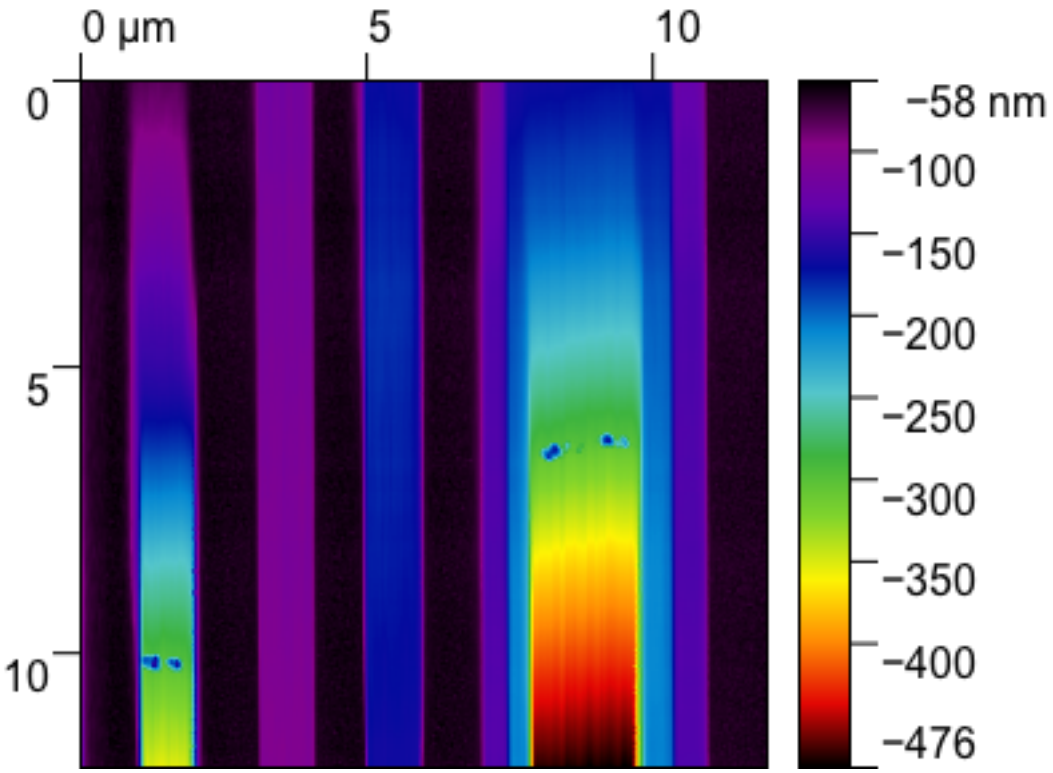


Kuva 9. HIM-työstetty piinäyte. Kuvassa näkyvät tummat pystysuorat viivat ovat työstöaluetta. Viivojen vieressä olevat, L-kirjaimen muodostavat kohoumat auttavat paikantamaan työstöalueet HIM-kuvausta varten.

metrejä, jotka voisivat olennaisesti vaikuttaa tuloksiin, ja pyrkiä näin kokeilemalla arvioimaan parametrien vaikutuksia. Tutkittuja parametrejä olivat:

- Peak Force Setpoint: Systeemiin syötetyn voiman arvo, jonka saavutettua mittausohjelmisto vaihtaa lähestymisestä erkaantumiseen.
- Peak Force Amplitude: Kärjen etäisyysakselin koko.
- Mittauskärki ja -palkki: Käytetyn kärjen materiaalin ja jousivakion vaikutus tuloksiin. Käytetyt kärkityypit: ScanAsyst Air, RTESPA300, RTESPA525, TAP300DLC (timanttikärki).

AFM:llä tehdyt mittaukset koostuivat aluksi laitteen käytön harjoittelemisesta. Tämän jäl-



Kuva 10. HIMillä eri kulmilla piihin työstettyjä työstöjä AFM:llä kuvattuna.

keen tein kuvaukset nanolaminaatti- ja metallikerrosnäytteille käyttäen ScanAsyst Air -kärkiä. Näiden mittausten jälkeen vaihdoin suuremman kulmakertoimen omaavaan RTSEPA525-kärkiin, joita käytin pääasiallisina mittauskärkinä. Seuraavat mittaukset keskittyivät tarkastelemaan mittausparametrien vaikutusta tuloksiin. AFM:llä tehdyistä mittauksista on kooste taulukossa 2

AFM:n voimakäyrämittaus tapahtuu palkista heijastuvan laserin avulla, valo heijastetaan valoilmaisimelle. Palkin nytkähtäessä kärjen vuorovaikuttaessa pinnan kanssa, laserin paikka valoilmaisimella muuttuu, mikä muuttaa valoilmaisimen jännitettä. Tarkan kalibraation myötä jännite voidaan muuttaa kärjen paikan poikkeamaksi. Näin ollen kärjen liikettä on mahdollista seurata systeemin avulla. Poikkeaman arvo on puolestaan mahdollista muuttaa voimaksi käyttäen palkin jousivakiota sekä yhtälöä $F = k\Delta x$, missä Δx on poikkeama.

Näytteiden epätasaisuuksien vuoksi tulee nostaa esille, kuinka AFM pyrkii pitämään kärjen ja pinnan välisen mahdollisimman tasaisena. Tämä tapahtuu takaisinkytkennän avulla, jossa systeemi laskee asetetun amplitudin ja todellisen amplitudin eron, ja tästä syntyvä signaali

kasvatetaan ja lähetetään skannerille. Näin takaisinkytkentä kykenee korjaamaan skannerin pystysuuntaista liikettä, pitäen mittauksen tasaisena.

Kaikki AFM:llä saadut voimakäyrät prosessoin lopulta oman ohjelmani kautta, jonka avulla pyrin selvittämään voimakäyristä saatavien suureiden jakaumaa mittauksessa. Ohjelmani toiminnasta on tarkemmin liitteessä A. Seuraavassa luvussa käyn läpi tällä ohjelmalla saamani kuvaajat ja pyrin niiden avulla tekemään päätelmiä mittaamistani näytteistä, kuten koostumuksesta ja topografiasta, sekä tuloksien jakaumista ja niiden perimmäisistä syistä.

Taulukko 2. AFM:llä tehtyjä mittauksia.

Päivämäärä	Käytetty kärki	Mittauksen tarkoitus
7.6.2018	ScanAsyst Air	Nanolaminaattien kuvantamista
24.8.2018	ScanAsyst Air	Metallikerroskalvon (Cu/Al/Cu/Al) mittaus
17.1.2019	RTESPA525	PFS:n ja PFA:n vaikutusten selvittäminen.
6.2.2019	RTESPA525	Ohutkalvojen (Taulukko 1) mittaus
8.2.2019	RTESPA525	Jatkoa 6.2.2019 mittaukselle
7.5.2019	ScanAsyst Air/RTESPA525	Piisubstraatin mittaus
17.7.2019	RTESPA525	Metallien ja resistipintojen vertailu
22.8.2019	RTESPA525	Tipin kulumisen tarkastelu
26.9.2019	RTESPA525	Metalli-Resistikalvokerrosten mittaus
10.10.2019	RTESPA300/TAP300DLC	Timanttitipin testaus ja vertailu

6 Mittaustulosten analysointi

Tässä luvussa käyn läpi työn aikana saamiani kuvaajia ja tuloksia, sekä pyrin selvittämään tulosten paikkansapitävyyttä teorian kannalta.

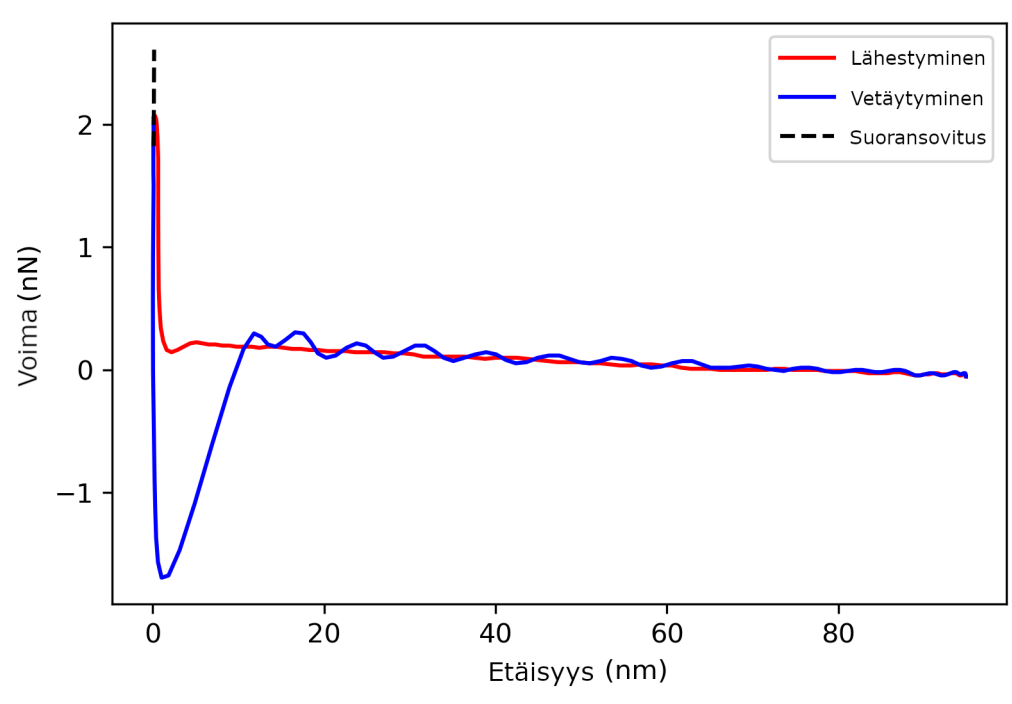
Kuvaajassa 11 on ScanAsyst Air-tipeillä otettu voimakäyrä alumiinioksidinäytteestä. Kuvaajasta huomataan, että voimakäyrä ei ole täysin vakio kärjen ollessa kaukana näytteestä, vaan voiman arvo heikkenee kärjen etääntyessä näytteestä. Tämä ei ole teorian mukaista käyttäytymistä, vaan voiman arvon pitäisi olla nollassa kärjen ollessa etäällä näytteestä. Häiriö voi johtua mahdollisesti joko optisesta interferessistä tai mahdollisesta pitkän etäisyyden EM-vuorovaikutuksesta.

Toinen merkittävä ongelma kuvaajassa, ja myös ratkaiseva syy siihen minkä takia ScanAsyst Air -kärjet eivät sovellu tutkielmaan, on vetäytymiskäyrän muoto lähellä näytettä. Kuten kuvasta huomataan, kuvaaja on koko voimakäyrää tarkasteltaessa lähes pystysuora. Tämä tarkoittaa, että kärki hädin tuskin läpäisee näytettä, vaan sen sijaan palkki taipuu voimakkaasti.

Ongelma aiheutuu työssä käytetystä palkista, jonka jousivakio k on $0,4$ N/m. Tämä on hyvin matala arvo ottaen huomioon, että näytteen Youngin moduuli > 100 GPa. Brukerin tietojen mukaan tälle alueelle sopiva jousivakio olisi luokkaa 350 N/m, mikä on yli 800-kertainen palkkiin verrattuna! Tätä voidaan myös tarkastella yhtälön (3.3) kautta: Sijoittamalla arvot yhtälöön näytteen jäykkyydeksi saadaan k_s 360 N/m. Tämä on jopa 900-kertainen palkin jousivakioon verrattuna!

Kuvaajassa 12 on voimakäyrän kulmakertoimen arvoja Peak Force Setpointin (PFS) arvolla 600 nN. Kuvaajasta huomataan, että tulokset noudattavat hyvin normaalijakaumaa. Tämä sopii yhteen sen kanssa, että AFM-mittauksissa saatu arvojakauma on myös normaalijakautunut. Tämä on seurausta keskeisestä raja-arvolauseesta, jonka mukaan satunnaisotos toisistaan riippumattomista ja identtisesti jakautuneista otoksista alkaa lähestymään normaalijakaumaa.

Kuvaajassa 13 on mitattuja kulmakertoimien arvoja eri PFS:n arvoille, kun näytteenä on ollut piisubstraatti ja AFM-kärkenä piinitridikärki (mallia RTESPA525). Mittaukset on tehty

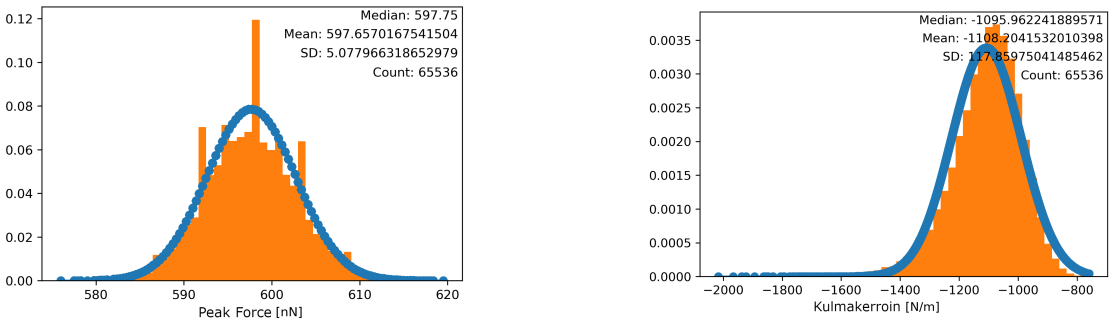


Kuva 11. Scanasyst Air-tipeillä saatu voimakäyräkuvaaja alumiinioksidinäytteen pinnalta.

peräkkäin. Kuvaajat nostavat esiin sen, että kulmakertoimien arvot saturoituvat, kun PFS > 600 nN. Vastaavanlainen käytös havaitaan myös resisteille tehdyissä mittauksissa.

Tärkeämpänä huomiona havaitaan, että resistin tapauksessa kulmakertoimien arvot ovat merkittävästi pienemmät piihin verrattuna 14. Tämä on hypoteesin mukainen, toivottu tulos: koska resistien kimmokerroin on huomattavasti matalampi, tulee myös kulmakertoimen olla itseisarvoltaan pienempi resistinäytteelle. Menetelmän avulla on siis mahdollista saada erotettua selvästi suuren kimmokertoimen (> 100 GPa) materiaalit sekä pienemmän kimmokertoimen (< 10 GPa) materiaalit toisistaan.

Tehtäessä peräkkäisiä mittauksia samalla mittauskärjellä, tulee ottaa huomioon kärjen kuluminen. Kun kärkeä painetaan toistuvasti pintaa vasten, kärjen päässä olevat atomit irtoavat ennen pitkää kärjestä, aiheuttaen sen tylsymisen. Tylsempi kärki ei enää tunkeudu näytteeseen yhtä hyvin, jolloin näyte tuntuu kovemmalta kuin aiemmin. Toinen tapa tarkastella asiaa on yhtälön (3.3) kautta: kun kärki kuluu, atomien karkaaminen päästä kasvattaa kärjen ja näytteen välistä kontaktisädettä. Sen seurauksena kontaktiosan kulmakerroin k_s kasvaa yhtälön (3.3) mukaan.



Kuva 12. Peak Forcen ja vetääntymiskäyrän kulmakertoimen arvojen jakaumat, sekä niille sovitetut normaalijakaumat piikalvoon tehdylle työstölle.

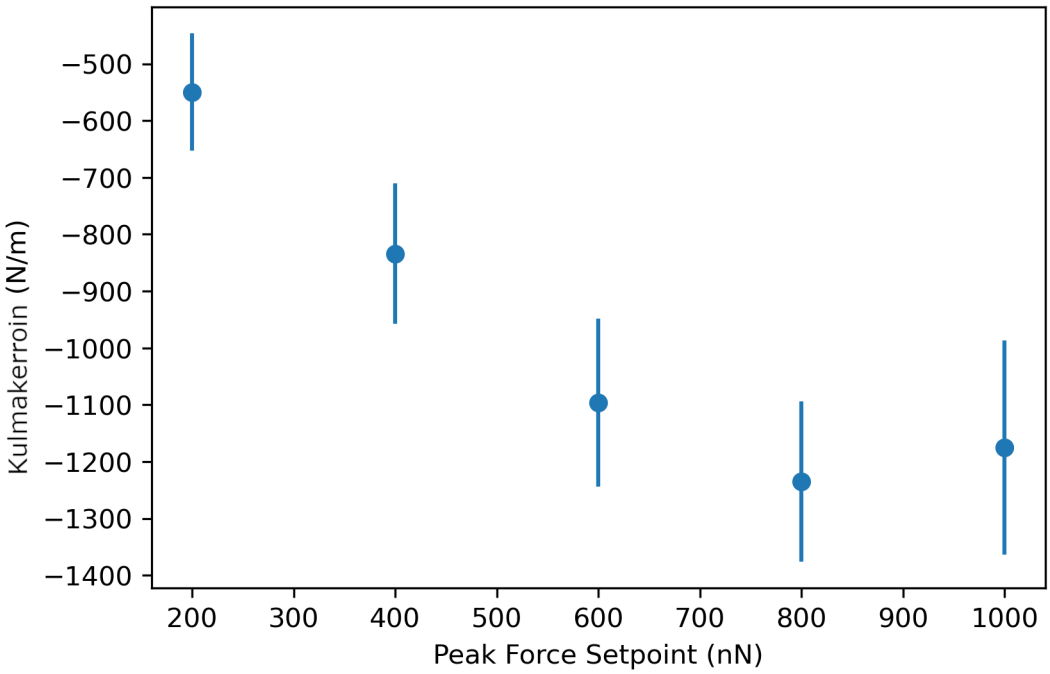
Kuvaajassa 15 on kulmakertoimien jakaumia piille tehtyyn mittaukseen eri skannauskoon arvoilla. Kaikissa skannauksissa kuvakoko on 1:1, joka on edellytys voimakäyrien mittaukseen Dimension Icon AFM:llä. Kuvaajasta havaitaan, että skannausalueen kasvattaminen näyttäisi aiheuttavan arvojen jakauman leviämisen.

Kuvasta on kuitenkin huomattava, että kaikki mittaukset suoritettiin samalla mittauskärjellä, joten edellä mainittu kärjen kuluminen vaikuttaa myös mitattuihin kulmakertoimien arvoihin. Kuvaajassa 16 on kuvattu kulmakertoimen arvojen jakauma mitattuna samalla kärjellä mittaussession alussa ja lopussa. Kuvaajasta havaitaan selvästi tipin kuluman vaikutus.

Kuvaajassa 17 on kulmakertoimien arvoja alumiini-kuparikerroksista koostuvaan ohutkalvoon tehdyn työstön alueelta. Kuvaajasta havaitaan selvästi arvojen vaihtelevan tehdyn työstön pätkältä. Tämä viittaa siihen, että työstöstä on mahdollista erottaa alumiini- ja kuparipinnat toisistaan.

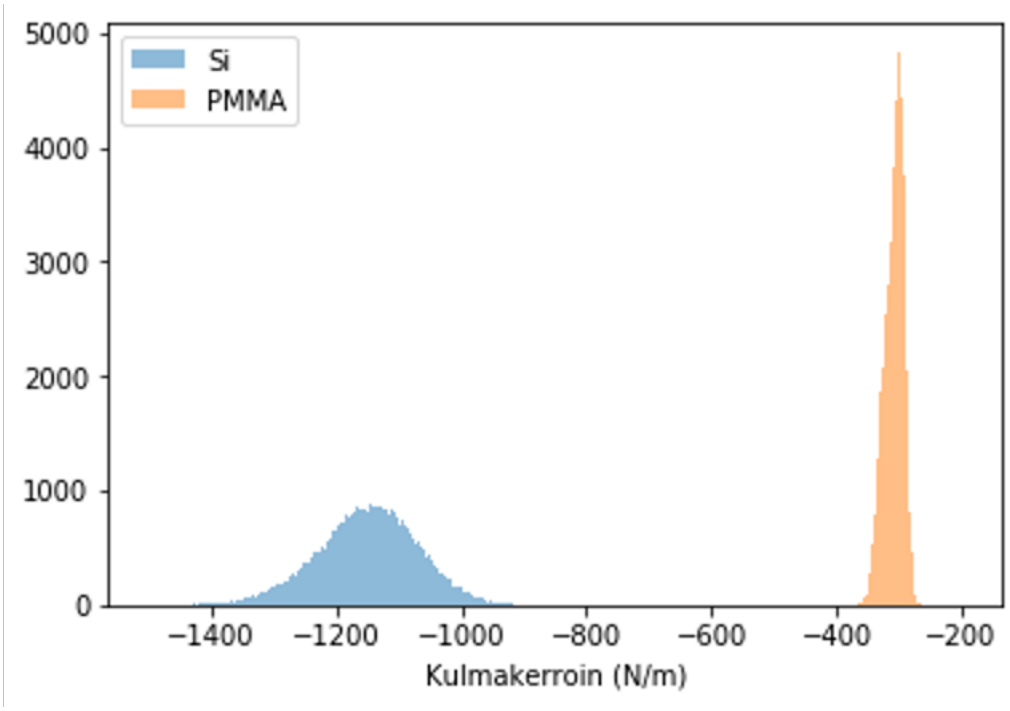
Mainitsin aiemmin tutkielmassa DMT-mallin (3.6), jota on käytetty selvittämään Youngin moduulin arvoa voimakäyristä. Tässä tutkimuksessa keskityin pääasiassa selvittämään voimakäyrien kulmakertoimia vetäytymiskäyrän avulla. DMT-mallin avulla olisi kuitenkin mahdollista saada kvantitatiivista tietoa materiaalien kimmokertoimista, mikä auttaisi tunnistamaan näytettä jos sen koostumusta ei tunneta. Tutkin mallin paikkansapitävyyttä saamiini voimakäyriin muuntamalla yhtälön (3.6) muotoon

$$\left(F - F_{\mathrm{Adh}})^{\frac{2}{3}}\right) = K \cdot \delta, \tag{6.1}$$

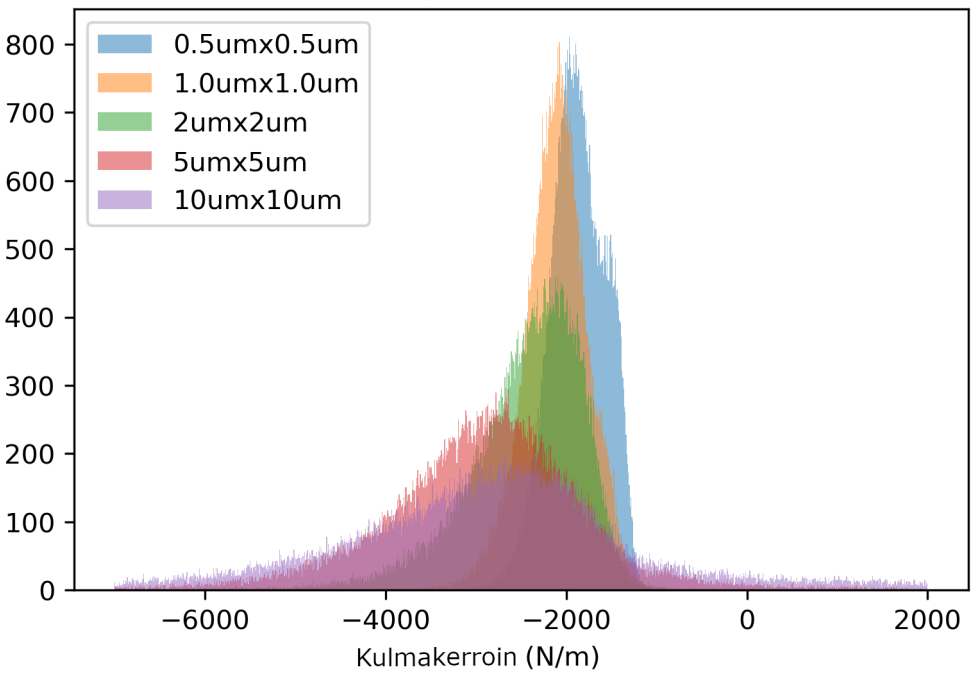


Kuva 13. Mitattujen kulmakertoimien arvoja piille eri PFS:n arvoilla. Pisteet kuvaavat saatujen jakaumien mediaaneja, virherajat kvartaalivälejä.

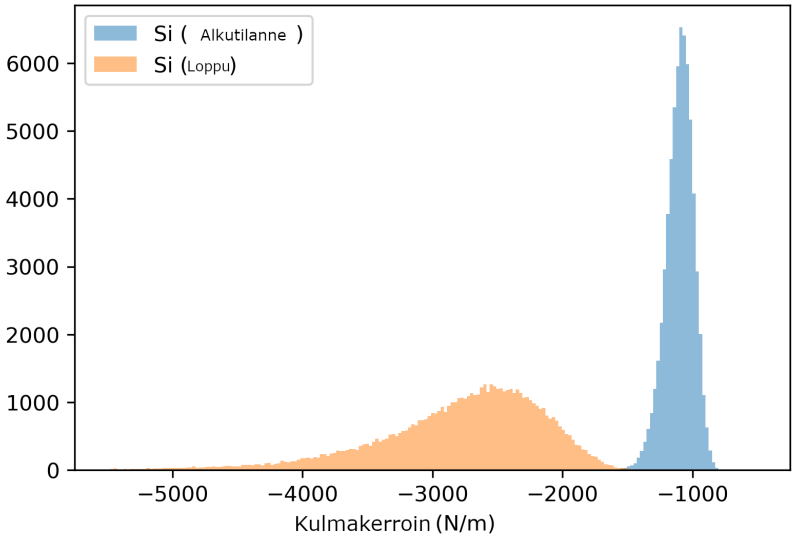
missä K on kulmakerroin, jonka tulisi olla vakio. Toisin sanoen sovittamalla voimakäyristä saamani pisteet, tarkemmin ottaen vetäytymiskäyrien pisteet kun etäisyys on pieni ja voimakäyrä on lähes lineaarinen, yhtälöön (6.1), tulisi tästä saatavan kuvaajan olla suora. Kuvassa 18 on muutamia tällaisia sovituksia eri materiaaleista mitatuista voimakäyristä. Kuten huomataan, kuvaajat eivät ole täysin suoria, vaan ne kaartuvat matalilla etäisyyksillä.



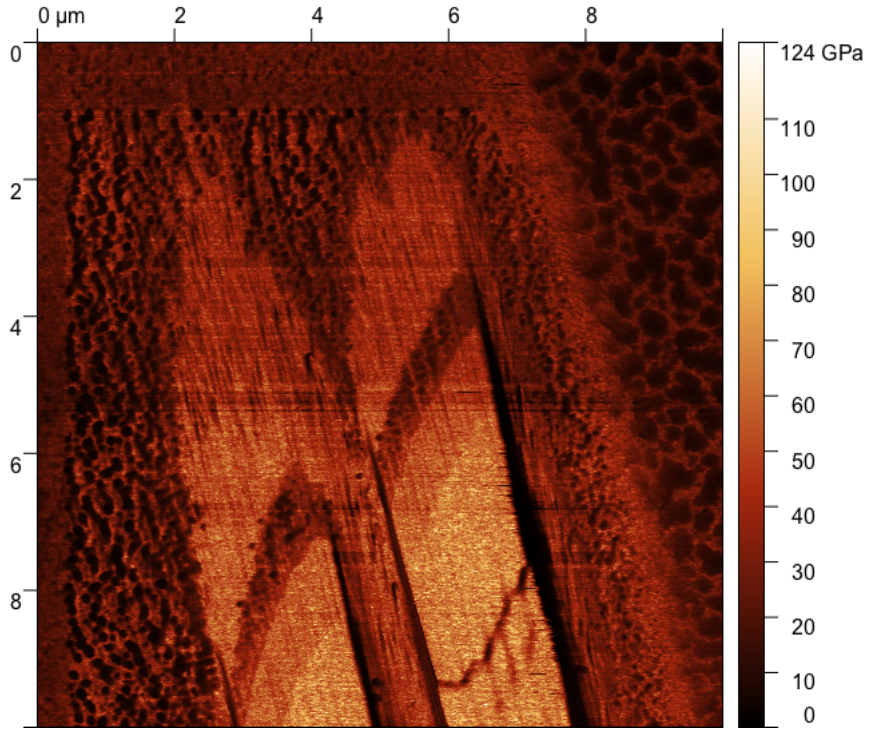
Kuva 14. Kulmakertoimien jakaumia piille ja resistille. Kummassakin mittauksessa PFS = 600 nN



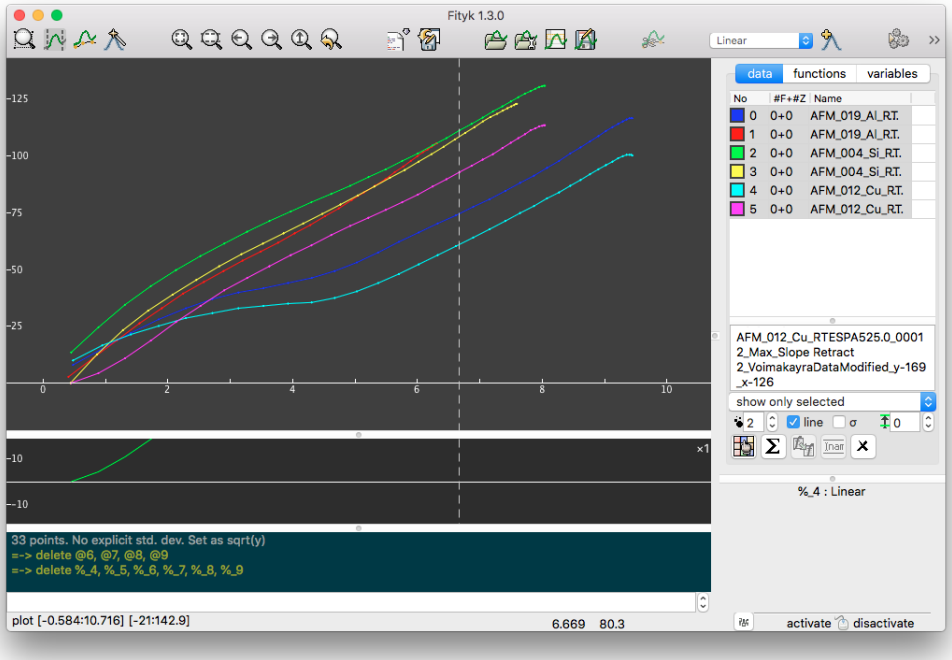
Kuva 15. Kulmakertoimen jakaumia piisubstraatissa erikokoisilla skannausalueilla.



Kuva 16. Kulmakertoimen jakaumia samalta alueelta pitkän mittausrupeaman alussa ja lopussa. Tipin tylsyminen on havaittavissa jakauman hajonnan merkittävänä kasvuna sekä keskipisteen siirtymisestä kohti jyrkempää kulmakerrointa.



Kuva 17. Kulmakertoimen arvoja alumiini-kupari-ohutkalvokerroksiin tehdyn työstön alueelta. Kuvasta voidaan arvioida kulmakertoimien arvojen avulla [2], että materiaalit ovat tummasta vaaleampaan alumiini, kupari ja pii.



Kuva 18. Yhtälöön (6.1) sovitettuja voimakäyrien arvoja vetäytymiskäyrän alkupäästä (Etäisyys < 10 nm). X-akseli on etäisyys, yksikkönä nm. Y-akseli on yhtälön (6.1) vasen puoli, yksikkönä $(\mathrm{nN})^{\frac{2}{3}}$. Näytteinä alumiini (sininen ja punainen viiva), kupari (turkoosi ja lila viiva) sekä pii (vihreä ja keltainen viiva). Käytetty tippi: RTESPA525. Kuvasta huomataan, että kuvaajat eivät ole täysin suoria kuin yhtälö (6.1) antaisi odottaa.

7 Pohdintaluku

Analyysi-osiossa saadut tulokset osoittavat, että voimakäyriä on mahdollista käyttää materiaalien erottamisessa. Piisubstraatin ja resistin huomattavasti erilaiset Youngin moduulit näkyvät voimakäyrien jyrkkyyksissä, ja kuten kuvaajasta 14 nähdään, saadut jakaumat eroavat merkittävästi toisistaan. Tässä piin tapauksessa voimakäyrä on merkittävästi jyrkempi kuin resistin tapauksessa.

Edellä saatu tulos selittyy sillä, miten kimmokerroin vaikuttaa voimakäyrään. Kuten teorialuvussa oli mainittu, kimmokerroin on materiaalin mekaaninen ominaisuus, joka kuvaa kappaleen jäykkyyttä. Toisin sanoen ominaisuus kuvaa miten paljon materiaali vastustaa työntävää kärkeä. Mitä suurempi vastus, sitä nopeammin voiman arvo laskee, kun kärkeä nostetaan näytteestä.

Menetelmällä on kuitenkin useita rajoituksia. Ensinnäkin kärki pysyy samanlaisena vain ihannetapauksessa. Käytännön kokeet ovat valitettavasti osoittaneet, että voimakäyrät jyrkkenevät mitä kauemmin samaa kärkeä käytetään mittauksessa. Oletan tämän johtuvan kärjen lukuisista kosketuksista näytteen pinnan kanssa, jonka seurauksena kärki tylsyy. Tämän lisäksi kärkeen tarttuu näytteen pinnalla olevia epäpuhtauksia, mikä näkyy tuloksissa samalla tapaa kuin tylsyminen.

Voisiko kärjen kulumista jotenkin estää? Valitettavasti tämä on hyvin hankalaa, sillä saatujen tulosten perusteella kulmakerroin pysyy mittauksissa vakiona ainoastaan, kun PFS > 600 nN. Tällä alueella kärki on puolestaan erityisen herkkä kulumaan osuessaan näin suurella voimalla näytteen pintaan. Kuitenkin hyvien arvojen saamiseksi nämä PFS:n arvot ovat olennaisia.

Jos mittauksia tehdään useita (tässä tapauksessa kerralla jopa 20 mittausta samasta kohdasta), kuluu kärki myös tällöin herkästi mittausten aikana. Tehdessä esimerkiksi 20 mittausta pikseliresoluutiolla 256x256, koskettaa kärki näytteen pintaa yli miljoona kertaa. Myöskään kärjen vaihtaminen jatkuvasti mittauksen aikana, tai timanttikärjen käyttö eivät käytännössä onnistu, sillä nämä lisäisivät merkittävästi AFM-mittauksen kuluja. Kärkiä pitäisi olla varalla kymmeniä, ja timantista valmistetut kärjet ovat huomattavasti kalliimpia tavanomaisiin

ScanAsyst Air -kärkiin verrattuna.

Jotta saatuja tuloksia voitaisiin lähteä kvantifioimaan, tulee kärjen ja jousen kuntoa pystyä seuraamaan tarkasti mittauksen edetessä. Tähän voisi olla apukeinona esimerkiksi referenssinäyte, jonka kulmakerroin olisi tiedossa käytetyillä mittausparametreilla. Esimerkiksi referenssiä voitaisiin mitata usein muiden mittausten lomassa, jolloin tulosten muutosta voisi tarkkaan seurata.

Toinen merkittävä ongelma mittauksissa on ohutkalvokerrosten paksuus verrattuna kärjen tunkeutumissyvyyteen. Kun kärki tunkeutuu syvälle materiaaliin, päällimmäisen kalvon syvyydestä riippuen alemmat kerrokset voivat vaikuttaa tuloksiin. Lisäksi kärkeä ei voi yrittää pitää vain pinnan tuntumassa, vaan tarkkojen tulosten saamiseksi kärjen on mentävä vähintään muutaman nanometrin verran syvälle materiaaliin. Tämä puolestaan aiheuttaa sen, että hyvin ohuissa kalvoissa vuorovaikutus alempien kerrosten kanssa on väistämätön.

Alla olevan substraatin osuutta tuloksissa voi tarkastella esimerkiksi HIM-työstön avulla. Tekemällä vinon työstön resistinäytteeseen ja tutkimalla voimakäyriä työstön suuntaan, meidän tulisi havaita jäykkyyden hidas kasvaminen, kun kärki alkaa yhä enemmän mitata resistin alla olevan piisubstraatin ominaisuuksia.

Koska tässä tutkimuksessa haettiin ainoastaan eroja voimakäyrien väleillä, mittauksia ei kalibroitu täysin oikeaoppisesti, vaan esimerkiksi palkin kulmakerroin oletettiin ideaaliseksi. Tuloksista saisi näin ollen tarkempia kalibroimalla tarkasti ilmaisimen ominaisuudet, kuten kärjen säteen ja jousen kulmakertoimen arvot. Koska tässä tutkimuksessa pyrin keskittymään kvalitatiiviseen analyysiin, minulle riitti vain havaita eroja voimakäyrien välillä. Edellä olevat toimet olisivat enemmänkin tarpeen kvantitatiivista analyysia tehdessä.

Kaiken kaikkiaan voidaan todeta, että AFM-mittauksissa on merkittäviä virhelähteitä. Yhdessä aiheeseen liittyvässä raportissa [22] on käyty läpi QNM:n virhelähteitä ja tultu tulokseen, että kimmokertoimen mittauksessa on n. 15 prosentin virherajat tyypillisiä, jotka kasvavat logaritmisesti näytteen (redusoidun) YM:n kasvaessa. Artikkelin antamat virherajat mittauslaitteistolle tukevat myös havaintoa, että tässä työssä käyttämäni RTESPA525-tipit ovat valitettavasti liian pehmeitä tiettyjen aineiden, kuten piin ja kuparin, karakterisointiin ohutkalvoissa. Tämä johtuu näiden materiaalien korkeasta kimmokertoimesta, joka on enemmän

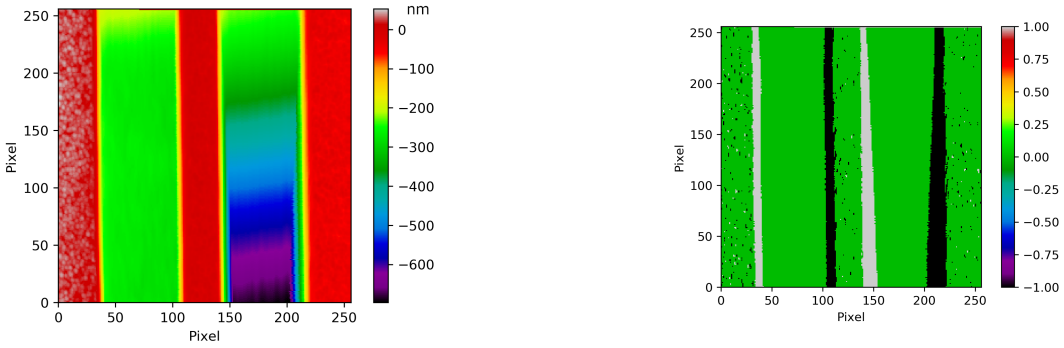
kuin RTESPA525-tipeille on tarkoitettu (> 50 GPa).

Onko kvantitatiivisen kimmomoduulin mittaaminen sitten ylipäänsä mahdollista? Ainakaan se ei olisi hyvin yksinkertaista, edelliset kohdat huomioiden. Jotta kvantitatiivisia tuloksia voitaisiin lähteä mittaamaan, tulee ensinnäkin AFM-mittausta parantaa. Koska kärjen säde ja jousen kulmakerroin vaikuttavat merkittävästi mitattuun kulmakertoimeen, tulee kehittää jonkinlainen järjestelmä monitoroimaan näitä suureita mittauksen aikana. Lisäksi mittaukset tulisi laatia siten, että nämä suureet eivät hirveästi muuttuisi mittauksen aikana. Ehdotan PFS:n pitämistä tarpeeksi matalana ja mittauksen pikselimäärän pitämistä kohtuullisena. Mittausta tehdessä kannattaa pohtia, tarvitaanko aina välttämättä suurinta mahdollista kuvakokoa, vai voisiko vähemmällä tulla toimeen, kuluttaen kärkeä vähemmän?

Kirjallisuuskatsauksessa nostin esiin DMT-mallin mahdollisena apuna Youngin moduulin määrittämisessä. Tässä tapauksessa yritettäessä soveltaa mallia tuloksiin havaitaan, että mallia ei pystytä sovittamaan saatuun mittaustulokseen. Sen sijaan, kuten kuvaajasta 18 huomataan, linearisoidut voimakäyrät eivät ole läheskään lineaarisia. Tulosten eroavuus mallin ennustuksesta johtunee kärjen pienestä tunkeutumissyvyydestä kovaa materiaalia vasten. Näin ollen kärki ei tunkeudu näytteeseen tarpeeksi syvälle, jotta Youngin moduulia voitaisiin tarkastella voimakäyrän avulla. Lisäksi epäilen, että hyvin jyrkässä voimakäyrässä esiintyy merkittävää virhettä käyrän alkupäässä, joka näkyy kuvassa 18 pienillä etäisyyden arvoilla. Tämän seurauksena menetelmällä on haastavaa saada tarkkoja YM:n arvoja, vaikeuttaen kvantitatiivista analysointia.

Aivan kaikkia mitattuja voimakäyriä ei voi käyttää kulmakertoimien määrittämiseen, kuten kuvaajasta 19 ilmenee. Tämä johtunee pinnan epätasaisuuksista, sillä esimerkiksi tässä ilmiö on selvin työstön reunoilla, missä on luonnollisesti myös epätasaisinta. Tämän takia voimakäyriä tutkittaessa olisi hyvä luoda jonkinlainen poisvalintajärjestelmä, jonka avulla selvästi epäkelvot voimakäyrät olisi mahdollista karsia pois.

Yksi keino karsia selvästi virheellisiä voimakäyriä olisi tarkistaa, poikkeaako käyrän PF:n arvo merkittävästi asetetusta PF:n arvosta. Jos arvoissa esiintyy huomattavaa (yli 10 prosenttiyksikön verran) poikkeamaa, niin tällöin voimakäyrän mittaamisessa on tapahtunut jokin ongelma, eikä mitattu voimakäyrä välttämättä edes muistuta ulkomuodoltaan tyypillistä voi-



Kuva 19. Yhden työstön topografiakuva (vasen kuva) ja pikselit, joista mitatuissa voimakäyrissä PF:n arvot poikkeavat yli 10 % asetetusta arvosta (oikea kuva, vihreästä poikkeavat pisteet). Oikeasta kuvasta havaitaan työstön reunojen olevan epätarkimpia pisteitä.

makäyrää. Tällöin myös käyrästä saadut arvot poikkeavat merkittävästi alueen keskiarvosta, aiheuttaen häiriötä tuloksiin.

Mistä arvojen poikkeavuus sitten johtuu? Teoriaosuudessa olin todennut, että PFS toimisi feedbackina, ja että kärkeä alettaisiin vetämään takaisin, kun kyseinen arvo ylitettäisiin. Kuitenkin saamani arvot osoittavat, että tämä sääntö ei päde epätasaisessa materiaalissa. Todellisuudessa PF voi heitellä suurestikin, jos näytteen pinta ei ole tasainen. Näin ollen PF:n virhe lieneekin riippuvainen topografian muutoksesta. Tämän vuoksi PF:n ääriarvojen karsiminen on yksi tapa karsia tuloksista pois epätarkkoja tuloksia.

Topografiavaihtelun lisäksi kuvasta 19 voidaan havaita myös, että ääriarvon tyyppi ja topografian muutoksen suunta ovat riippuvaisia keskenään. Tässä AFM-mittaus tapahtui vasemmalta oikealle, jolloin työstön vasemman reunan kohdalle mitattu korkeus tippui rajusti. Samalla myös PF:n arvot kasvoivat merkittävästi. Päinvastainen tapaus havaitaan työstön oikealla reunalla. Tästä voidaan päätellä, että Peak Forcen arvoja tarkastelemalla voidaan havaita ainakin merkittävimpiä topografian muutoksia näytteen pinnalla. Muutoksen suuntaa tarkastellessa pitää tietysti ottaa huomioon AFM-mittauksen suunta, jotta muutos olisi varmasti oikeansuuntainen.

8 Yhteenveto

Tämän työn tarkoitus oli perehtyä HIM-työstöllä ja AFM:n nanomekaanisella kartoituksella tapahtuvaan uuteen menetelmään ohutkalvojen materiaalikerrosten karakterisoimiseksi. Työn aikana minun tuli pohtia menetelmän toimivuutta, ja jos sen avulla saataisiin minkäänlaisia tuloksia, selvittää keinoja tulosten tarkentamiseksi.

Materiaalien erottaminen toisistaan onnistuu tarkastelemalla AFM-mittauksista saatuja voimakäyriä ja analysoimalla saatujen käyrien kontaktiosioiden kulmakertoimien jakaumia. Näin olen osoittanut, että menetelmän avulla on mahdollista erottaa ohutkalvossa olevat materiaalit toisistaan ainakin, kun näiden kimmokertoimet poikkeavat riittävän paljon toisistaan.

Tuloksia kuitenkin haittaa ohutkalvomateriaalien merkittävän suuret kimmokertoimet, jotka ovat tavanomaisille tipeille liian suuria. Näin ollen ohutkalvotutkimukseen menetelmä vaatisi esimerkiksi timanttikärkiä metallien pinnan tehokkaampaan läpäisemiseen. Kyseisten tippien korkean hinnan takia tämä ei kuitenkaan ole taloudellisesti kannattavaa kuin erikoistapauksissa.

Tässä työssä olen menetelmää tutkiessani keskittynyt tulosten kvalitatiiviseen analyysiin. Jos tuloksia haluttaisiin tutkia kvantitatiivisesti, tulee menetelmää varten luoda useampia referenssimateriaaleja, joiden avulla voidaan tarkemmin määritellä työn aikana syntyneitä tuloksia.

Yksi merkittävistä työssä nousseista ongelmista on AFM-mittauslaitteistossa tipin kärjen kunto, joka tutkimuksen mukaan heikkenee selvästi mittauksen aikana ja muuttaa saatuja tuloksia merkittävästi. Yksi parannuskohde menetelmässä olisikin selvittää tarkalleen, kuinka kärjen kunto tarkalleen muuttuu työn kuluessa. Tämä onnistuisi esimerkiksi tässäkin työssä käytetyllä menetelmällä, jossa kärkeä käytettäisiin mittausten välillä referenssinäytteellä. Myös mittauslaitteiston kalibrointi olisi tarpeen kvantitatiivisten tulosten saamiseksi. En perehtynyt kalibrointiin merkittävästi, koska työni keskittyi kvalitatiiviseen analyysiin.

Edellä tehtävän referenssianalyysin tueksi voisi myös perehtyä tarkemmin AFM-mittauksessa käytettyjen parametrien vaikutukseen. Tässä työssä kävin läpi hieman PFA:n ja PFS:n vai-

kutuksia tuloksiin, ja sain suuntaa antavia tuloksia siitä, minkä suuruisia arvojen tulisi olla. Koska metalleilla tutkittavat voimakäyrät ovat usein jyrkkiä, tarvitaan pieni PFA:n arvo, jotta vuorovaikutuskohdassa olisi riittävästi tarkasteltavaa tietoa. PFS:n arvoa määritettäessä tulisi löytää tasapaino tarvittavan deformaation ja tipin kontaminaation välttämisen väliltä.

Tipin lisäksi toinen merkittävä tekijä, joka vaikuttaa tulosten luotettavuuteen, on HIM-työstön tasaisuus. Kuten osoitin edellisessa luvussa, pinnan epätasaisuus näkyy huomattavina poikkeuksina saaduissa voimakäyrissä. Tämä aiheuttaa merkittävän virheen tuloksiin. Hyvin usein täysin tasainen pinta on kuitenkin hankala toteuttaa, mikä näkyy myös tässä työssä tehdyissä työstöissä. Näin ollen tulosten parantamiseksi tulisi myös tutkia keinoja parantaa HIM-työstön laatua erilaisissa ohutkalvoissa.

Lähteet

[1] Lawrence L. Kazmerski. ”Analysis and characterization of thin films: A tutorial”. en. *Solar Cells* 24.3 (heinäkuu 1988), s. 387–418. ISSN: 0379-6787. DOI: 10.1016/0379-6787(88)90091-9. URL: https://www.sciencedirect.com/science/article/pii/0379678788900919 (viitattu 22. 05. 2022).

[2] *Young's Modulus, Tensile Strength and Yield Strength Values for some Materials*. URL: https://www.engineeringtoolbox.com/young-modulus-d_417.html (viitattu 14. 07. 2022).

[3] Reza Rahemi ja Dongyang Li. ”Variation in electron work function with temperature and its effect on the Young's modulus of metals”. *Scripta Materialia* 99 (2015), s. 41–44. ISSN: 1359-6462. DOI: https://doi.org/10.1016/j.scriptamat.2014.11.022. URL: http://www.sciencedirect.com/science/article/pii/S135964621400493X.

[4] *ScanAsyst - Air | Silicon Nitride Probes*. URL: https://www.brukerafmprobes.com/p-3726-scanasyst-air.aspx (viitattu 14. 07. 2022).

[5] Hans-Jürgen Butt, Brunero Cappella ja Michael Kappl. ”Force measurements with the atomic force microscope: Technique, interpretation and applications”. *Surface Science Reports* 59.1 (lokakuu 2005), s. 1–152. ISSN: 0167-5729. DOI: 10.1016/j.surfrep.2005.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0167572905000488 (viitattu 07. 08. 2019).

[6] Steven M. George. ”Atomic Layer Deposition: An Overview”. *Chemical Reviews* 110.1 (tammikuu 2010). Publisher: American Chemical Society, s. 111–131. ISSN: 0009-2665. DOI: 10.1021/cr900056b. URL: https://doi.org/10.1021/cr900056b (viitattu 19. 03. 2023).

[7] Benedikt R. Neugirg ym. ”AFM-based mechanical characterization of single nanofibres”. *Nanoscale* 8 (16 2016), s. 8414–8426. DOI: 10.1039/C6NR00863A. URL: http://dx.doi.org/10.1039/C6NR00863A.

[8] Marco P. E. Wenger ym. ”Mechanical Properties of Collagen Fibrils”. *Biophysical Journal* 93.4 (elokuu 2007), s. 1255–1263. ISSN: 0006-3495. DOI: 10.1529/biophysj.106.103192. URL: http://www.sciencedirect.com/science/article/pii/S0006349507713838 (viitattu 01. 08. 2019).

[9] Vahid Reza Adineh ym. ”Multidimensional characterisation of biomechanical structures by combining Atomic Force Microscopy and Focused Ion Beam: A study of the rat whisker”. *Acta Biomaterialia* 21 (2015), s. 132–141. ISSN: 1742-7061. DOI: 10.1016/j.actbio.2015.03.028. URL: http://www.sciencedirect.com/science/article/pii/S1742706115001385 (viitattu 20. 12. 2018).

[10] Maxim E. Dokukin ja Igor Sokolov. ”Quantitative Mapping of the Elastic Modulus of Soft Materials with HarmoniX and PeakForce QNM AFM Modes”. *Langmuir* 28.46 (marraskuu 2012), s. 16060–16071. ISSN: 0743-7463. DOI: 10.1021/la302706b. URL: https://doi.org/10.1021/la302706b (viitattu 22. 08. 2019).

[11] Jozef Adamcik ym. ”Measurement of intrinsic properties of amyloid fibrils by the peak force QNM method”. en. *Nanoscale* 4.15 (heinäkuu 2012), s. 4426–4429. ISSN: 2040-3372. DOI: 10.1039/C2NR30768E. URL: https://pubs.rsc.org/en/content/articlelanding/2012/nr/c2nr30768e (viitattu 22. 08. 2019).

[12] Adriana N. Frone ym. ”Morphology and thermal properties of PLA–cellulose nanofibers composites”. *Carbohydrate Polymers* 91.1 (tammikuu 2013), s. 377–384. ISSN: 0144-8617. DOI: 10.1016/j.carbpol.2012.08.054. URL: http://www.sciencedirect.com/science/article/pii/S0144861712008284 (viitattu 22. 08. 2019).

[13] Koo-Hyun Chung, Yong-Ha Lee ja Dae-Eun Kim. ”Characteristics of fracture during the approach process and wear mechanism of a silicon AFM tip”. *Ultramicroscopy* 102.2 (tammikuu 2005), s. 161–171. ISSN: 0304-3991. DOI: 10.1016/j.ultramic.2004.09.009. URL: http://www.sciencedirect.com/science/article/pii/S0304399104001810 (viitattu 21. 08. 2019).

[14] Jingjing Liu ym. ”Method for Characterizing Nanoscale Wear of Atomic Force Microscope Tips”. *ACS Nano* 4.7 (heinäkuu 2010), s. 3763–3772. ISSN: 1936-0851. DOI:

10.1021/nn100246g. URL: https://doi.org/10.1021/nn100246g (viitattu 21. 08. 2019).

[15] Núria Gavara ja Richard S. Chadwick. ”Determination of the elastic moduli of thin samples and adherent cells using conical atomic force microscope tips”. en. *Nature Nanotechnology* 7.11 (marraskuu 2012), s. 733–736. ISSN: 1748-3395. DOI: 10.1038/nnano.2012.163. URL: https://www.nature.com/articles/nnano.2012.163 (viitattu 25. 06. 2019).

[16] A. M. Feroze Rashid ja Zahid Hossain. ”Morphological and nanomechanical analyses of ground tire rubber-modified asphalts”. en. *Innovative Infrastructure Solutions* 1.1 (joulukuu 2016). ISSN: 2364-4176, 2364-4184. DOI: 10.1007/s41062-016-0036-5. URL: http://link.springer.com/10.1007/s41062-016-0036-5 (viitattu 11. 06. 2019).

[17] B. W. Ward, John A. Notte ja N. P. Economou. ”Helium ion microscope: A new tool for nanoscale microscopy and metrology”. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena* 24.6 (marraskuu 2006), s. 2871–2874. ISSN: 1071-1023. DOI: 10.1116/1.2357967. URL: https://avs.scitation.org/doi/full/10.1116/1.2357967 (viitattu 05. 08. 2019).

[18] K. C. A. Smith ja C. W. Oatley. ”The scanning electron microscope and its fields of application”. en. *British Journal of Applied Physics* 6.11 (marraskuu 1955). Publisher: IOP Publishing, s. 391–399. ISSN: 0508-3443. DOI: 10.1088/0508-3443/6/11/304. URL: https://doi.org/10.1088/0508-3443/6/11/304 (viitattu 22. 04. 2022).

[19] J. Hazekamp ym. ”Focussed ion beam milling at grazing incidence angles”. English. *JOURNAL OF MICROSCOPY* 242.1 (huhtikuu 2011), 104–110. ISSN: 0022-2720. DOI: {10.1111/j.1365-2818.2010.03466.x}.

[20] Mickey G. Huson ym. ”Focused ion beam milling of carbon fibres”. *Materials Chemistry and Physics* 168 (marraskuu 2015), s. 193–200. ISSN: 0254-0584. DOI: 10.1016/j.matchemphys.2015.11.022. URL: http://www.sciencedirect.com/science/article/pii/S0254058415304491 (viitattu 11. 06. 2019).

[21] Dan Mao ym. ”Molecular Depth Profiling by Wedged Crater Beveling”. *Analytical Chemistry* 83.16 (2011). PMID: 21744861, s. 6410–6417. DOI: 10.1021/ac201502w. eprint: https://doi.org/10.1021/ac201502w. URL: https://doi.org/10.1021/ac201502w.

[22] Bede Pittenger ja Dalia Yablon. *Improving the Accuracy of Nanomechanical Measurements with Force-Curve-Based AFM Techniques*. Joulukuu 2017. DOI: 10.13140/RG.2.2.15272.67844.

Liitteet

A Tutkielmaani liittyvät ohjelmakoodit

Oma Python-koodini:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

Program reads a .pfc-file and extracts Force Curve data, calculating
various values and printing graphs from the data.
NOTE: Needs a program PointInPolygon.py to run properly.

TODO:   Optional: Limit measured values to improve graphs
        Optional: Optimization For Speed
        Add/Update Comments

@author: Niklas Valjakka
@version: 20220424

"""
# System imports

import struct
import collections
import time
import math
import numpy as np
import scipy as sp
from scipy import stats
import pylab as pl
import matplotlib.pyplot as plt
import os
import errno
```

```
from numpy.polynomial import polynomial

# Local source tree imports

from PointInPolygon import wn_PnPoly

# Constants

DEF_VALUE = 0.0  # Default value if errors encountered
# Data limit to differentiate two different versions (before 1.1.2019 and after 31.12.2018)
DATA_LIMIT = 2**10
CMAX_INDEX = 128
MAX_DISTANCE = 10**-9 # Used in functions to prevent dividing by zero
KEV_CONVERSION = 6.241509 / 1000 # Conversion factor from (nN*nm) to keV
PI_CONSTANT = math.pi
SIN_FUNC = math.sin
F_VALUES = 256 # Number of force data points in a Force Curve
BG_REMOVAL_DEGREES = [2,2] # Degrees for polyfit2d-function for background removal
ZLEVEL_START = 5 # The index of the point where the calculations for the z-level begin
ZLEVEL_END = 65 # The index of the point where the calculations for the z-level end
SLOPE_START_OFFSET = 4
SLOPE_END_OFFSET = 5
# Typical Force Fit Boundary values.
# Values taken from Dimension Icon User Guide
SLOPE_FIT_START = 0.90
SLOPE_FIT_END = 0.30
DISSIPATION_THRESHOLD = 25
# Radius of the AFM-tip in use (RTESPA525). Sourced from:
# https://www.brukerafmprobes.com/p-3915-rtespa-525.aspx
R_TIP = 8
HISTOGRAM_BINS = 50 # Number of bins in the histograms
```

```
def main():
    """
    The main program

    Returns
    -------
    None

    """
    # Starts a timer to test
    #start = time.time()

    # READING THE DATA FROM THE FILE

    # The user is asked for the name of a .pfc-file
    filename = input("File to read: ")
    # The file is located in system and the file path is returned
    filepath = find(filename,"..")
    filesize = os.path.getsize(filepath) # Tutkittavan tiedoston koko
    # Reads the header of the file, returns necessary info
    header_data = read_header(filepath)
    # Reads topography & Force Curve data
    topography_data_bin,fcurve_data_bin = read_height_and_fc_data(filepath,filesize,header_data[0])

    # Make the save folders, if necessary.
    savepath = filepath[:-4] + "_File/";
    make_folder(savepath)
    savepath_fc = savepath + 'Force_Curves/'
    savepath_me = savepath + 'FC_mean/'
```

```
    make_folder(savepath_fc)
    make_folder(savepath_me)

    # Change binary height data to float
    if (filesize/(header_data[0]**2) > DATA_LIMIT):
        topography_data_iter = struct.iter_unpack("i", topography_data_bin)
    else:
        topography_data_iter = struct.iter_unpack("h", topography_data_bin)
    z_mult = header_data[5] * header_data[7] # Multiplication factor for height from [raw data] to [nm]
    topography_data = z_mult * np.array(list(topography_data_iter)) # Unit: [nm]

    # Plot the topography data
    plot_data(topography_data,header_data[0],savepath=savepath,data_type="Topography")
    # Ask the user if background is to be calculated & removed
    char = input("Remove background? (Useful for flat surfaces) [Y/N]: ")
    if (char == 'Y' or char == 'y'):
        topography_data = remove_background(topography_data,header_data[0])
        # Plot topography minus background
        plot_data(topography_data,header_data[0],savepath=savepath,data_type="Topography")

    # Change binary Force Curve data to float
    if (filesize/(header_data[0]**2) > DATA_LIMIT):
        fcurve_data_iter = struct.iter_unpack("i", fcurve_data_bin)
    else:
        fcurve_data_iter = struct.iter_unpack("h", fcurve_data_bin)
    # Change values to nN
    force_mult = header_data[4] * header_data[8] * header_data[9] # Conversion factor from [raw data] to
[nN]
    fcurve_data = force_mult * np.array(list(fcurve_data_iter)) # Unit: nN
```

```
    # Specify areas, if necessary


    char = input("Do you want to specify an area? [Y/N]: ")
    if (char == 'Y' or char == 'y'):
        # Set up area specifying loop
        areaint = 1;
        area_points = [ask_for_coordinates()];
        savepath_fc = savepath + 'Force_Curves/Area_' + str(areaint) + '/';
        make_folder(savepath_fc)
        while True:
            char = input("Do you want to specify another area? [Y/N]: ")
            if not (char == 'Y' or char == 'y'):
                # Done asking points -> Plot area, end loop

plot_data(topography_data,header_data[0],area_points,savepath=savepath,data_type="Topography_Areas
")
                break
            else:
                area_points.append(ask_for_coordinates())
                areaint += 1
                savepath_fc = savepath + 'Force_Curves/Area_' + str(areaint) + '/';
                make_folder(savepath_fc)
    else:
        area_points = None;


    # Initialize result arrays
    adh_arr = np.zeros(header_data[0]**2)
    defor_arr = np.zeros(header_data[0]**2)
    pfor_arr = np.zeros(header_data[0]**2)
    diss_arr = np.zeros(header_data[0]**2)
    slope_arr = np.zeros(header_data[0]**2)
    #dmt_arr = np.zeros(header_data[0]**2)
```

```
# ITERATE THROUGH EVERY FORCE CURVE, PUT VALUES IN ARRAYS

savepath_fc = savepath + 'Force_Curves/'
for i in range(header_data[0]**2):
    # Process a force curve and get values
    value_arr = process_force_curve(i,fcurve_data[i*F_VALUES:(i+1)*F_VALUES],
                                    header_data,area_points,savepath_fc)
    # Put values into appropriate arrays
    adh_arr[i] = value_arr[0]; defor_arr[i] = value_arr[1];
    pfor_arr[i] = value_arr[2]; diss_arr[i] = value_arr[3];
    slope_arr[i] = value_arr[4];
    #dmt_arr[i] = value_arr[5];


# PLOT DATA INTO HISTOGRAMS

if area_points is None:
    # If no areas are specified
    savepath_hg = savepath + 'Histograms/'
    make_folder(savepath_hg)
    plot_histogram(adh_arr,header_data[0],area_points,savepath_hg,"Adhesion","nN")
    plot_histogram(defor_arr,header_data[0],area_points,savepath_hg,"Deformation","nm")
    plot_histogram(pfor_arr,header_data[0],area_points,savepath_hg,"Peak_Force","nN")
    plot_histogram(diss_arr,header_data[0],area_points,savepath_hg,"Dissipation","keV")
    plot_histogram(slope_arr,header_data[0],area_points,savepath_hg,"Slope","N/m")
    #plot_histogram(dmt_arr,header_data[0],area_points,savepath_hg,"Modulus")
else:
    # For every area specified
    for i in range(len(area_points)):
        savepath_hg = savepath + 'Histograms/Area_' + str(i + 1) + '/'
```

```
        make_folder(savepath_hg)
        plot_histogram(adh_arr,header_data[0],area_points[i],savepath_hg,"Adhesion","nN")
        plot_histogram(defor_arr,header_data[0],area_points[i],savepath_hg,"Deformation","nm")
        plot_histogram(pfor_arr,header_data[0],area_points[i],savepath_hg,"Peak_Force","nN")
        plot_histogram(diss_arr,header_data[0],area_points[i],savepath_hg,"Dissipation","keV")
        plot_histogram(slope_arr,header_data[0],area_points[i],savepath_hg,"Slope","N/m")
        #plot_histogram(dmt_arr,header_data[0],area_points[i],savepath_hg,"Modulus")

# MAP DATA TO IMAGES

savepath_pl = savepath + 'Plots/'
make_folder(savepath_pl)
plot_data(adh_arr,header_data[0],area_points,savepath_pl,"Adhesion")
plot_data(defor_arr,header_data[0],area_points,savepath_pl,"Deformation")
plot_data(pfor_arr,header_data[0],area_points,savepath_pl,"Peak_Force")
plot_data(diss_arr,header_data[0],area_points,savepath_pl,"Dissipation")
plot_data(slope_arr,header_data[0],area_points,savepath_pl,"Slope")
#plot_data(dmt_arr,header_data[0],area_points,savepath_pl,"Modulus")

# Plot mean/median curves in the y-direction

if area_points is None:
    savepath_ot = savepath + 'Other_Files/'
    make_folder(savepath_ot)
    plot_mean_values(adh_arr,header_data[0],area_points,savepath_ot,"Adhesion")
    plot_mean_values(defor_arr,header_data[0],area_points,savepath_ot,"Deformation")
    plot_mean_values(pfor_arr,header_data[0],area_points,savepath_ot,"Peak_Force")
    plot_mean_values(diss_arr,header_data[0],area_points,savepath_ot,"Dissipation")
    plot_mean_values(slope_arr,header_data[0],area_points,savepath_ot,"Slope")
    #plot_mean_values(dmt_arr,header_data[0],area_points,savepath_ot,"Modulus")
else:
```

```
    for i in range(len(area_points)):
        savepath_ot = savepath + 'Other_Files/Area_' + str(i + 1) + '/'
        make_folder(savepath_ot)
        plot_mean_values(adh_arr,header_data[0],area_points[i],savepath_ot,"Adhesion")
        plot_mean_values(defor_arr,header_data[0],area_points[i],savepath_ot,"Deformation")
        plot_mean_values(pfor_arr,header_data[0],area_points[i],savepath_ot,"Peak_Force")
        plot_mean_values(diss_arr,header_data[0],area_points[i],savepath_ot,"Dissipation")
        plot_mean_values(slope_arr,header_data[0],area_points[i],savepath_ot,"Slope")
        #plot_mean_values(dmt_arr,header_data[0],area_points,savepath_ot,"Modulus")

# PLOT DATA TO HEIGHT

savepath_ot = savepath + 'Other_Files/'
height_to_value(adh_arr,topography_data,savepath_ot,"Adhesion")
height_to_value(defor_arr,topography_data,savepath_ot,"Deformation")
height_to_value(pfor_arr,topography_data,savepath_ot,"Peak_Force")
height_to_value(diss_arr,topography_data,savepath_ot,"Dissipation")
height_to_value(slope_arr,topography_data,savepath_ot,"Slope")
#height_to_value(dmt_arr,topography_data,savepath_ot,"Modulus")

# Make a Force Curve from the mean of the values of FCs in the area
if area_points is not None:
    for i in range(1,len(area_points)+1):
        savepath_fc = savepath + 'Force_Curves/Area_' + str(i) + '/'
        savepath_me = savepath + 'FC_mean/Area_' + str(i) + '/'
        make_folder(savepath_me)
        plot_mean_curve(savepath_fc,savepath_me,header_data[11],[header_data[12],header_data[13]])

# Ends the timer
#end = time.time()
#print("Total time: %9.5f" % (end - start))
```

```
def find(name,path):
    """
    Returns a file path to the specified file

    Parameters
    ----------
    name : A name of the file that is searched.
    path : A filepath from which onward the file is searched.

    Returns
    -------
    string
        Filepath to the file.

    """

    result = []
    for root, dirs, files in os.walk(path):
        if name in files:
            result.append(os.path.join(root, name))

    # Raise Error if no file found
    if (len(result) == 0):
        raise NameError("Specified file could not be found!")
    # List multiple results, ask user to choose
    elif (len(result) > 1):
        print("Multiple files with same name found, please pick a correct file with a number");
        for i in range(len(result)):
            print("%i %s" % (i,result[i]))
```

```
        print("Please give the number of the file you want: ")
        time.sleep(1) # Separates print and input-commands to avoid possible errors
        file_num = int(input())
        return result[file_num]
    # Only one result found
    else:
        return result[0]



def make_folder(savepath):
    """
    Make a folder in the specified path

    Parameters
    ----------
    savepath : A path to where to make the folder.

    Returns
    -------
    None.

    """
    try:
        os.makedirs(savepath)
    except OSError as e:
        if e.errno != errno.EEXIST:
            raise



def read_header(file_path):
    """
```

Read important data from header

Information about file format:

http://nanophys.kth.se/nanophys/facilities/nfl/afm/icon/bruker-help/Content/Cover%20page.htm

Parameters

---

file_path : The path to the file which is read.

Returns

An array containing the necessary data, in order:

[Data resolution,

Sync Distance QNM,

Peak Force Amplitude,

Frequency,

Sens. DeflSens,

ZSensSens,

Soft HarmoniX Setpoint,

Z Scale Z (Height scaling factor),

Spring Constant,

Z Scale Defl. (Z Deflection scaling factor),

Peak Force Setpoint (Not in header, calculated),

Deformation Fit Region

Maximum Force Fit Boundary,

Minimum Force Fit Boundary].

"""

Open specified file in read-mode

with open(file_path, 'rb') as f:

```
header_data = [] # Initialize array of header rows
char_num = 0 # Initialize character size
# Iterate through every line in a header
while True:
    # Read next line
    # (split-command removes linebreak at the end of the row)
    line = f.readline().split(b'\r\n')[0]
    # End loop if line begins with byte "x1a"
    # (Special character that specifies the end of the header)
    if line.startswith(b'\x1a'):
        break
    # Otherwise add row to the header_data -array, and update character size
    else:
        header_data.append(line)
        char_num = char_num+len(line)+2 # +2 comes from the removed linebreak
print("Number of characters: " + str(char_num)) # Print the number of characters
# Get the relevant values from the header
for i in range(len(header_data)):
    linestr = str(header_data[i])
    # print(str(i) + " " + linestr[3:-1])
    # Print rows by number. Unnecessary "b'\" ja "'" are removed
    # from the beginning and the end of the row,
    # so that rows would match ones from UNIX more-command.

    # Data resolution value
    if (linestr[3:-1].find("\Lines") > -1):
        resolution = int(get_value_from_header(linestr[3:-1]))
        print("Resolution = %3.1i" % (resolution))

    # Sync Distance value
    elif (linestr[3:-1].find("\Sync Distance QNM:") > -1):
```

```
    sdqnm = get_value_from_header(linestr[3:-1])
    print("Sync Distance QNM = %i" % (sdqnm))

# Peak Force Amplitude value
elif (linestr[3:-1].find("\Peak Force Amplitude:") > -1):
    pfa = get_value_from_header(linestr[3:-1])
    print("Peak Force Amplitude = %4.1f" % (pfa))

# PFT Frequency value
elif (linestr[3:-1].find("\PFT Freq:") > -1):
    freq = 1000 * get_value_from_header(linestr[3:-1])
    print("Frequency = %5.1f" % (freq))

# Deflection Sens. value
elif (linestr[3:-1].find("\@Sens. DeflSens") > -1):
    deflsens = get_value_from_header(linestr[3:-1])
    print("Sens. DeflSens = %3.1f" % (deflsens))

# Z sens. value
elif (linestr[3:-1].find("\@Sens. ZsensSens") > -1):
    zsens = get_value_from_header(linestr[3:-1])
    print("ZSensSens = %7.4f" % (zsens))

# Soft HarmoniX Setpoint value
elif (linestr[3:-1].find("\@2:SoftHarmoniXSetpoint:") > -1):
    shxsp = get_value_from_header(linestr[3:-1])
    print("SHXSP = %8.7f" % (shxsp))

# Z Scale value
elif (linestr[3:-1].find("\@2:Z scale:") > -1):
    istart = linestr.find('(') + 1
```

```
    iend = linestr.find(')')
    zscalez = get_value_from_header(linestr[istart:iend])
    print("ZScaleZ = %13.12f" % (zscalez))

# Spring Constant value
elif (linestr[3:-1].find("\Spring Constant:") > -1):
    k = get_value_from_header(linestr[3:-1])
    print("Spring Constant = %4.1f" % (k))

# Deformation Fit Region value
elif (linestr[3:-1].find("\Deformation Fit Region:") > -1):
    deformation_fit = get_value_from_header(linestr[3:-1])
    print("Deformation Fit Region = %3.2f" % (deformation_fit))

# Z scale value
elif (linestr[3:-1].find("\@4:Z scale:") > -1):
    istart = linestr.find('(') + 1
    iend = linestr.find(')')
    zscaledefl = get_value_from_header(linestr[istart:iend])
    print("ZScaleDefl = %13.11f" % (zscaledefl))

# Maximum Force Fit Boundary Value
elif (linestr[3:-1].find("\Max Force Fit Boundary:") > -1):
    FFB_max = get_value_from_header(linestr[3:-1]) / 100.0
    print("Maximum Force Fit Boundary = %3.2f" % (FFB_max))

# Minimum Force Fit Boundary Value
elif (linestr[3:-1].find("\Min Force Fit Boundary:") > -1):
    FFB_min = get_value_from_header(linestr[3:-1]) / 100.0
    print("Minimum Force Fit Boundary = %3.2f" % (FFB_min))
```

```
    # Calculate the Peak Force Setpoint (unit: [nN])
    pfs = shxsp * k * deflsens
    print("Peak Force Setpoint = %5.1f" % (pfs))
    # Gather the data into a single array
    header_info = [resolution,sdqnm,pfa,freq,deflsens,zsens,shxsp,zscalez,k,
                   zscaledefl,pfs,deformation_fit,FFB_max,FFB_min]
    return header_info


def get_value_from_header(header_string,default=DEF_VALUE):
    """
    Read a value from a header line

    Parameters
    ----------
    header_string : A string line to read.
    default: A value given in a case no data is read

    Returns
    -------
    float
       Value from header string.

    """
    # Define a boolean
    got_value = False
    # Split to substrings, go over them all
    for substr in header_string.split():
       try:
          # Try to get a value from substring
```

```
            retval = float(substr)
            got_value = True
        except ValueError:
            # No luck, next substring
            pass
    if got_value:
        # Program succeessful, return value
        return retval
    else:
        # Program failure, return default
        print("Error! Data could not be read! Return default value %13.9f" % (default))
        return default


def read_height_and_fc_data(file_path,file_size,resolution):
    """
    Read height and Force Curve data from a .pfc -file

    Parameters
    ----------
    file_path : A path to a .pfc file.
    file_size : The size of the .pcf file
    resolution : A resolution of the data.

    Returns
    -------
    height_data: Topography data in a binary array.
    fc_data: Force Curve data in a binary array.

    """
```

```
    # Check for the .pfc-file version and choose imagesize & datasize accordingly
    if (file_size/(resolution**2) > DATA_LIMIT):
        imagesize = (resolution**2) * 4 # Size of height data in bytes
        datasize = (resolution**2) * 2 * (2**7) * 4 # Size of force curve data in bytes
    else:
        imagesize = (resolution**2) * 2 # Size of height data in bytes
        datasize = (resolution**2) * 2 * (2**7) * 2 # Size of force curve data in bytes
    # Get height and force curve data using values above
    with open(file_path, 'rb') as f:
        f.seek(file_size - datasize - imagesize) # Skip to the start of FC data
        height_data = np.fromfile(f,'B',imagesize) # Read Height Data
        fc_data = np.fromfile(f,'B',datasize) # Read FC Data

    return height_data,fc_data



def plot_data(data,resolution,areas=None,savepath=None,data_type=None):
    """
    Plot data in a square (resolution**2) area

    Parameters
    ----------
    data : Data to be plotted. (NOTE: Has to be an array of length resolution**2)
    resolution : Side of a square plot area.
    area : Area to be focused on
    savepath : Path to the save file
    data_type : The type of data to be shown on the title

    Returns
    -------
```

```
None.

"""
# Reshape data to match the area
data_array = data.reshape(resolution,resolution)
# Plot the figure
fig = plt.figure()
ax = plt.gca()

# Plot the area, if any given
if areas is not None:
    for area in areas:
        for i in range(len(area)-1):
            xplot = [area[i].x,area[i+1].x]
            yplot = [area[i].y,area[i+1].y]
            ax.plot(xplot,yplot,'-r')

imgplot = ax.imshow(data_array, origin='lower')
fig.colorbar(imgplot, ax=ax)
imgplot.set_cmap('nipy_spectral')

# Speficy title and axis labels
if data_type is not None:
    plt.title('2D map for ' + data_type)
else:
    plt.title('2D map')
plt.xlabel('Pixel')
plt.ylabel('Pixel')

# Save figure if save path given
if savepath is not None:
```

```
        savefile = savepath + data_type + '_plot.png'
        plt.savefig(savefile, dpi=300, bbox_inches='tight')

    # Show figure
    plt.show()


def ask_for_coordinates():
    """
    Ask user for coordinate points. Returns given points

    Returns
    -------
    Point array
        An array containing area edge points.

    """

    # Asks for the first coordinate
    print("Please give the first coordinate in form ""x y"", where x,y stand for pixels: ")
    time.sleep(1) # Separates print and input-commands to avoid possible errors
    first_point = [int(x) for x in input().split()]
    area = [Point(first_point[0],first_point[1])]
    while True:
        # Keep asking points until first one is given again
        print("Please give the next coordinate, or retype the first coordinate to close area: ")
        time.sleep(1) # Separates print and input-commands to avoid possible errors
        next_point = [int(x) for x in input().split()]
        area.append(Point(next_point[0],next_point[1]))
        if (next_point[0] == first_point[0] and next_point[1] == first_point[1]):
            # First point given again -> Return Points
```

```
        return area


def remove_background(data,resolution):
    """
    Remove background from an image by a polynomial fit

    Parameters
    ----------
    data : Data points from which background
    resolution : Resolution of data.

    Returns
    -------
    data_modified : Data with background removed.

    """
    xFit = np.linspace(0, 500, resolution)
    yFit = np.linspace(0, 500, resolution)
    zFit = [[data[j*resolution+i] for i in range(resolution)] for j in range(resolution)]
    X = np.array(np.meshgrid(xFit,yFit))
    c = polyfit2d(X[0], X[1], zFit, BG_REMOVAL_DEGREES)
    f = polynomial.polyval2d(X[0], X[1], c)
    data_modified = np.array([(zFit[j][i]-f[j][i]) for j in range(len(f)) for i in range(len(f[0]))])
    #f = np.resize(f,(1,len(dataImMod)))
    return data_modified


def polyfit2d(x, y, f, deg):
    """
    Program fits a polynomial to the data
```

```
    Parameters
    ----------
    x : X-coordinates of the data points.
    y : Y-coordinates of the data points.
    f : Data that polynomial is fitted to.
    deg : The degrees of the fitting polynomial in a vector [xdeg, ydeg].

    Returns
    -------
    TYPE
        A polynomial fit to the data.

    """
    x = np.asarray(x)
    y = np.asarray(y)
    f = np.asarray(f)
    deg = np.asarray(deg)
    vander = polynomial.polyvander2d(x, y, deg)
    vander = vander.reshape((-1,vander.shape[-1]))
    f = f.reshape((vander.shape[0],))
    c = np.linalg.lstsq(vander, f, rcond=-1)[0]
    return c.reshape(deg+1)


def process_force_curve(i,data,header_data,areas,savepath):
    """
    Process raw floating point data into a force curve

    Parameters
    ----------
```

```
    i : The pixel in question.
    data : Force data on pixel i.
    header_data : Array containing useful array data of the file.
    area : An area from which curve curves are pulled from.
    savepath : The path to the save file.

    Returns
    -------
    values : Useful values derived from force curves.

    """
    # Begin by correcting the force values and creating distance values
    # Data is currently disorganized and contains "trash bytes".
    # Data needs to be properly organized
    force_arr,dist_arr = create_curve_points(data,CMAX_INDEX - 1 -
int(round(header_data[1])),header_data)
    ffb = [header_data[12],header_data[13]]
    values = get_values(force_arr,dist_arr,header_data[11],ffb)

    # Plot the force curves in the specified area(s).
    if areas is not None:
        for j in range(len(areas)):
            x,y = pixel2point(i,header_data[0]) # Transform index i to pixel coordinates [x,y]
            if (wn_PnPoly(Point(x,y),areas[j],len(areas[j])-1) != 0):
                savepath_full = savepath + 'Area_' + str(j + 1) + '/';
                save_fc_data(force_arr,dist_arr,[x,y],savepath_full)
    return values



def pixel2point(pixel,res):
    """
```

```
    Transform a pixel placement into a placement on a 2D map

    Parameters
    ----------
    pixel : A pixel value.
    res : Measurement resolution.

    Returns
    -------
    x : x-position of the pixel point.
    y : y-position of the pixel point.

    """
    x = int(pixel%res)
    y = int(round((pixel-x)/res))
    return x,y


def create_curve_points(data,cutoff_point,header_data):
    """
    Create force curve points from data.
    Rearranges force data in a properly order.
    (Required as the raw data in a file is not organized as-is)

    Parameters
    ----------
    data : Force values of a force curve.
    cutoff_point : Index of the point where the value is cut-off.
    header_data : Data from header

    Returns
```

```
    -------
    y : A force data organized in a way that it presesnts a force curve.
    x : A distance values to match the force values

    """
    # Rearrange force values
    # NOTE: data[127] is not a valid data point.
    # Instead, data[256-cutoff_point-2], which corresponds
    # to the beginning of the curve, is used twice instead
    force_arr = np.array([(data[i]) for i in range(256-cutoff_point-2,256,1)])
    force_arr = np.append(force_arr,[(data[i]) for i in range(126,-1,-1)])
    force_arr = np.append(force_arr,[(data[i]) for i in range(128,256-cutoff_point-1,1)])

    # Define the time steps
    t = np.linspace(0,1.0/header_data[3],len(force_arr))
    # Transform x-axis from z-piezo distance
    x_0 = [header_data[2]*(1-SIN_FUNC(2*PI_CONSTANT*header_data[3]*(t[i])-(PI_CONSTANT/2))) for i in
range(len(t))]

    # Correct force values in data so that "tail" is near zero
    zerolevel = np.median(force_arr[ZLEVEL_START:ZLEVEL_END])
    y = force_arr - zerolevel;
    defl_max = header_data[10]/header_data[8]
    # Correct z-piezo distance to true distance by taking account cantilever movement
    x = [i-defl_max+(j/header_data[8]) for (i,j) in zip(x_0,y)]

    return y,x


def get_values(force_arr,dist_arr,deformation_fit,ffb):
    """
```

```
Get values from force curves

Parameters
----------
force_arr : Force values of the force curve points.
dist_arr : Distance values of the force curve points.
deformation_fit : Deformation Fit Region value (from header)
ffb : Force fit boundary values

Returns
-------
list
    A list of values calculated from the force curve data.

"""

global x_ind
global adhes_val

# Calculate peak force value (Unit: [nN])
peakforce_val = max(force_arr)
peakforce_val_index = np.where(force_arr == peakforce_val)[0][0]

# Calculate adhesion value (Unit: [nN])
adhes_val = min(force_arr)
adhes_val_index = np.where(force_arr == adhes_val)[0][0]
adhes_val_abs = abs(adhes_val)

# Calculate deformation value (Unit: [nm])

# Calculate first an offset value on distance (Based on Deformation Fit -value)
```

```
    defor_val = calc_deformation(force_arr,dist_arr,deformation_fit,peakforce_val)

    # Calculate dissipation value (Unit: [keV])
    diss_val = calc_dissipation(force_arr,dist_arr)

    # Calculate slope value (Unit: [N/m])
    slope_val = calc_slope(force_arr,dist_arr,peakforce_val_index,adhes_val_index,ffb)

    #dmt_val = calc_dmt(force_arr,dist_arr,adhes_val_index)

    #return [adhes_val_abs,defor_val,peakforce_val,diss_val,slope_val,dmt_val]
    return [adhes_val_abs,defor_val,peakforce_val,diss_val,slope_val]


def calc_deformation(y,x,def_fit,pf_val,default=DEF_VALUE):
    """
    Calculate deformation value of the force curve.
    (Deformation defined as the distance from the
     Deformation Force Level position to the peak interaction force position)

    Calculated value is not the full deformation value, but rather
    the Deformation Fit Region value is used.
    Same value is used by Bruker to reduce baseline noise.

    Unit: [nm]

    Parameters
    ----------
    y : Force values of force curve points.
    x : Distance values of force curve points.
    def_fit: Deformation Fit Region value
```

```
pf_val : Value of Peak Force
default : The default value returned in case of error.

Returns
-------
float
   The deformation value of the force curve.

"""
# Initialize variables to control while-loop
points_above = True; find_value = True;
#i = np.where(y == pf_val)[0][0]
i = np.where(x == min(x))[0][0]
j = i
cut_off = (1.00 - def_fit) * pf_val
while points_above:
   if j < 0:
      # An end was reached before a suitable index j was found!
      find_value = False
      points_above = False
   elif (y[j] < cut_off and y[j] > y[j-1]):
      # A Suitable index j was found: Loop can be ended
      # Calculate Deformation using secant method
      points_above = False
      y_modified = y - ((1.00 - def_fit) * pf_val)
      # In case y-values are too close
      if (abs(y_modified[j]-y_modified[j+1]) < MAX_DISTANCE):
         x_end = (x[j]+x[j+1])/2.0;
      else:
         x_end = x[j] - y_modified[j] * (x[j] - x[j+1])/(y_modified[j] - y_modified[j+1])
   else:
```

```
            # Step to the next index j
            j = j-1
    if not find_value:
        # Value not Found -> Error, return DEFAULT
        return default
    else:
        # Return deformation value
        return x_end - x[i]


def calc_dissipation(y,x):
    """
    Calculate the dissipation value of the curve, defined as the
    area between approach and retract curves.

    Unit: [keV]

    Parameters
    ----------
    y : Force values of force curve points.
    x : Distance values of force curve points.

    Returns
    -------
    Float
        The dissipation value of the force curve.

    """
    # Set a start index and index range for the analysis
    start_index = np.where(x == min(x))[0][0]
```

```
    index_range = min(255-start_index,start_index-1)
    ret_val_nm = num_integral(y,x,start_index,index_range)
    ret_val = KEV_CONVERSION * ret_val_nm
    return max(0,ret_val)


def num_integral(y,x,i_start,i_range,default=DEF_VALUE):
    """
    Calculate the area under the FC Extend and Retract -curves numerically

    Parameters
    ----------
    y : y-coordinates of the FC points.
    x : x-coordinates of the FC points.
    i_start : Start index of integral.
    i_end : End index of integral.

    Returns
    -------
    area : The area under the FC.

    """
    # Set index points, set total area to 0.0
    j0 = i_start; j1 = i_start - 1; j2 = i_start + 1; area = 0.0;
    while True:
        # Test whether the en index has been reached for either of the subindeces
        if ( (j2 - i_start) == i_range or (i_start - j1) == i_range):
            return default
        elif (y[j1] < y[j2] and (i_start - j1) >= DISSIPATION_THRESHOLD and (j2 - i_start) >=
DISSIPATION_THRESHOLD):
            # End reached: Break loop
```

```
            break
        else:
            # End reached: Break loop
            # End not reached: Calculate Tri. A and add to total
            area = area + calc_triangle([x[j0],y[j0]],[x[j1],y[j1]],[x[j2],y[j2]])
            # Move the indeces to the closest point in either direction
            j_apu = min(x[j1-1],x[j2+1])
            if j_apu == x[j1-1]:
                j0 = j1
                j1 = j1 - 1
            else:
                j0 = j2
                j2 = j2 + 1
    # Return integral or total area under the
    return area


def calc_triangle(x0,x1,x2):
    """
    Calculate the area of a triangle using its vertices

    Parameters
    ----------
    x0 : First vertice of a triangle
    x1 : Second vertice of a triangle
    x2 : Third vertice of a triangle

    Returns
    -------
    TYPE
        Area of a triangle.
```

```
    """
    return abs((1/2) * ( x0[0]*(x1[1]-x2[1]) + x1[0]*(x2[1]-x0[1]) + x2[0]*(x0[1]-x1[1]) ) )


def calc_slope(y,x,start_index,end_index,ffb,default=DEF_VALUE):
    """
    Calculate the slope of a force curve

    Unit : [N/m]

    Parameters
    ----------
    y : Force values of the force curve.
    x : Distance values of the force curve.
    start_index : The index of the maximum of the force curve.
    end_index : The index of the minimum of the force curve.
    ffb: Force Fit Boundary values
    default : The default value returned in case of error.

    Returns
    -------
    float
       The value of the slope.

    """
    # Set maximum and minimum force values
    y_pf = y[start_index]; y_adh = y[end_index]

    # Set maximum and minimum force values for slope value analysis
    #y_max = calc_FFB(ffb[0],y_adh,y_pf)
```

```
#y_min = calc_FFB(ffb[1],y_adh,y_pf)
y_max = calc_FFB(SLOPE_FIT_START,y_adh,y_pf)
y_min = calc_FFB(SLOPE_FIT_END,y_adh,y_pf)

# Calculate start index of the slope scan
# (PeakForceValue * DeformationFitRegion (value from header))
i = start_index
while True:
    if (i == 256):
        x0 = start_index
        break
    if (y[i] < y_max):
        x0 = i
        break
    else:
        i = i + 1

# Calculate end index of slope scan
# (AdhesionValue * SLOPE_FIT_END (custom value))
i = end_index
while True:
    if (y[i] > y_min):
        xn = i + 1
        break
    else:
        i = i - 1

# Fit a slope to the points [x[x0:xn],y[x0:xn]]
if (xn - x0 < 3):
    # Not enough points for an analysis -> Error, return default
    return default
```

```
    else:
        y_slope = y[x0:xn]
        x_slope = x[x0:xn]
        # Fit a 1-degree polynomial to the points
        slope,y0 = np.polyfit(x_slope,y_slope,1)
        if (slope > 0):
            # Slope should be negative, positive result -> Error, return default
            return default
        else:
            return slope


def calc_FFB(x,F_Adh,F_PF):
    """
    Calculates Force Fit Boundary value for multiplier x

    Parameters
    ----------
    x : A multiplier, 0 <= x <= 1
    F_Adh : Adhesion Force value
    F_PF : Peak Force value

    Returns
    -------
    Force Fit Boundary for multiplier x.

    """
    return F_Adh + (F_PF - F_Adh) * x


def calc_dmt(y,x,indMax,default=DEF_VALUE):
```

```
"""

Calculate the young modulus value in GPa using DMT model

(NOTE: NOT IN USE: WILL SIGNIFICANTLY SLOW DOWN THE PROGRAM & NEEDS PRESET VALUES
 TO WORK!)

Parameters
----------
y : Force values of the force curve.
x : Distance values of the force curve.
indMax : The index of the minimum point of the force curve
default : Default return value, if program encounters an error.

Returns
-------
float
   Value of calculated Young's Modulus.

"""
try:
   Arr = sp.optimize.curve_fit(myfunc,
                        x[CMAX_INDEX+SLOPE_START_OFFSET:indMax-SLOPE_END_OFFSET],
                        y[CMAX_INDEX+SLOPE_START_OFFSET:indMax-SLOPE_END_OFFSET],
                        p0=[100.0],bounds=([1.0],[300.0]))

except ValueError:
   #print("There's something wrong with the values! Check the program!")
   return default

except RuntimeError:
```

```
        #print("Optimal parameters not found! Check the program!")
        return default

    return Arr[0][0]


def myfunc(x,e):
    """Funktio DMT-fittauksen tekoon
        (Lähde: https://pubs.acs.org/doi/10.1021/la302706b)

    Parametrit:
    x -- Muuttujan arvo
    e -- (Redusoidun) Youngin Moduluksen arvo

    Palautus:
    Funktion arvo pisteessä x
    """

    return (4/3) * e * np.sqrt(R_TIP*(abs(x_ind-x)**3)) - adhes_val


def save_fc_data(y_arr,x_arr,point=None,savepath=None):
    """
    Save Force Curve data in a text file, ASCII format

    Parameters
    ----------
    y_arr : The y-coordinate values of the points
    x_arr : The x-coordinate values of the points.
    point : The location of the data point in pixel units.
    savepath : The path to the save file.
```

```
    Returns
    -------
    None.

    """

    if point is not None:
        file_name = savepath + 'Data_y_' + str(point[1]) + '_x_' + str(point[0]) + '.txt'
    else:
        file_name = savepath + '_Value_Curve.txt'
    with open(file_name,'w') as text_file:
        text_file.write("  i nm           nN \n")
        for i in range(len(x_arr)):
            text_file.write("%3i %12.9f %8.5f\n" % (i, x_arr[i], y_arr[i]))



def plot_histogram(data,resolution,area,savepath,data_type,data_unit):
    """
    Plot the calculated values in a histogram.

    Parameters
    ----------
    data : Data to be presented.
    resolution : The resolution of the data.
    area : The area given by user, None if none given.
    savepath : Path to the save folder.
    data_type : The type of data in a string.
    data_unit : The unit of the data in a string

    Returns
```

```
-------
None.

"""
# Remove data points outside the area
if area is not None:
   data = [data[i] for i in range(resolution**2)
         if (wn_PnPoly(Point(pixel2point(i,
               resolution)[0],pixel2point(i,resolution)[1]),area,len(area)-1) != 0)]

data_sorted = sorted(data)

fig = plt.figure()
ax = fig.add_subplot(111)
# Fit a gauss cirve to the data
fit = stats.norm.pdf(data_sorted, np.mean(data_sorted), np.std(data_sorted))
# Plot gauss curve and make a histogram
pl.plot(data_sorted,fit,'-o')
pl.hist(data_sorted,density=True,bins=HISTOGRAM_BINS)
# Convert values to a string
str_median = "Median: " + str(np.median(data_sorted))
str_mean = "Mean: " + str(np.mean(data_sorted))
str_sd = "SD: " + str(np.std(data_sorted))
iqr = np.percentile(data_sorted, 75) - np.percentile(data_sorted, 25)
str_iqr = "IQR: " + str(iqr)
str_count = "Count: " + str(len(data_sorted))
# Write mean values to a corner
ax.text(0.99, 0.99, str_median, horizontalalignment='right',
      verticalalignment='top', transform=ax.transAxes)
ax.text(0.99, 0.92, str_mean, horizontalalignment='right',
      verticalalignment='top', transform=ax.transAxes)
```

```
    ax.text(0.99, 0.85, str_sd, horizontalalignment='right',
            verticalalignment='top', transform=ax.transAxes)
    ax.text(0.99, 0.78, str_iqr, horizontalalignment='right',
            verticalalignment='top', transform=ax.transAxes)
    ax.text(0.99, 0.71, str_count, horizontalalignment='right',
            verticalalignment='top', transform=ax.transAxes)
    plt.xlabel(data_type + " [" + data_unit + "]")
    pl.title("Histogram for " + data_type)
    savefile = savepath + 'Histogram_' + data_type + '.png'
    pl.savefig(savefile, dpi=300, bbox_inches='tight')
    pl.show()
    # Write histogram data to a text file
    datafile = savepath + 'Histogram_data_' + data_type + '.txt'
    with open(datafile,"w") as text_file:
        for i in range(len(data_sorted)):
            text_file.write("%13.9f \n" % (data_sorted[i]))


def plot_mean_values(data,resolution,area,savepath,data_type):
    """
    Plot the mean and median values for every y-pixel in the area.

    Parameters
    ----------
    data : Array of values.
    resolution : Data resolution.
    area : Points of a polygon.
    savepath : Path to the save file.
    data_type : Value type.

    Returns
```

```
-------
None.

"""
# Reshape data array into map
data_array = data.reshape(resolution,resolution)
# Initialize arrays
x = []; data_means = []; data_medians = [];
# If area is defined
if area is not None:
   for i in range(resolution):
      # Gather values inside area
      row_insides = [data_array[i][j]
                  for j in range(resolution)
                  if (wn_PnPoly(Point(j,i),area,len(area)-1) != 0)]
      # Add a value from row_insides if not NULL
      if row_insides:
         x.append(i)
         # Calculate the mean and median, add to array
         row_mean = np.mean(row_insides)
         row_median = np.median(row_insides)
         data_means.append(row_mean);
         data_medians.append(row_median)
# If area is not defined
else:
   for i in range(resolution):
      row_insides = [data_array[i][j]
                  for j in range(resolution)]
      x.append(i)
      row_mean = np.mean(row_insides)
      row_median = np.median(row_insides)
```

```
        data_means.append(row_mean);
        data_medians.append(row_median)

    # Create a figure
    plt.figure()
    plt.plot(x,data_means,'or',x,data_medians,'ob')
    plt.xlabel("Y position")
    plt.ylabel(data_type)
    pl.title("Curves for mean (red) and median (blue)")
    savefile = savepath + 'Mean_Graphs_For_' + data_type + '.png'
    pl.savefig(savefile, dpi=300, bbox_inches='tight')
    plt.show()


def height_to_value(y,x,path,data_type):
    """
    Plots value vs. height graph

    Parameters
    ----------
    y : Force values of the force curve.
    x : Distance values of the force curve.
    path : Path to the save file.
    data_type : Value type.

    Returns
    -------
    None.

    """
    plt.figure()
```

```
    plt.plot(x,y,'or')
    plt.xlabel("Height")
    plt.ylabel(data_type)
    pl.title("Value vs. height data for " + data_type)
    savefile = path + data_type + '_Height.png'
    pl.savefig(savefile, dpi=300, bbox_inches='tight')
    plt.show()


def plot_mean_curve(curves_path,path,deformation_fit,ffb):
    """
    Plot a mean and median curves of every FC in an area.

    Parameters
    ----------
    curves_path : Path to the file with the force curves.
    path : Path to save file.
    deformation_fit : fitting area of deformation, needed to calculate values
                      in save_values
    ffb: Force Fit Boundary values

    Returns
    -------
    None.

    """

    # Get every force curve file
    folder_files = [f for f in os.listdir(curves_path) if f[-3:] == 'txt']
    # Initialize useful variables
    N = len(folder_files); M = 256; i= 0;
```

```
    # Initialize arrays to gather values
    xs = np.zeros([N,M]); ys = np.zeros([N,M])
    for file in folder_files:
        with open(curves_path + file, 'r') as f:
            lines = f.readlines()
            x0 = []; y0 = [];
            for x in lines[1:]:
                x0.append(float(x.split()[1])); # The x-values of a force curve
                y0.append(float(x.split()[2])); # The y-values of a force curve
            # Add the x- and y-positions of the curve to the total array
            xs[i] = x0; ys[i] = y0; i += 1;

    # Create mean/median force curves from data
    x1 = np.mean(xs, axis = 0); y1 = np.mean(ys, axis = 0)
    x2 = np.median(xs, axis = 0); y2 = np.median(ys, axis = 0)
    save_fc_data(y1,x1,savepath=path+"Mean");
    save_fc_data(y2,x2,savepath=path+"Median");
    save_values(y2,x2,deformation_fit,ffb,path)



def save_values(y,x,deformation_fit,ffb,savepath):
    """
    Save median force curve values to a file

    Parameters
    ----------
    y : Force values of the force curve.
    x : Distance values of the force curve.
    deformation_fit : fitting area of deformation
    ffb: Force Fit Boundary values
    savepath : Path to the save file.
```

```
    Returns
    -------
    None.

    """
    # Get values of a force curve
    values = get_values(y,x,deformation_fit,ffb)
    strings = ["Adhesion","Deformation","Peak Force","Dissipation","Slope"]
    # Write values to a file
    datafile = savepath + 'Median_FC_Data.txt'
    with open(datafile,"w") as text_file:
        for i in range(5):
            text_file.write("%s %13.9f\n" % (strings[i], values[i]))



if __name__ == "__main__":
    """
    Starts up the program
    """
    # Specify a Point, which is needed to use PointInPolygon.py
    Point = collections.namedtuple('Point', ['x','y'])
    # Start the main program
    main()
```

PointInPolygon-ohjelma (Tarvitaan yllä olevan koodin suorittamiseen):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Copyright 2000 softSurfer, 2012 Dan Sunday
# This code may be freely used and modified for any purpose
# providing that this copyright notice is included with it.
# SoftSurfer makes no warranty for this code, and cannot be held
# liable for any real or imagined damage resulting from its use.
# Users of this code must verify correctness for their application.



# a Point is defined by its coordinates {int x, y;}
#===================================================================



# isLeft(): tests if a point is Left|On|Right of an infinite line.
#    Input:  three points P0, P1, and P2
#    Return: >0 for P2 left of the line through P0 and P1
#            =0 for P2  on the line
#            <0 for P2  right of the line
#    See: Algorithm 1 "Area of Triangles and Polygons"
def isLeft( P0, P1, P2 ):

    return ( (P1.x - P0.x) * (P2.y - P0.y)
            - (P2.x -  P0.x) * (P1.y - P0.y) );


#===================================================================
```

```
# cn_PnPoly(): crossing number test for a point in a polygon
#      Input:   P = a point,
#               V[] = vertex points of a polygon V[n+1] with V[n]=V[0]
#      Return:  0 = outside, 1 = inside
# This code is patterned after [Franklin, 2000]
def cn_PnPoly( P, V, n ):

    cn = 0;    # the  crossing number counter

    # loop through all edges of the polygon
    for i in range(n):    # edge from V[i]  to V[i+1]
       if (((V[i].y <= P.y) and (V[i+1].y > P.y))     # an upward crossing
        or ((V[i].y > P.y) and (V[i+1].y <=  P.y))): # a downward crossing
            # compute  the actual edge-ray intersect x-coordinate
            vt = (float)(P.y  - V[i].y) / (V[i+1].y - V[i].y);
            if (P.x <  V[i].x + vt * (V[i+1].x - V[i].x)): # P.x < intersect
                cn += 1;   # a valid crossing of y=P.y right of P.x

    return (cn&1);    # 0 if even (out), and 1 if  odd (in)



#===================================================================


# wn_PnPoly(): winding number test for a point in a polygon
#      Input:   P = a point,
#               V[] = vertex points of a polygon V[n+1] with V[n]=V[0]//
#      Return:  wn = the winding number (=0 only when P is outside)

def wn_PnPoly( P, V, n ):
```

```
    wn = 0;    # the  winding number counter

    # loop through all edges of the polygon
    for i in range(n):    # edge from V[i] to  V[i+1]
        if (V[i].y <= P.y):          # start y <= P.y
            if (V[i+1].y  > P.y):      # an upward crossing
                if (isLeft( V[i], V[i+1], P) > 0):  # P left of  edge
                    wn += 1;           # have  a valid up intersect

        else:                        # start y > P.y (no test needed)
            if (V[i+1].y  <= P.y):     # a downward crossing
                if (isLeft( V[i], V[i+1], P) < 0):  # P right of  edge
                    wn -= 1;           # have  a valid down intersect

    return wn;

#===================================================================
```