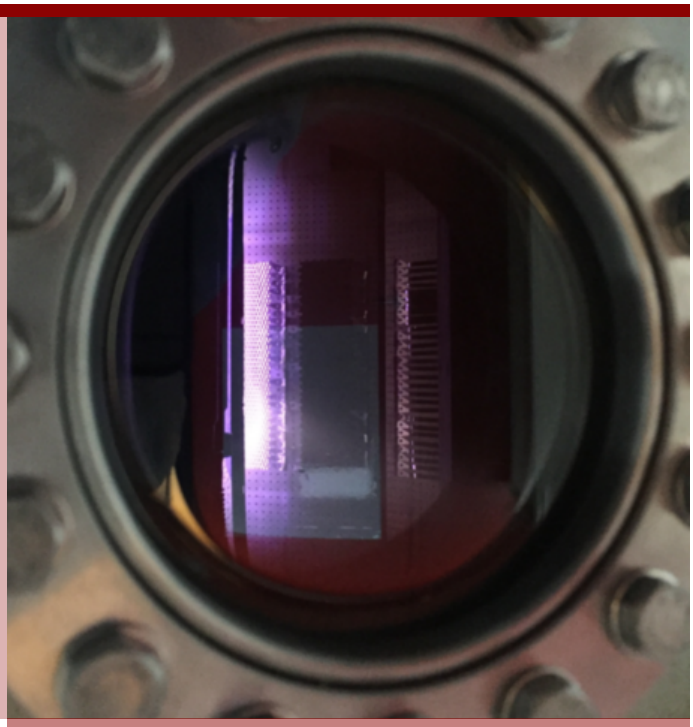
Development of Ni/B4C coatings for high energy X-ray telescopes using reactive magnetron sputtering

Master Thesis



Christoffer Raad Petersen
July 2017

DTU Space
National Space Institute

Development of Ni/B_4C coatings for high energy X-ray telescopes using reactive magnetron sputtering

By:
Christoffer Raad Petersen

Supervisor:
Senior Scientist Finn Erland Christensen
July 7, 2017

Abstract

In this thesis, the properties and behaviour of nickel (Ni) and boron carbide (B_4C) are studied to determine their usability and fitness for high energy X-ray mirrors. Specifically, the study comprises of three phases that structurally examine the impact of coating pressure and gas composition on both single layers of nickel and multilayers of nickel and boron carbide produced by direct current magnetron sputtering.

It is found that reactive sputtering of nickel have a negative effect on the material roughness and dynamic deposition rate, but a positive effect on the same properties of boron carbide.

For single layers of nickel the optimal conditions in terms of reflectance, roughness and dynamic deposition rate are achieved when the thin films are produced in pure argon at 2.5 [mTorr]. For multilayers of nickel and boron carbide the optimal conditions in terms of reflectance, roughness and dynamic deposition rate are achieved when produced in 80% argon and 20% nitrogen at 3.0 [mTorr].

Title:
Development of Ni/B_4C coatings for high energy X-ray telescopes using reactive magnetron sputtering.

Written by:
Christoffer Raad Petersen s093888@student.dtu.dk

DTU Supervisor:
Finn Erland Christensen finn@space.dtu.dk

DTU Space
Elektrovej, 327 & 328
2800 Kgs. Lyngby
Denmark

Project period:	02-01-2017 - 07-07-2017
ECTS:	30
Degree:	Master of Science and Engineering (MSc Eng)
Field:	Earth and Space Physics and Engineering, Space research
Classification:	Public
Edition:	1st edition
Remarks:	This report is submitted as partial fulfilment of the requirements for graduation in the above education at the Technical University of Denmark.

Contents

1 Project design and description

The work carried out during this project, aim to determine the properties and behaviour of nickel (Ni) and boron carbide (B_4C) as materials for high energy X-ray mirrors. To do so, single layer (SL) thin films and multilayer (ML) thin films, produced under different coating pressures and gas compositions are investigated.
All thin films for this project are fabricated in the multilayer laboratory at DTU Space. The films are all deposited on substrates of silicon (Si) by direct current (DC) magnetron sputtering and subsequently measured by 8.047 [keV] X-ray reflectomatry (XRR). The XRR measurements are completed at the XRR laboratory located at DTU Space.
Characterisation of the fabricated thin films are carried out by a developed MATLAB based program, that builds upon the IMD software.

The work is carried out in three phases. The first phase analyses thin films produced in pure argon gas at four pressures. To begin with SL's of nickel are investigated and the outcomes hereof are then used to produce ML's of nickel and boron carbide. Phase 1 are concluded with an overall analysis of both the SL's and ML's, which determines the pressure where the thin films exhibit the best performance.
The second phase studies thin films produced by reactive sputtering in a mixture of argon and nitrogen gas. At the optimal pressure found in phase one, the impact of different nitrogen gas fractions are studied.
Again, the study starts out by investigating SL's of nickel and then proceeds to ML's of nickel and boron carbide. The analysis in phase two end up by concluding for which gas fraction the SL's and ML's show the best characteristics.
In the last phase, phase three, thin films are studied at four pressures at the optimal gas fraction from phase 2. The pressures are the same as for phase one to enable direct comparisons. First SL's of nickel are studied and then ML's of nickel and boron carbide are look upon. Phase three is finished by concluding at which pressure the reactively produced thin films show the best performance.

2 X-ray theory

X-rays are electromagnetic waves with energies ranging from approximately 100 [eV] to 100 [keV]. By this, X-rays can be described by the wave equation, which in 3D is given by equation (2.1). In (2.1), ν_p denotes the phase velocity and is the rate at which the phase of the wave propagate through the medium. It is given and related to other properties by equation (2.2), where μ and ε is the permeability and the permittivity of the medium respectively, c is the speed of light in vacuum, n is the refractive index of the medium, λ is the wavelength of the wave, T is the period of the wave, ω is the angular frequency of the wave and k is the wave number.

$$\frac{\partial^2 \mathbf{\Phi}}{\partial t^2} = \nu_p^2 \nabla^2 \mathbf{\Phi} \tag{2.1}$$

$$\nu_p = \frac{1}{\sqrt{\mu\varepsilon}} = \frac{c}{n} = \frac{\lambda}{T} = \frac{\omega}{k} \tag{2.2}$$

If X-rays travel in vacuum where no charges are present the wave equation can be decomposed in terms of the electric and magnetic field. If so, equation (2.1) becomes as equation (2.3) and (2.4).

$$\frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t^2} - \nabla^2 \mathbf{E} = 0 \tag{2.3}$$

$$\frac{1}{c^2}\frac{\partial \mathbf{B}}{\partial t^2} - \nabla^2 \mathbf{B} = 0 \tag{2.4}$$

For a planar harmonic wave at the spatial position $\mathbf{r}$ at time t, the solution to equation (2.3) and (2.4), are as equation (2.5) and (2.6), where $\mathbf{E_0}$ and $\mathbf{B_0}$ are the amplitudes of the electric and the magnetic field and $\mathbf{k} = (k_x, k_y, k_z)$ is the wave vector.

$$\mathbf{E}(\mathbf{r}, t) = \mathbf{E_0} \exp(i(\mathbf{k} \cdot \mathbf{r} - \omega t)) \tag{2.5}$$

$$\mathbf{B}(\mathbf{r}, t) = \mathbf{B_0} \exp(i(\mathbf{k} \cdot \mathbf{r} - \omega t)) \tag{2.6}$$

2.1 Reflection and transmission from ideal dielectric boundary

When a monochromatic plane wave, with electric and magnetic field as described by equations (2.5) and (2.6), encounter a transition from one medium to another with an incident angle, θ_i, to the normal of the incident plane, it can either be reflected, refracted or a combination of the two. In Figure 1 such a situation is sketch, where the the reflected and transmitted wave subtend angle θ_r and θ_t respectively.

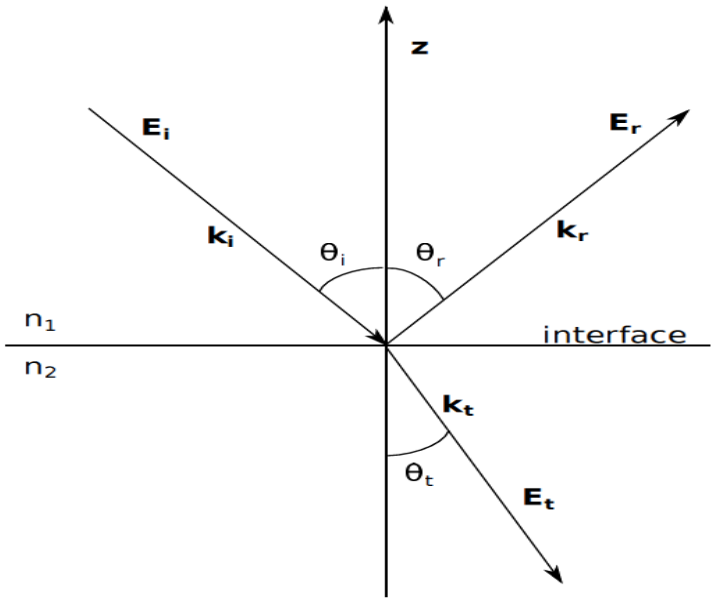


Figure 1: Sketch of interface between two mediums where light is reflected and transmitted.

At position $z = 0$, it is know from electrostatics that the components of the electric and magnetic field, which are tangent to the surface, must be continuous across the boundary [1, 2]. Thus the boundary condition of the electric and magnetic field becomes as below, and equation (2.11) can be set up to describe the relation between the magnetic field on both sides of the boundary.

$$E_{\parallel}^{above} = E_{\parallel}^{below} \tag{2.7}$$
$$B_{\parallel}^{above} = B_{\parallel}^{below} \tag{2.8}$$
$$\varepsilon_1 E_{\perp}^{above} = \varepsilon_2 E_{\perp}^{below} \tag{2.9}$$
$$B_{\perp}^{above} = B_{\perp}^{below} \tag{2.10}$$

$$\mathbf{B_i} \exp(i(\mathbf{k_i} \cdot \mathbf{r} - \omega t)) + \mathbf{B_r} \exp(i(\mathbf{k_r} \cdot \mathbf{r} - \omega t)) = \mathbf{B_t} \exp(i(\mathbf{k_t} \cdot \mathbf{r} - \omega t)) \tag{2.11}$$

Given that the magnetic field amplitude are constant, a solution to equation (2.11) can only be found when

$$\mathbf{k_i} \cdot \mathbf{r} = \mathbf{k_r} \cdot \mathbf{r} = \mathbf{k_t} \cdot \mathbf{r} \tag{2.12}$$

This implies that the incident, reflected and transmitted wave all are in the same plane, and further consideration can therefore be made by only considering the x-z plane, i.e. y equals 0.
From Figure 1, the equations in (2.13) can be constructed.

$$\sin(\theta_i) = \frac{k_{ix}}{k_i} \qquad \sin(\theta_r) = \frac{k_{rx}}{k_r} \qquad \sin(\theta_t) = \frac{k_{tx}}{k_t} \tag{2.13}$$

From equation (2.13), along with the knowledge from equation (2.12) and (2.2), the law of reflection and Snell's law can be derived as seen in equation (2.14) and (2.15) respectively.

$$\sin(\theta_i)k_i = \sin(\theta_r)k_r \quad \Leftrightarrow \quad \sin(\theta_i)\frac{\omega n_1}{c} = \sin(\theta_r)\frac{\omega n_1}{c} \quad \Leftrightarrow \quad \theta_i = \theta_r \tag{2.14}$$

$$\sin(\theta_i)k_i = \sin(\theta_r)k_t \quad \Leftrightarrow \quad \sin(\theta_i)\frac{\omega n_1}{c} = \sin(\theta_r)\frac{\omega n_2}{c} \quad \Leftrightarrow \quad \sin(\theta_i)n_1 = \sin(\theta_t)n_2 \tag{2.15}$$

2.2 Fresnel equations

If an electromagnetic wave encounters an idealised interface between to adjacent mediums, the reflected and transmitted amplitudes of the electric field are described by Fresnel's equations, seen in equation (2.16) to (2.19), where $\mathbf{n}$ is the complex index of refraction corresponding to the relevant medium and is defined as $\mathbf{n} = n + ik$. n is the refractive index and k is the extinctions coefficient. Equation (2.16) and (2.17) describe the situation of s-polarization where the electric field, $\mathbf{E}$, is perpendicular to the plane of incidence. Equation (2.18) and (2.19) describe the situation of p-polarization where the electric field, $\mathbf{E}$, is parallel to the plane of incidence [3].

$$\frac{|\mathbf{E_r}|}{|\mathbf{E_i}|} = \frac{\mathbf{n_1}\cos(\theta_i) - \mathbf{n_2}\cos(\theta_t)}{\mathbf{n_1}\cos(\theta_i) + \mathbf{n_2}\cos(\theta_t)} \equiv r^s \tag{2.16}$$

$$\frac{|\mathbf{E_t}|}{|\mathbf{E_i}|} = \frac{2\mathbf{n_1}\cos(\theta_i)}{\mathbf{n_1}\cos(\theta_i) + \mathbf{n_2}\cos(\theta_t)} \equiv t^s \tag{2.17}$$

$$\frac{|\mathbf{E_r}|}{|\mathbf{E_i}|} = \frac{\mathbf{n_1}\cos(\theta_t) - \mathbf{n_2}\cos(\theta_i)}{\mathbf{n_1}\cos(\theta_t) + \mathbf{n_2}\cos(\theta_i)} \equiv r^p \tag{2.18}$$

$$\frac{|\mathbf{E_t}|}{|\mathbf{E_i}|} = \frac{2\mathbf{n_1}\cos(\theta_i)}{\mathbf{n_1}\cos(\theta_t) + \mathbf{n_2}\cos(\theta_i)} \equiv t^p \tag{2.19}$$

2.3 Reflection and transmission from thin films

Considering reflection and refraction from thin films in reality, the minimum number of separated mediums involved are always three. There are the ambient medium, the actual medium of the film and the medium of the substrate. Because the medium of the film has a thin finite thickness, the arrangement permits some fraction of the incident wave to be reflected multiple times within the medium of the film. Each time the incoming wave encounter an interface between mediums some part is reflected and some part is refracted. In Figure 2 the principle is sketched.

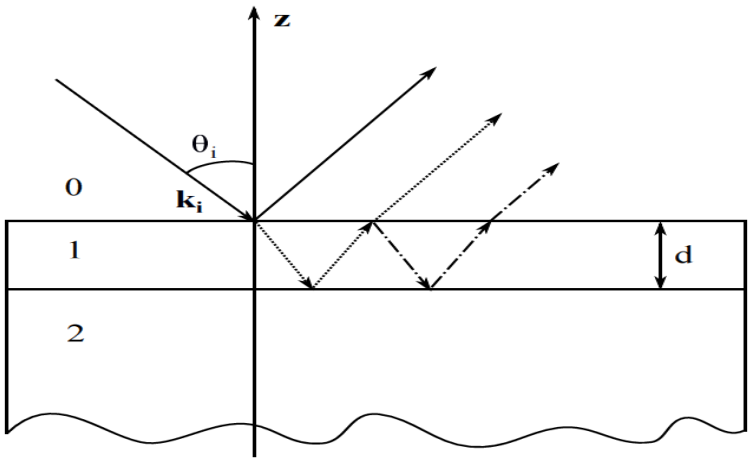


Figure 2: Relection from thin film with one layer and two interfaces. The ambient layer has the refractive index 0, the medium of the film has the refractive index 1 and the substrate has the refractive index 2. [4].

All parts of the wave, in Figure 2, that are reflected or transmitted away from the first interface in positive z direction, contribute to the total reflectance of the thin film, although the contributions becomes exponentially smaller with the number of internal reflections.
The net reflection, r_i, and transmission, t_i, from the thin film in Figure 2 is mathematically expressed as in equation (2.20) and (2.21) respectively, if $i = 0$, $j = \mathrm{i}{+}1$, $r_2 = 0$, r_{ij} and t_{ij} is calculated from the Fresnel equations, and β is as in (2.22).

$$r_i = \frac{r_{ij} + r_j \cdot \exp(2i\beta_i)}{1 + r_{ij} r_j \cdot \exp(2i\beta_i)} \tag{2.20}$$

$$t_i = \frac{t_{ij} + t_j \cdot \exp(2i\beta_i)}{1 + r_{ij} r_j \cdot \exp(2i\beta_i)} \tag{2.21}$$

$$\beta_i = \frac{2\pi d_i \mathbf{n_i} \cos(\theta_i)}{\lambda} \tag{2.22}$$

Even though, equation (2.20) and (2.21) in the above are used to describe the reflectance and transmittance from a SL they can just as well be used to describe a ML with N layers and N+1 interfaces. This is done through a recursive iteration process known as Paratt's recursive method [3, 5].

The reflectance and transmittance, which are measures of the energy reflected from or transmitted through the film, respectively are then given by equation (2.23) and (2.24) and the absorptance is approximated equation (2.25).

$$R = |r|^2 \tag{2.23}$$

$$T = \mathrm{Re}\left(\frac{\mathbf{n_s}\cos(\theta_s)}{\mathbf{n_a}\cos(\theta_a)}\right) \cdot |t|^2 \tag{2.24}$$

$$A = 1 - R - T \tag{2.25}$$

2.3.1 Interface imperfections

In reality a transition between mediums is newer completely perfect and this has to be taken into account when describing how an electromagnetic wave reflects and transmit through interfaces.
In order to account for the loss in specular reflection, two physical phenomena have to be considered. The first phenomena is the surface roughness, σ_r, see Figure 3 (a).
The surface roughness can be described as the profile elevation. Primarily the surface roughness rise due to the fact that a layer is deposited by sputtering. The sputter rate is not perfectly constant and uniform over the whole surface area, which results in variations of the surface elevation. Furthermore the a layer is always deposit upon a another medium which surface roughness transfer through deposited layer.
The second phenomena is diffuseness, σ_d. When individual layers are created in the sputtering process they are contaminated by atoms from the surroundings along with atoms from the adjacent layer. This give rise to a diffusion zone where a mix of atoms from the two adjacent mediums is found. A diffusion zone is sketch in Figure 3 (b). Because of the small scales of σ_r and σ_d they are difficult to distinguish from each other and are for practical reasons normally described together as $\sigma_r ms$, see equation (2.26).

$$\sigma_{rms} = \sqrt{\frac{\sigma_d^2 + \sigma_r^2}{2}} \tag{2.26}$$

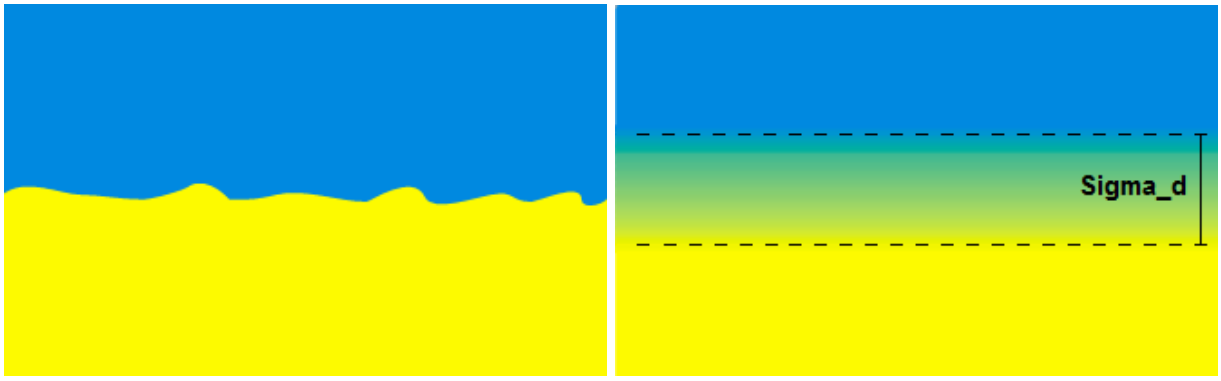


Figure 3: (a): Sketch of interface where only surface roughness are present between mediums. (b): Sketch of interface where only diffuse roughness are present between mediums.

When modelling the specular reflectance from an interface, the surface roughness and diffuseness can be described by a profile function $p(z)$. $p(z)$ describes an interface as the normalized, average value of the dielectric function, $\varepsilon(x)$, and is given as in equation (2.27) where $\varepsilon(x)$ is as in (2.28) [3].

$$p(z) = \frac{\int\int \varepsilon(x) dx dy}{(\varepsilon_i - \varepsilon_j) \int\int dx dy} \tag{2.27}$$

$$\varepsilon(x) = \begin{cases} \varepsilon_i, z \to +\infty \\ \varepsilon_j, z \to -\infty \end{cases} \tag{2.28}$$

The most frequently used interface profile function is the error function, which can be seen in equation (2.29).

$$p(z) = \frac{1}{\sqrt{(\pi)}} \int_{-\infty}^{z} \exp\left(\frac{t^2}{2\sigma^2}\right) \tag{2.29}$$

This function can be used to modify the Fresnel reflection coefficients to account for interface imperfection. To do this, the Fresnel reflections is multiplied by a function $\widetilde{w}(s)$, see equation 2.30, which is the Fourier transformation of $w(s) = \frac{dp}{dz}$ [3].

$$r_{ij}^{'} = r_{ij}\widetilde{w}(s_i) \tag{2.30}$$

where

$$\widetilde{w}(s) = \exp\left(-\frac{s^2\sigma_{rms}^2}{2}\right) \tag{2.31}$$

$$s_{ij} = 4\pi\frac{\sqrt{\cos(\theta_i)\cos(\theta_j)}}{\lambda} \tag{2.32}$$

3 Facilities and software

3.1 Multilayer laboratory

The Multilayer laboratory in room 227 in building 328 of DTU Space have been used to produce the thin films in this project. The laboratory is a class 10000 clean room and contains a vacuum chamber where direct current (DC) magnetron sputtering can be performed. Besides the chamber, the multilayer laboratory includes the necessary equipment for operating the chamber and preparing samples. Computers, running SPEC, control the operation of the coating chamber and electronics and a cooling system ensure that cathodes do not overheat. A down flow module provides an extra clean environment to mount and demount the samples before and after the coating. A DEKTAK profilometer enable that elevation profiles of stress samples can be measured pre- and post coating so that tensile and compressive stresses can be determined. In Figure 4 the coating chamber is imaged.

(a) **(b)**

Figure 4: (a): Vacuum chamber in the multilayer laboratory with lowered bell (closed). (b): Vacuum chamber in the multilayer laboratory with raised bell (open). Magnetron 4 with a Ni target mounted is viewed in the middle of the picture.

The configuration of the vacuum chamber is that four magnetrons are fixed inside a moving ring element. Each magnetron is facing outwards and they are separated with a 90° interval. Different target materials can be vertically mounted at each magnetron to permit the fabrication of up to 4 different thin films with up to 4 different material combinations in a coating session. In Figure 5 a top view sketch of the vacuum chamber configuration can be seen.

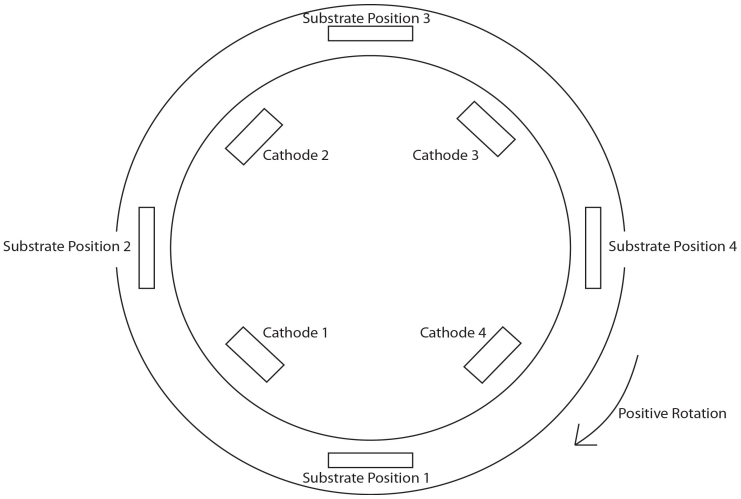


Figure 5: Top view of the vacuum chamber configuration

The magnetrons in the coating chamber consist of a copper cathode surrounded by a steel shield that acts as anode. From inside the cathode three permanent magnets with changing poles provide a magnetic field in front of the cathode. By applying a negative voltage from the anode to cathode, free electrons from the target move into the gas and get trapped by the magnetic field. This create a high density of electrons in front the target. The trapped electrons will ionize the gas in the chamber, creating a plasma in front of the target. The positive ions in the plasma will due to the applied electrical field be accelerated towards the target. Here they will collide with the target material and rip off atoms. The ripped off target atoms will travel away from the target with high kinetic energy and be deposited on the substrate that sits opposite the target. In Figure 6 the principal of a magnetron is sketched.

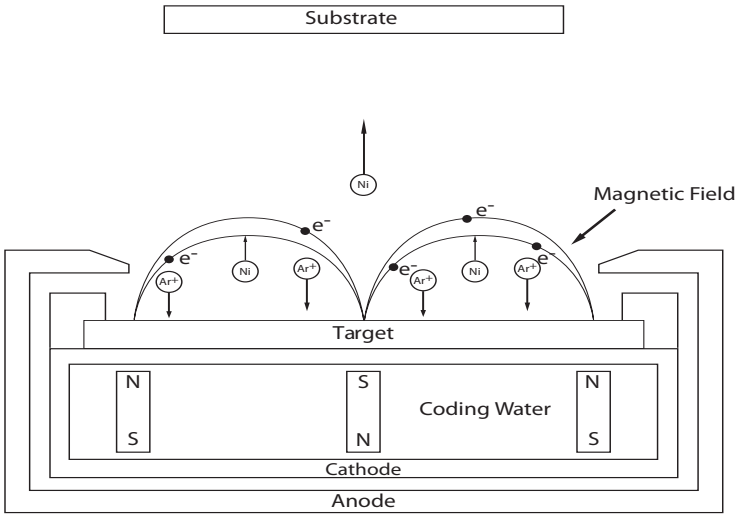


Figure 6: Principal of the magnetrons in the vacuum chamber.

3.1.1 The procedure of coating session

For this project, a coating session is initiated by mounting the prepared substrates at their respective positions in the chamber, see Figure 5. The chamber is then cleaned with a vacuum cleaner and ethanol before it is closed.
Once the chamber is closed, it is pumped down to a base pressure of approximate 10^{-6} [Torr] while heating is applied to help possible liquids evaporate.
Ideally the base pressure should be at least 10^{-3} of the coating pressure, to ensure a clean environment, where very little contamination of the of the coating can happen. However, during this project, a leak in the cooling supply of magnetron 4 prevented the base pressure to get below $2 \cdot 10^{-5}$ [Torr]. The pump down takes about 6 hours and is executed by a roughening pump at high pressures and a molecular turbo pump at low pressures.

When the base pressure is reached, the actual coating procedure can be performed, by running a custom coating script from the operating computer.
Initially the coating script turns on a procedure that every 5 seconds is cataloguing the coating parameters like pressure, cathode powers and voltage etc. Then the script establishing the correct gas flow in [sccm] which subsequently also determines the coating pressure. If the session is to be done in pure Ar gas, only the Ar valve is open and likewise the Ar and N_2 valve are opened if the the session have to be done in a mixed gas.
When the coating pressure have settled, the power supply of the magnetrons are turned on and the coating can begin. By switching the power supply to the relevant magnetron on and off along with controlling the speed of witch the substrates rotate by the magnetrons, it is possible to control which material that is deposited on the substrate and the thickness of it. For multilayers this is done iteratively.
For all coating session done in this project the distance between the cathodes the substrates have been fixed to 155 [mm] with a honeycomb grid in front of the cathode to collimate the sputter direction. Furthermore the applied power of the Ni cathode was fixed at 600 [W] and the power of the B_4C cathode was fixed to 1000 [W].

3.2 X-Ray reflection laboratory

In this project all X-Ray reflectometry (XRR) measurements are done in the X-ray laboratory at DTU Space. The laboratory is located in room 903/907 in the basement of building 328. In Figure 7 a sketch of the laboratory set up can be seen.

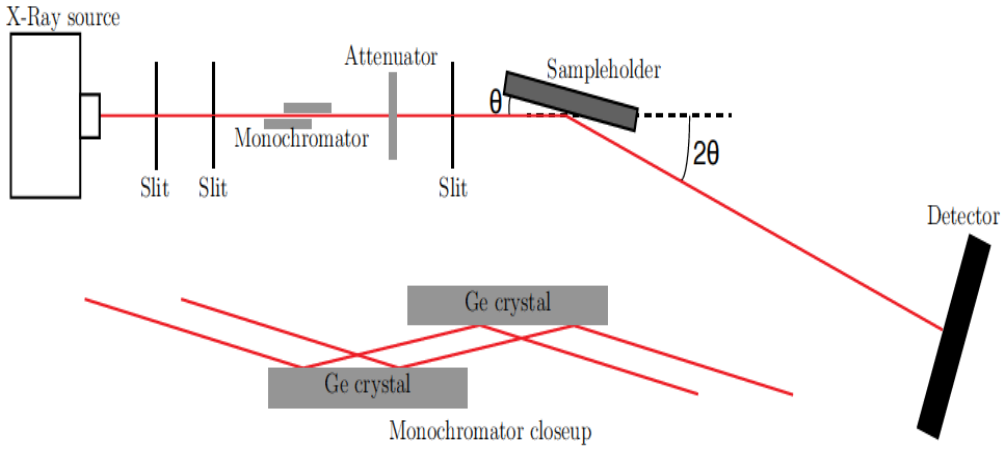


Figure 7: Sketch of the how the equipment is set up in the XRR laboratory [5].

In the set up of the XRR equipment, a rotating copper anode produces the X-ray radiation. When the X-rays leave the source, they are first guided through two slits, that narrow down the footprint of the beam. Secondly the X-ray hit a monochromator made from two germanium crystals. The germanium crystals are aligned, so that only the $K_{\alpha 1}$ emission line at 8.047 [keV] are passed through. After the monochromator, the X-ray beam is directed through an adjustable attenuator made of aluminium, milled down to different thicknesses. The purpose of the attenuator is to ensure that the intensity of the beam not is to high to saturate the detector. When the beam are past the attenuator it is ones again guided through a slit reducing it's divergence before it reaches the sample.
The sample holder is mounted on a platform which can be rotated with an incident angle, θ, from the z-axis. From the platform the detector is also hold in place with an angle 2θ from the z-axis. By this set up the incident beam is reflected from the sample and detected by the detector.

3.3 Analysis software

A software available to simulate and analyse X-ray reflectance from thin films is IMD. IMD is written in the programming language, Interactive Data Language (IDL), and provides the user with numerous options of control for both analysis and design of thin films. The vigorous control options in IMD, is among others things made possible by an extensive library of optical constants and elective physical models that comes along with the programme.
However, IMD is primarily a graphical user interface (GUI) program[1], where film designs, choice of fitting parameters, exports of results etc. have to be done repeatedly. Due to this and the many individual thin films of this project a MATLAB based program have been developed.
The MATLAB program is build on the basis of IMD a can produce the same results to the extend that was necessary for the analysis in this project. Automation ensure consistent settings throughout the analysis and consistent produced plots of the many samples in this project.

3.4 IMD in MATLAB

The MATLAB based program is a segmented program, build to handle both SL's and ML's. It consists of six definition functions, six fitting functions and one main script, where the initial settings of a thin film are declared. If every segment, i.e. all functions and the main script, are kept in the same folder, the program can be executed by only running the main script.

[1] Unless you are an experienced IDL user, which I am not.

3.4.1 The main script

The main script of the MATLAB based program, constitute the interface of the program, where the user can specify the initial condition. The source code of the main script can be found in Appendix B.1.
For the program to run it is necessary to decide if a fit or simulation of a thin film is desired. If a simulation of the specular reflectance from a thin film is desired, the film designs have to be specified followed by the function calls of the definition functions `FilmDefinitions` and `IMD`.
If it is desired to fit a XRR reflectance curve, the film designs have to be specified along with some fitting specifications. In Figure 8 a guide of how to fill out the main script according to what is desired can be seen.

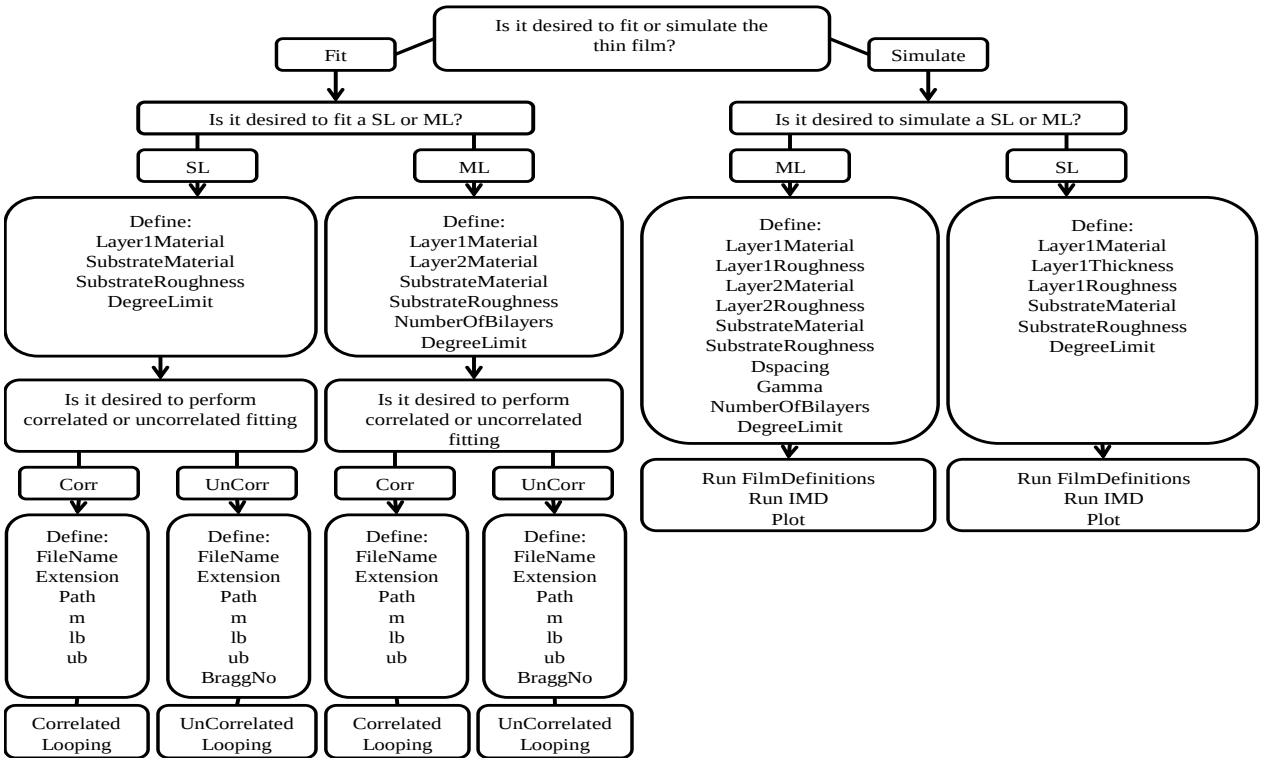


Figure 8: Flowchart of how to declare the designs of a thin film along with the nessasary function calls in the main script of the MATLAB based program.

3.4.2 The definition functions

The six definition functions form the foundation of the MATLAB based program. This means that it is these functions that set up and define the framework of how the model should be.
Most important is the functions `IMD` and `FilmDefinitions` whose source codes can be seen in Appendix B.2 and B.3. Together these two function are responsible for constructing the thin film and calculating the corresponding reflectance.
`IMD` is build from the information in [3] and uses Paratt's recursive method to calculate the specular reflection from a thin film.
In Paratt's recursive method, Snell's law are used to calculate the angles of reflection and transmission through the layers of a thin film on basis of only the initial incident angle and the refractive indexes. This enables the determination of the net reflection, r_i, of the bottom layer of the film

stack where there is no reflectivity from below. r_i is calculated from equation (2.20) that include the modified Fresnel coefficients in equation (2.30).
When the net reflection from the bottom layer of the film stack is known, a recursive iterative process up through the film stack can be applied. This will at last determine the net reflection from the whole multilayer.
The `IMD` function is executed by the calling sequence seen below:

```
[R] = IMD(ThetaI,NumberOfLayers,n,d,sigma,lambda)
```

The `IMD` input variables, `ThetaI`, `NumberOfLayers`, `n`, `d`, `sigma` and `lambda`, are parameters of the thin film that is to be modelled. These are computed by the definition function `FilmDefinitions`, which has the calling sequence:

```
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(varargin)
```

The input variable of `FilmDefinitions`, `varargin`, is what are defined in the main scripts as either `vSL` or `vML`.

Besides the two definitions function just described, four more definition functions help define how the reflection model are set up. These function are `LoadXRRData`, `FitArea`, `Bounds` and `DataReducing` and their source codes can be found in Appendix B.4 to B.7.

The definition function `LoadXRRData` are responsible for loading the measured XRR data so a comparison of the modelled reflectance and the measured reflectance can be performed. `FitArea`, which only are used when uncorrelated fitting are performed, is responsible for defining the degree interval of the critical angle along with intervals for the present Bragg peaks.
The function `Bounds` is accountable of defining 40 points within the user defined lower and upper bounds in the main script and `DataReducing` ensuring the model only is calculated in the user specified degree interval.

3.4.3 The fitting functions

The fitting functions of the MATLAB based program are responsible for the fitting of a thin film if specified by the main script. The fitting functions `CorrelatedFittingDG`, `CorrelatedFittingSS`, `CorrelatedFittingZS` and `UncorrelatedFitting` perform the actual fitting whereas the fitting functions `CorrelatedLooping` and `UncorrelatedLooping` perform a looping sequence where the other four fitting functions subsequently are called. The source code of all the fitting functions can be seen in Appendix B.8 to B.13.

The problem the fitting algorithms are trying to solve is as below:

$$\min \chi^2 = \min \frac{S}{N-n} \tag{3.1}$$

Where S is as in equation (3.2), n is the degrees of freedom and W is the a instrumental weighting corresponding to the uncertainties of the XRR measurement.

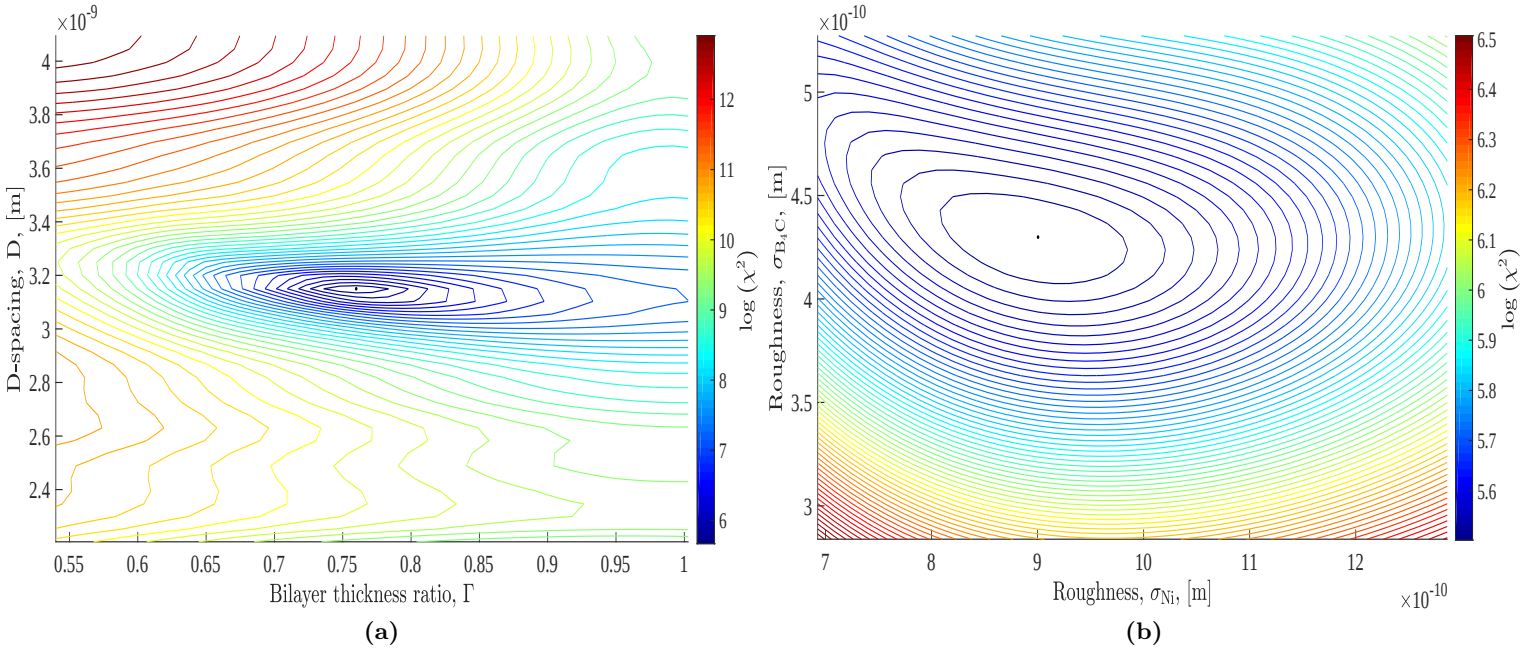
$$S = \sum_{i=1}^{N} \frac{(R_{model}(i) - R_{measured}(i))^2}{W(i)^2} \tag{3.2}$$

3.4.4 Example of fits

Correlated fitting:
To fit the reflectance from the thin film with the identification number si6478, the main script is set up by the procedure described in Figure 8.
From the coating session it is known that si6478 is a ML composed of 10 bilayers of nickel, Ni, and boron carbide, B_4C. The ML is sputtered on substrate of silicon, Si, which the manufacturer provide with roughness at 2.5 [Å].
At the XRR laboratory, the ML have been measured at gracing incident angles from 0° to 4° and it took three measurements to adjust the filters of the detector to ensure that the it did not become saturated, i.e. the extension of the file ID is 3.
The bilayer thickness also called the D-spacing, is aimed to be 70 [Å], with a bilayer thickness ratio, $\Gamma = 0.6$. The roughness' of the two materials are not really know. It is desired to make a correlated fit of the ML.
This information is declared in the main script as seen below:

```
% Initialising:
clear all
close all
clc

%% Defining multilayer:
Layer1Material     = 'B4C';
Layer1Roughness    = [];
Layer2Material     = 'Ni';
Layer2Roughness    = [];
SubstrateMaterial  = 'Si';
SubstrateRoughness = 2.5*10^(-10);
Dspacing           = [];
Gamma              = [];
NumberOfBilayers   = 10;
DegreeLimit        = [0.2 4 nan nan]';

% Concatenate definitions:
vML = {SubstrateMaterial,SubstrateRoughness,Layer1Material,Layer1Roughness,...
       Layer2Material,Layer2Roughness,Dspacing,Gamma,NumberOfBilayers,DegreeLimit};

%% Defining file:
SampleName = 'si6477';
Extension = 2;
Path = strcat('D:\Dropbox\Master\XRRData\',SampleName,'\');
FileName = strcat(SampleName,'_',num2str(Extension));

% Printing witness sample specifications:
fid = fopen('FileToBeFitted.txt','wt');
fprintf(fid,'%s\n%s',FileName,Path);
fclose(fid);

%% Fitting si6477_2
%    [Dspacing    Gamma  SigmaB4C    SigmaNi     ]
lb = [20*10^(-10) 0.1    1*10^(-10)  4*10^(-10)  ]';
m  = [30*10^(-10) 0.6    6*10^(-10)  12*10^(-10) ]';
ub = [40*10^(-10) 0.9    12*10^(-10) 20*10^(-10) ]';

% Executing correlated fitting:
[Out] = CorrelatedLooping(m,lb,ub,vML);

% Reducing fitting interval:
OutChange1 = Out(1,end)*0.3;
OutChange2 = Out(2,end)*0.3;
OutChange3 = Out(3,end)*0.3;
OutChange4 = Out(4,end)*0.3;

% Defining new bounds:
lb = [Out(1,end)-OutChange1 Out(2,end)-OutChange2,...
      Out(3,end)-OutChange3  Out(4,end)-OutChange4 ]';
m  = [Out(1,end)             Out(2,end)            ,...
      Out(3,end)              Out(4,end)             ]';
ub = [Out(1,end)+OutChange1 Out(2,end)+OutChange2,...
      Out(3,end)+OutChange3  Out(4,end)+OutChange4 ]';

% Executing correlated fitting:
[Out] = CorrelatedLooping(m,lb,ub,vML);
```

The result of the correlated fit can be seen in Figure 9 and in Figure 10 the output from the MATLAB commandwindow is shown.



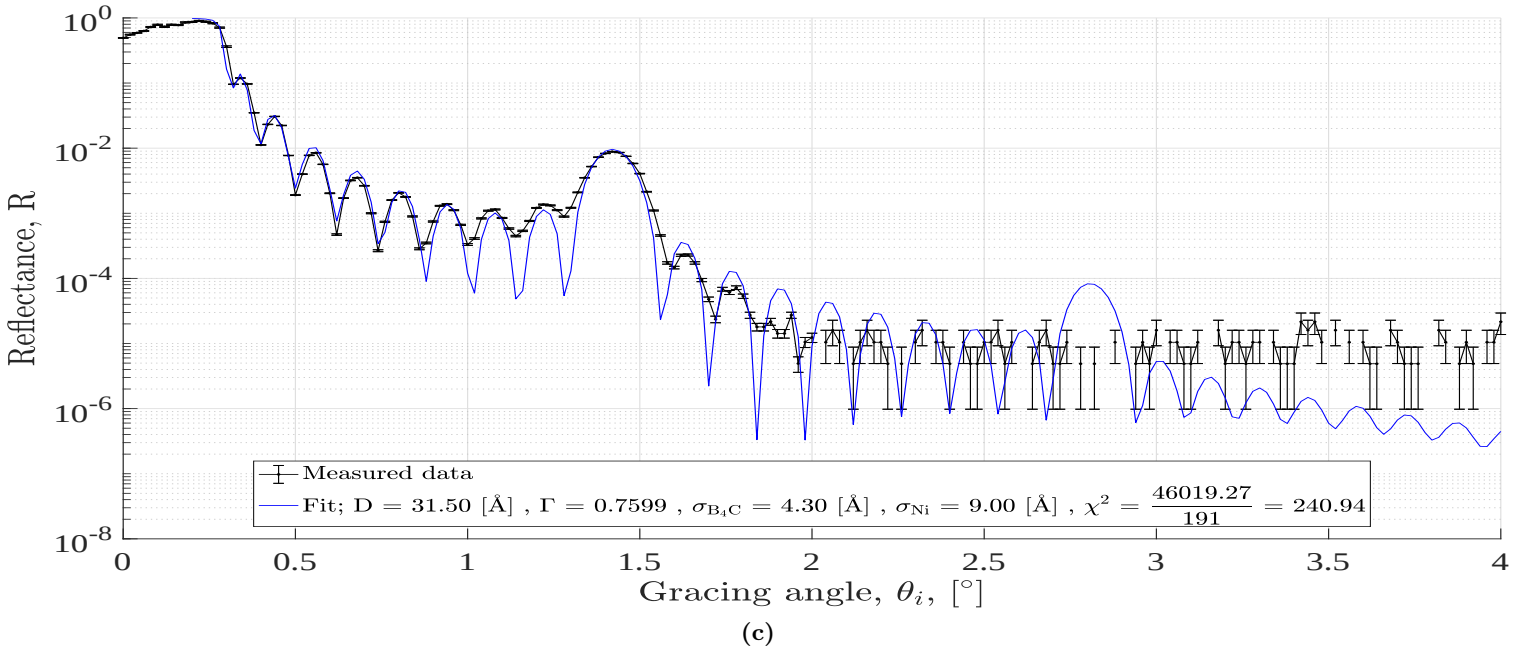


Figure 9: (a): Contour plot with the logarithmic values of χ^2 as function of the D-spacing and Γ. The black dot indicate the determined minimum. (b): Contour plot with the logarithmic values of χ^2 as function of the roughness of Ni and the roughness of B_4C. The black dot indicate the determined minimum. (c): The measured reflectance and fit, when the values of D, Γ, σ_{Ni} and σ_{B_4C} are set to the values corresponding to the minimum of χ^2.

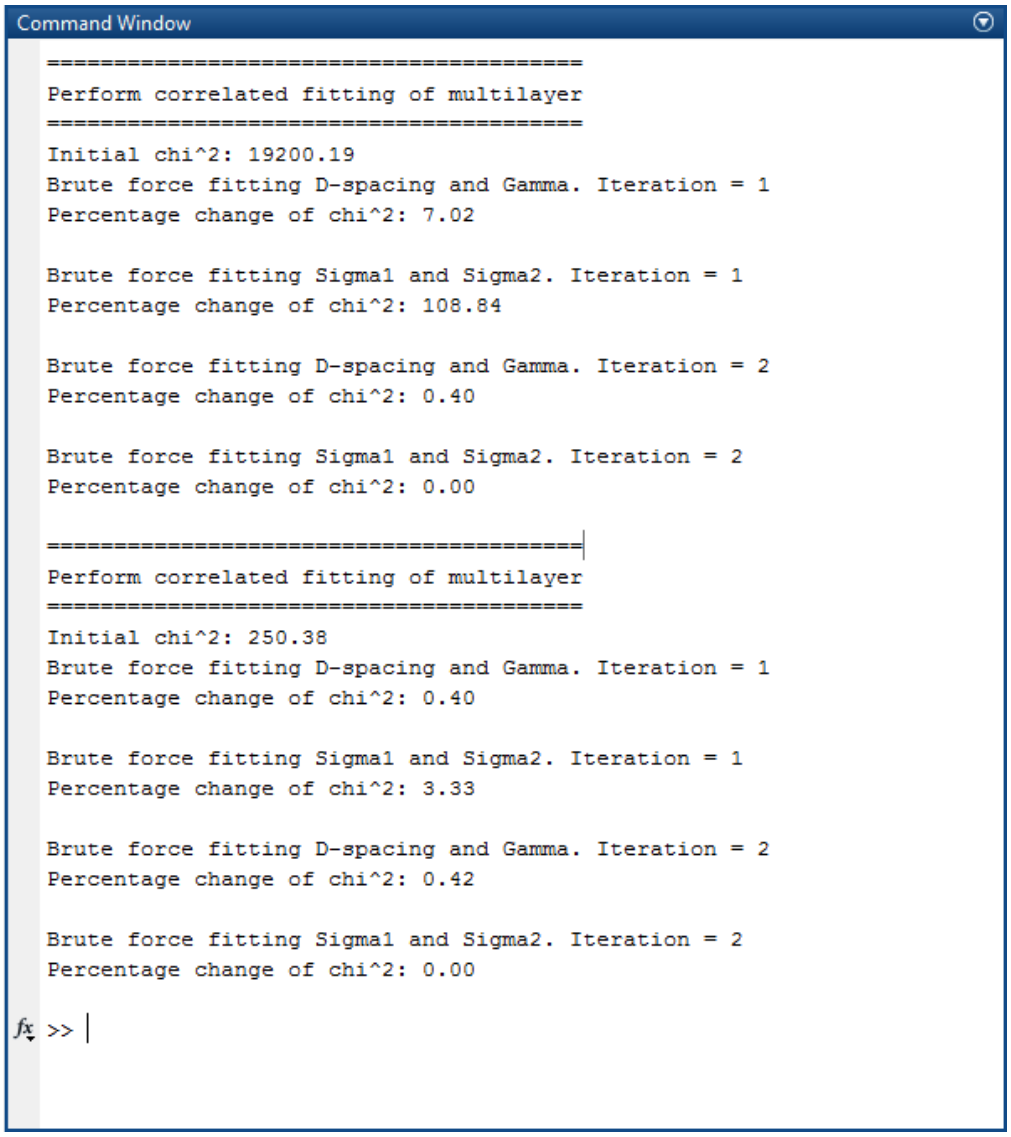


Figure 10: Output from the MATLAB commandwindow during the correlated fitting. The relative change of the minimal χ^2 value is displayed for each iteration. When the the minimal χ^2 value do not change anymore the fitting iteration process is stopped.

4 Results and analysis

4.1 Phase 1 - Coatings in pure argon

Phase 1 of the project studies the properties and behaviour of coatings produced in 100% argon. Four SL's of different thicknesses were produced for four different pressures. The thicknesses of the SL's were aimed to be 50, 100, 200 and 300 [Å] at the pressures 2.5, 3.0, 4.0 and 5.0 [mTorr].
To obtain the thickness of 50 [Å] a coating rate at 2000 [steps $\cdot$ s^{-1}] was used and likewise aimed thicknesses at 100, 200 and 300 [Å] were produced at coating rates equal to 1000, 500 and 375 [steps $\cdot$ s^{-1}].
Analysis of the produced thin films, determined that the coating rates happened to be too fast, resulting in thicknesses less than the aims. This is however not an issue for the analysis and the rates were therefore kept throughout the project to enable a direct comparison across the different phases.
The parameters of each film in phase 1, are listed in Table 1. D, Γ, z_{Ni}, z_{B_4C}, σ_{Ni}, σ_{B_4C} and χ^2, are obtained by fitting the measured reflectance using the developed fitting program described in section 3.4. DDR_{Ni} and DDR_{B_4C} are calculated by the function seen in Appendix B.14.

Table 1: Table of thin films produced in phase 1 along with their respective parameters. ID lists the thin films identification number. Type note if the thin film is a single layer (SL) or a multilayer (ML). Phase indicate which phase the film belongs to. Gas indicate in which gas/gasses the film is produced. The gas fraction, GF, specify the gas fractions of the chamber while the thin film was produced. P denotes under which coating pressure the thin film was produced. D is the thickness of the bilayers (also called D-spacing) of which the ML is composed of. Γ is the bilayer thickness ratio, defined as $\Gamma = z_{B_4C}/D$. The columns z_{Ni} and z_{B_4C} are the fitted thickness of the Ni and B_4C layers and likewise σ_{Ni} and σ_{B_4C} are the fitted roughness. The χ^2 column indicate how well the fits describes the XRR data. DDR_{Ni} and DDR_{B_4C} are the dynamic deposition rates of Ni and B_4C.

ID	Type	Phase	Gas	GF [Ar%/N%]	P [mTorr]	D [Å]	Γ	z_{Ni} [Å]	z_{B_4C} [Å]	σ_{Ni} [Å]	σ_{B_4C} [Å]	χ^2	DDR_{Ni} $\left[\frac{nm\cdot m}{min}\right]$	DDR_{B_4C} $\left[\frac{nm\cdot m}{min}\right]$
si6433	SL	1	Ar	100/0	2.5	-	-	47.00	-	12.02	-	17.05	2.5199	-
si6434	SL	1	Ar	100/0	2.5	-	-	80.00	-	8.35	-	9.45	2.1446	-
si6435	SL	1	Ar	100/0	2.5	-	-	163.96	-	9.94	-	8.98	2.1976	-
si6436	SL	1	Ar	100/0	2.5	-	-	219.57	-	11.00	-	19.37	2.2073	-
si6437	SL	1	Ar	100/0	4.0	-	-	43.34	-	13.97	-	26.81	2.3236	-
si6438	SL	1	Ar	100/0	4.0	-	-	73.00	-	13.08	-	15.22	1.9569	-
si6439	SL	1	Ar	100/0	4.0	-	-	143.96	-	14.72	-	27.60	1.9296	-
si6440	SL	1	Ar	100/0	4.0	-	-	201.30	-	18.28	-	35.28	2.0236	-
si6441	SL	1	Ar	100/0	5.0	-	-	43.00	-	13.08	-	28.37	2.3054	-
si6442	SL	1	Ar	100/0	5.0	-	-	72.00	-	12.32	-	6.41	1.9301	-
si6443	SL	1	Ar	100/0	5.0	-	-	136.38	-	14.95	-	36.37	1.8280	-
si6444	SL	1	Ar	100/0	5.0	-	-	189.13	-	18.08	-	24.49	1.9013	-
si6449	ML	1	Ar	100/0	2.5	51.25	0.5607	22.51	28.74	12.00	8.00	5319.80	1.9527	0.4584
si6450	ML	1	Ar	100/0	3.0	28.21	0.5951	11.42	16.79	14.73	12.76	2832.68	1.9810	0.5361
si6451	ML	1	Ar	100/0	3.0	47.00	0.5167	22.72	24.28	12.00	7.43	4522.24	1.9709	0.3873
si6452	ML	1	Ar	100/0	2.5	30.80	0.5423	14.10	16.70	12.00	8.29	2840.95	2.4459	0.5332
si6453	ML	1	Ar	100/0	4.0	29.05	0.5949	11.77	17.28	18.41	17.78	415.26	2.0417	0.5517
si6454	ML	1	Ar	100/0	5.0	43.97	0.5615	19.28	24.69	20.72	18.87	1834.20	1.6725	0.3938
si6455	ML	1	Ar	100/0	5.0	22.59	0.6197	8.59	14.00	29.33	23.02	99.91	1.4901	0.4470
si6456	ML	1	Ar	100/0	4.0	47.69	0.5098	23.38	24.31	12.00	8.41	21299.62	2.0282	0.3877

The fitted and calculated film parameters, depends strongly on how well the measured reflectance is fitted and hence all measured reflectance curves along with their corresponding fits can be found in Appendix A.1 to A.4.

In Figure 11 the reflectance as function of gracing incident angle is shown for all SL coatings in phase 1. SL's that are produced at the same coating rates are plotted together. In Figure 11 (a) the SL's with an aimed thickness at 50 [Å] are shown and likewise SL's with aimed thicknesses at 100, 200 and 300 [Å] are shown in (b), (c) and (d).
From the figures it is seen that that low coating pressures seem to produce the best SL's. For all thicknesses the SL's produced at 2.5 [mTorr] show noticeable higher reflectance. When this is said it is also noted that the 2.5 [mTorr] films are thicker. The Kiessig fringes are shifted towards lower incident angles which corresponds to that the films have a greater thicknesses. This trend is also what is seen from the thicknesses in Table 1.

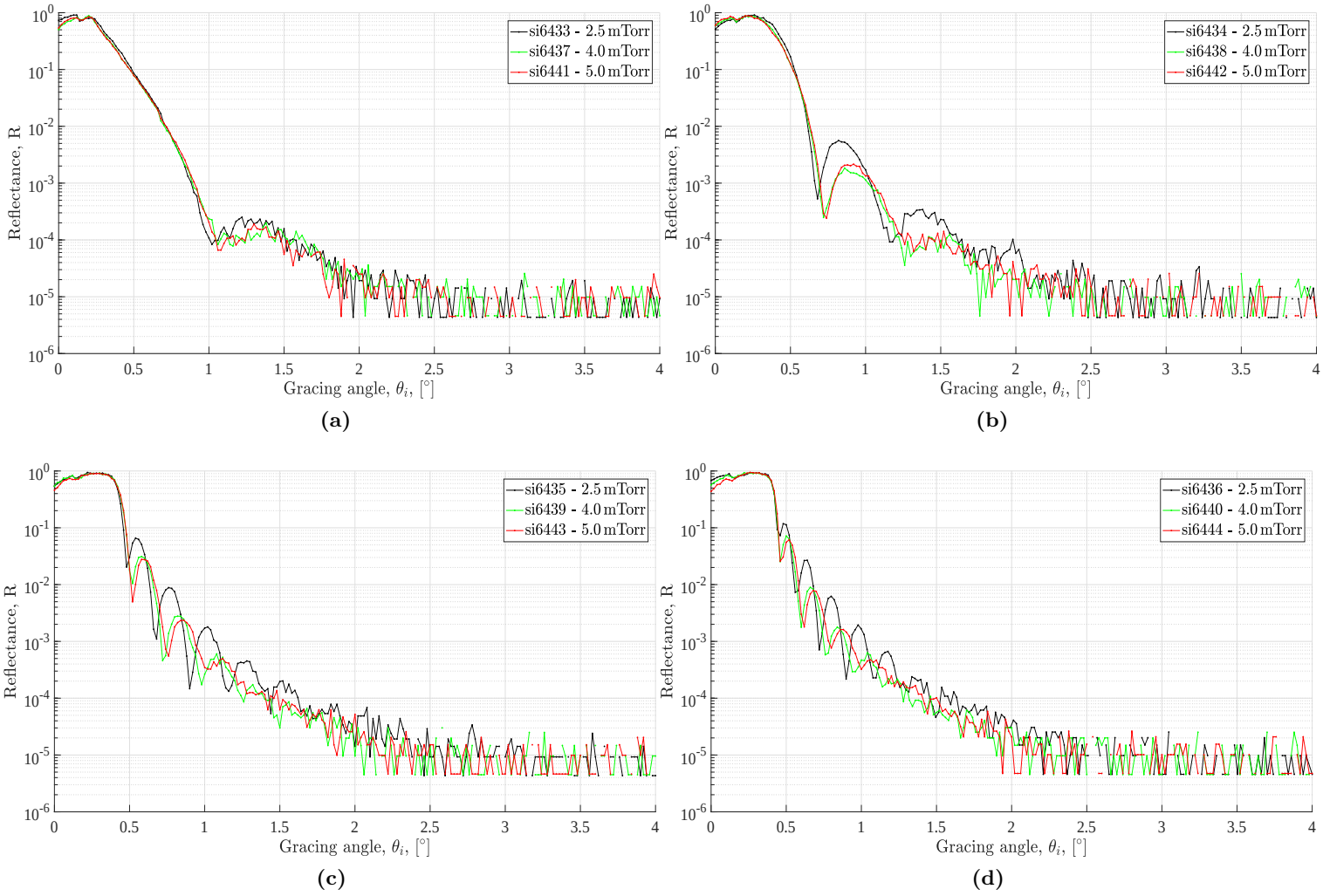
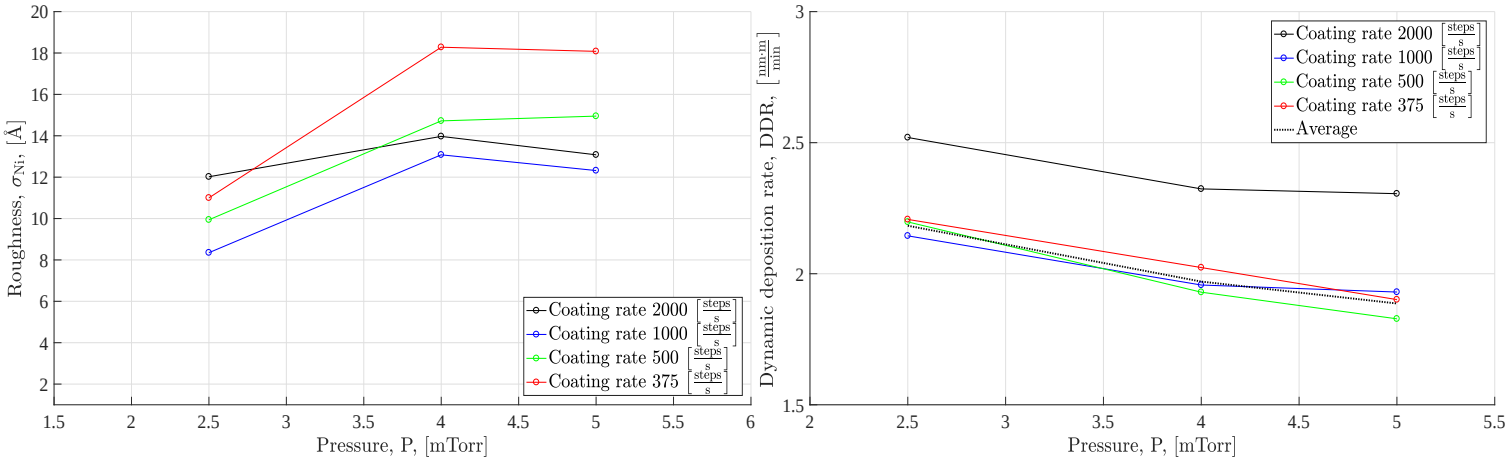


Figure 11: (a): SL coatings produced with a coating rate at 2000 [steps $\cdot$ s^{-1}] in 100% Ar gas. The average thickness, z_{avg}, of the films are 44.45 [Å] (b): SL coatings produced with a coating rate at 1000 [steps $\cdot$ s^{-1}] in 100% Ar gas. The average thickness, z_{avg}, of the films are 75.00 [Å]. (c): SL coatings produced with a coating rate at 500 [steps $\cdot$ s^{-1}] in 100% Ar gas. The average thickness, z_{avg}, of the films are 148.10 [Å]. (d): SL coatings produced with a coating rate at 375 [steps $\cdot$ s^{-1}] in 100% Ar gas. The average thickness, z_{avg}, of the films are 203.33 [Å].

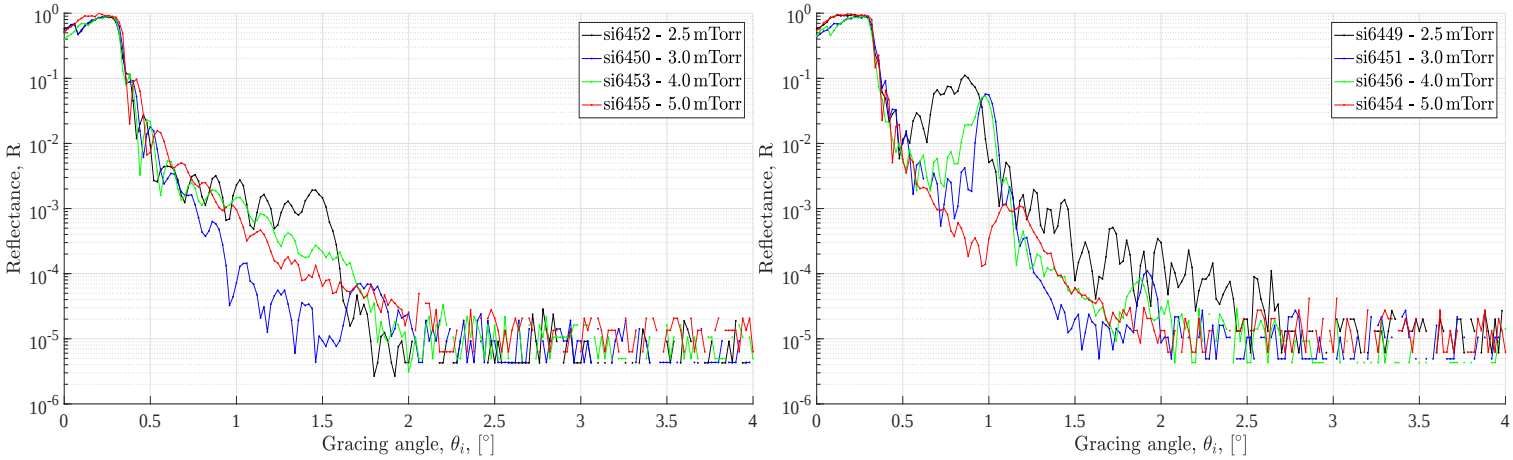
To to investigate the SL's further, their roughnesses and dynamic depositions rates are plotted in Figure 12 (a) and (b) respectively. From Figure 12 (a) it is seen that the overall trend of the roughness is that it increases with increasing coating pressure up until 4 [mTorr] and then decrease slightly. This is consistent with the fact that the coating pressure at 2.5 [mTorr] seems to be the best.
From Figure 12 (b) it found that the dynamic depositions rates decrease almost linearly for all thicknesses, with increasing pressure.



(a) **(b)**

Figure 12: (a): Roughness of Ni as function of coating pressure for all SL's produced in phase 1. (b): Dynamic deposition rate as function of coating preassure for all SL's in phase 1.

With the SL's of phase 1 analysed, the focus of investigation is now directed to the ML's of phase 1. In Figure 13 the reflectance of the ML's produced in phase 1 are shown. The ML's in (a) have aimed D-spacings equal to 35 [Å] and the ML's in (b) have aimed D-spacings equal to 70 [Å]. Again one ML of each D-spacing is produced at the four different pressures described for the SL's.



(a) **(b)**

Figure 13: (a): Reflectance of ML's with aimed D-spacing equal to 35 Å as function of incident gracing angle in phase 1. (b): Reflectance of ML's with aimed D-spacing equal to 70 Å as function of incident gracing angle in phase 1.

Initially it is seen from Figure 13 that something odd is accruing. One would presume that the order of the ML's reflectance curves should correspond with the the order of the SL's, i.e. that the ML's produced at 2.5 [mTorr] should have the highest reflectance and the ML's at 5.0 [mTorr] should have the lowest. The ML's in Figure 13 (b) follow this trend but the ones in (a) does not.
To determine what is causing the mix up of the coatings with aimed D-spacings at 35 [Å], their fitted D-spacing and the roughness have to be considered.
As describe in section 2 the interface between two adjacent mediums is modelled by a interface profile function $p(z)$, which is set to an error function. The error function have a full with at half maximum (FWHM) as in equation (4.1).

$$\text{FWHM} = 2\sqrt{2\ln(2)} \cdot \sigma \approx 2.35482 \cdot \sigma \tag{4.1}$$

From the FWHM it is possible to estimate the average thickness of the bilayers diffusion zone. This is done as in equation (4.2).

$$z_{\text{Diff}} = \frac{\text{FWHM}_{\text{Ni}} + \text{FWHM}_{\text{B}_4\text{C}}}{2} \tag{4.2}$$

For the ML's in Figure 13 (a), z_{Diff} are found to be 29.09, 27.04, 42.52 and 61.51 [Å] for the ML's at 2.5 to 5.0 [mTorr] respectively.
The two first diffusion zones are comparable to the D-spacings of the respective ML's and the two last diffusion zones are greater than their respective ML's D-spacings. Diffusion zones greater than the D-spacing does not have any physical meaning, which indicate that the fit of the ML's not are good enough. Diffusion zones comparable to the D-spacings are physically valid, but correspond to, that the bilayers within the ML's are slaps of randomly mixed Ni and B_4C.
These findings mean that the D-spacing of the ML's in Figure 13 (a) simply are to small compared to the roughness' of the materials to maintain their physical design.
Because of this the ML's in 13 (a) will not be used in the determination of the optimal coating pressure.

Looking at Figure 13 (b) the order of the reflectance curves seems correct. However the ML

produced at 2.5 [mTorr] do not display the correct features of a well defined Bragg peak. This indicate the layer structure within the ML are disrupted in some way. To clear up if something unusual happen during the coating session of ML si6449 its coating log was consulted. In Figure 14 the cathode powers during the coating session of ML si6449 are shown.

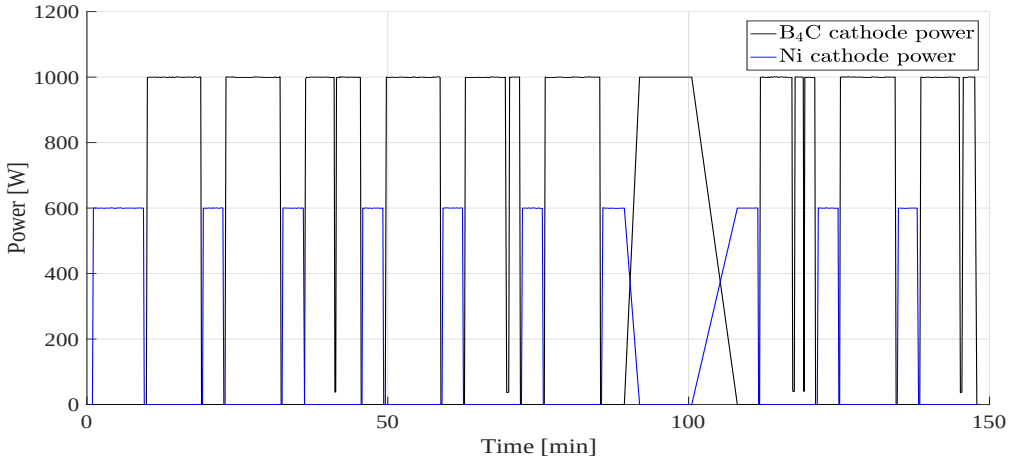


Figure 14: Cathode powers during the coating session of ML si6449.

It seen the that a power outage accrued while the coating was in progress which explains the odd reflectance curve. Besides the power outage it is also noted that drop outs are present when B_4C is coated.

With every atypical ML accounted for and excluded, the optimal coating pressure can be determined on the basis of the both the SL's and ML's of phase 1. Of the three ML's left in the Figure 13 (b) the one produced at 3.0 [mTorr] show the best reflectance. This ML also have the most distinguish Kiessig fringes and Bragg peak. That the ML at 3.0 [mTorr] is the best is further confirmed by Figure 15 where the roughnesses and DDR's of the ML's are plotted as function of pressure. Since the dynamic deposition rate are almost constant for both Ni and B_4C any change in reflectance comes from roughness where the ML at 3.0 [mTorr] has the lowest.

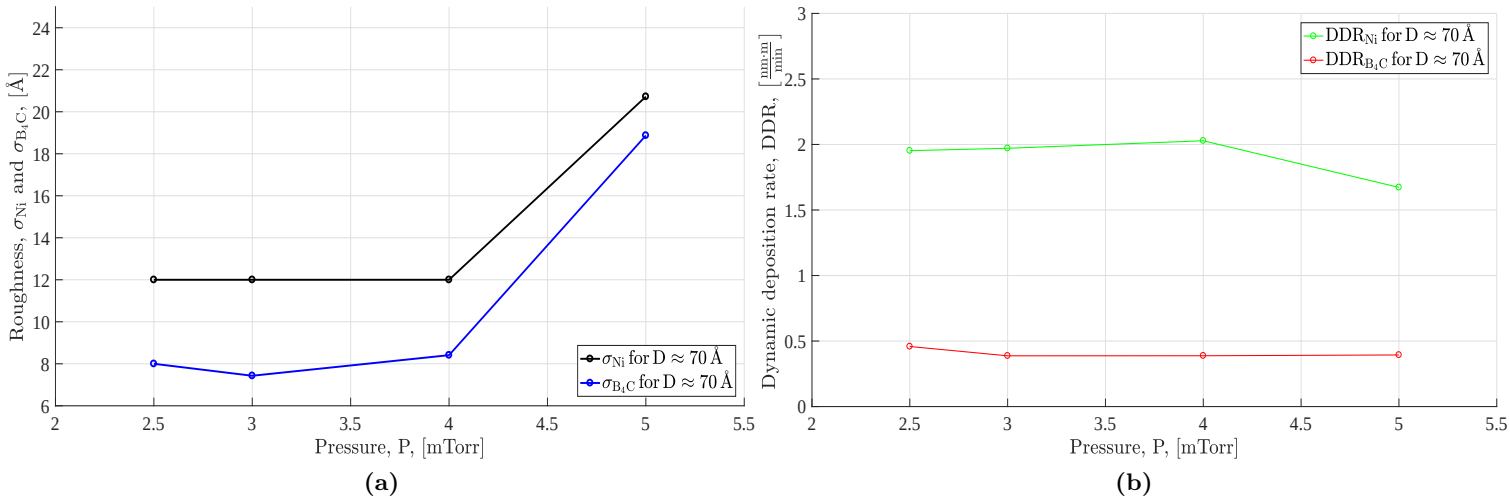


Figure 15: (a): Roughnesses of Ni and B_4C as function of coating pressure for the ML's in phase 1. (b): Dynamic deposition rate as function of coating preassure for ML's in phase 1.

4.2 Phase 2 - Reactive sputtering for different gas fractions

Phase 2 of the project studies the properties and behaviour of coatings produced by reactive sputtering with argon as the inert gas and nitrogen as the reactive gas.
Four SL's of different thicknesses were produced for four different gas fractions. The thicknesses of the coatings were aimed to be the same as for the SL's in phase 1 and the N_2 gas fractions were set to be 5, 15, 20 and 30%. Along with SL's, ML's with aimed D-spacings equal to 35 [Å] and 70 [Å] were also produce for the four different gas fractions.

The parameter of each film in phase 2, are listed in Table 2. As for phase 1, D, Γ, z_{Ni}, z_{B_4C}, σ_{Ni}, σ_{B_4C} and χ^2, are obtained by fitting the measured reflectance using the developed fitting program. DDR_{Ni} and DDR_{B_4C} are determined by function seen in Appendix B.14. The fitted and calculated film parameters depends strongly on how well the XRR measurements are fitted and hence all fits of the reflectance curves in phase 2 can be found in Appendix A.5 to A.8.

Table 2: Table of thin films produced in phase 2 along with their respective parameters. ID lists the thin films identification number. Type note if the thin film is a single layer (SL) or a multilayer (ML). Phase indicate which phase the film belongs to. Gas indicate in which gas/gasses the film is produced. The gas fraction, GF, specify the gas fractions of the chamber while the thin film was produced. P denotes under which coating pressure the thin film was produced. D is the thickness of the bilayers (also called D-spacing) of which the ML is composed of. Γ is the bilayer thickness ratio, defined as $\Gamma = z_{B_4C}/D$. The columns z_{Ni} and z_{B_4C} are the fitted thickness of the Ni and B_4C layers and likewise σ_{Ni} and σ_{B_4C} are the fitted roughness. The χ^2 column indicate how well the fits describes the XRR data. DDR_{Ni} and DDR_{B_4C} are the dynamic deposition rates of Ni and B_4C.

ID	Type	Phase	Gas	GF [Ar%/N%]	P [mTorr]	D [Å]	Γ	z_{Ni} [Å]	z_{B_4C} [Å]	σ_{Ni} [Å]	σ_{B_4C} [Å]	χ^2	DDR_{Ni} $\left[\frac{nm \cdot m}{min}\right]$	DDR_{B_4C} $\left[\frac{nm \cdot m}{min}\right]$
si6457	SL	2	Ar/N	95/5	3	-	-	41.37	-	9.44	-	9.44	2.2180	-
si6458	SL	2	Ar/N	95/5	3	-	-	198.26	-	12.27	-	128.15	1.9930	-
si6459	SL	2	Ar/N	95/5	3	-	-	146.15	-	10.82	-	176.19	1.9589	-
si6460	SL	2	Ar/N	95/5	3	-	-	75.00	-	8.12	-	74.73	2.0105	-
si6461	SL	2	Ar/N	85/15	3	-	-	40.38	-	13.28	-	7.97	2.1649	-
si6462	SL	2	Ar/N	85/15	3	-	-	195.22	-	13.36	-	105.22	1.9625	-
si6463	SL	2	Ar/N	85/15	3	-	-	138.46	-	12.09	-	150.20	1.8558	-
si6464	SL	2	Ar/N	85/15	3	-	-	74.00	-	13.08	-	47.79	1.9837	-
si6465	SL	2	Ar/N	80/20	3	-	-	40.38	-	10.07	-	4.36	2.1649	-
si6466	SL	2	Ar/N	80/20	3	-	-	189.13	-	12.45	-	92.57	1.9013	-
si6467	SL	2	Ar/N	80/20	3	-	-	140.54	-	9.47	-	59.39	1.8837	-
si6468	SL	2	Ar/N	80/20	3	-	-	72.00	-	13.08	-	49.70	1.9301	-
si6469	SL	2	Ar/N	70/30	3	-	-	34.00	-	14.24	-	144.12	1.8229	-
si6470	SL	2	Ar/N	70/30	3	-	-	169.23	-	11.66	-	88.07	1.7012	-
si6471	SL	2	Ar/N	70/30	3	-	-	126.92	-	11.53	-	108.37	1.7012	-
si6472	SL	2	Ar/N	70/30	3	-	-	67.00	-	12.80	-	195.91	1.7961	-
si6473	ML	2	Ar/N	95/5	3	31.85	0.7533	7.86	23.99	12.38	5.72	4174.64	1.2493	1.1762
si6474	ML	2	Ar/N	85/15	3	48.38	0.6429	17.28	31.10	12.40	8.86	2604.16	1.3434	0.7383
si6475	ML	2	Ar/N	85/15	3	29.05	0.6643	9.75	19.30	11.60	5.72	10084.53	1.5159	0.9463
si6476	ML	2	Ar/N	95/5	3	52.31	0.6262	19.55	32.76	12.00	5.17	9062.01	1.5534	0.8036
si6477	ML	2	Ar/N	80/20	3	31.38	0.7714	7.17	24.21	11.20	4.03	268.34	1.1131	1.1870
si6478	ML	2	Ar/N	80/20	3	52.75	0.6643	17.71	35.04	8.63	2.39	10985.46	1.3749	0.8595
si6479	ML	2	Ar/N	70/30	3	32.00	0.6754	10.39	21.61	9.89	6.59	948.22	1.3534	1.0595
si6480	ML	2	Ar/N	70/30	3	53.03	0.6026	21.07	31.96	11.54	3.22	9098.51	1.3720	0.7839

Using the same argumentation as in phase 1, it seen from Figure 16, that coatings produced in gas with 5% N_2 display the best reflectance. This is especially clear from Figure 16 (b), where the SL coated at 5% nitrogen stands out opposed to Figure 16 (a), (c) and (d) where the 5, 15 and 20% coatings are hard to to discriminate from each other.

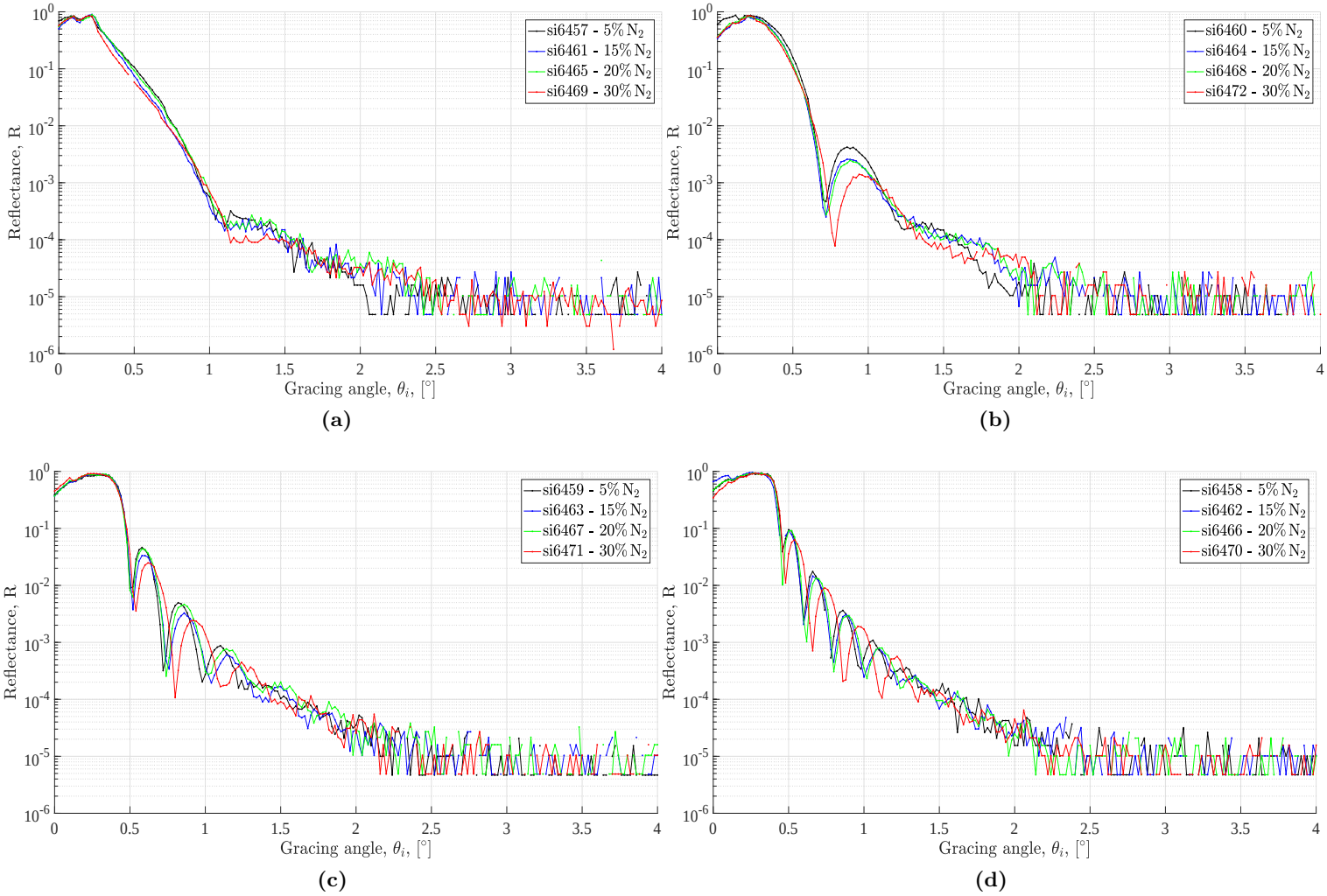
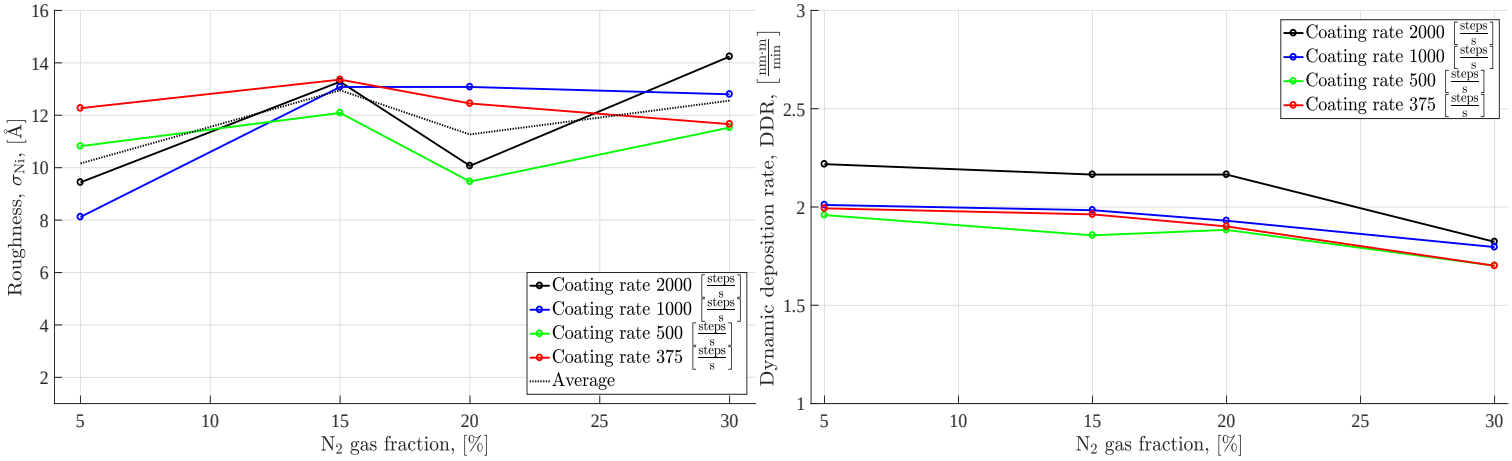


Figure 16: (a): SL coatings produced with a coating rate at 2000 [steps $\cdot$ s^{-1}] in mixed gas. (b): SL coatings produced with a coating rate at 1000 [steps $\cdot$ s^{-1}] in mixed gas. (c): SL coatings produced with a coating rate at 500 [steps $\cdot$ s^{-1}] in mixed gas. (d): SL coatings produced with a coating rate at 375 [steps $\cdot$ s^{-1}] in mixed gas.

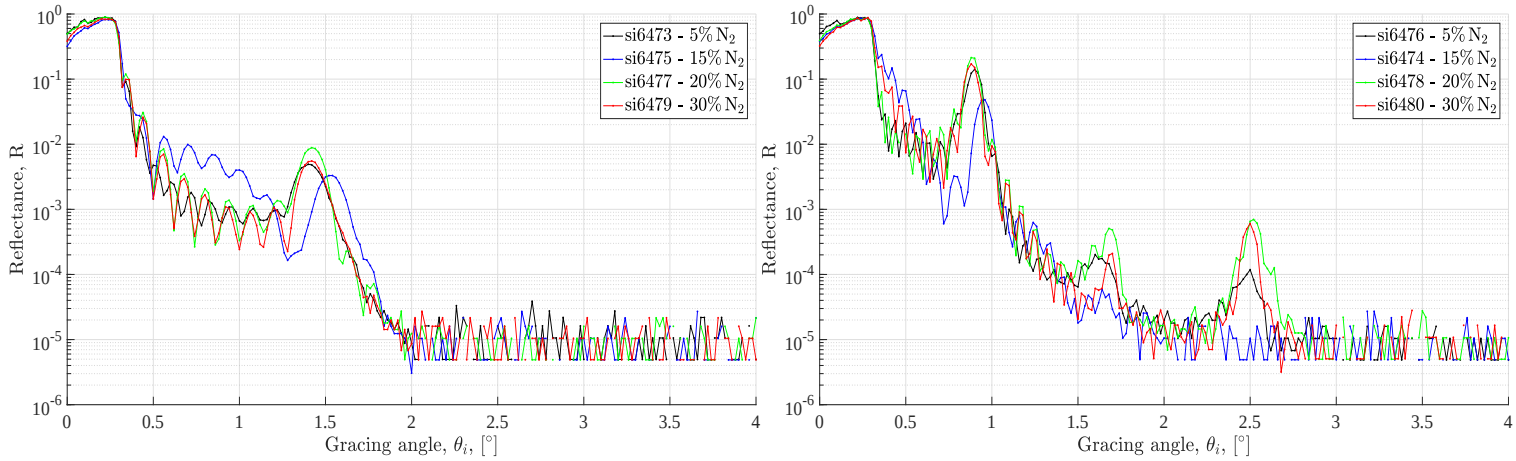
That the 5% N_2 gas fraction provides the best SL's, is also confirmed when Figure 17 is consolidated. In Figure 17 (b) the DDR's versus gas fraction of the SL's in phase 2 are shown. The rates are almost constant for the three lowest nitrogen fractions, and hence the differences in reflectance must arise due to variations in σ_{Ni}. Figure 17 (a) shows σ_{Ni} for the different gas fractions. 5% and 20% N_2 show the lowest roughnesses, but when comparing the average of the SL's for each gas fraction, it is found the 5% have a slightly better one at 10.16 [Å] compared to 11.27 [Å] for the 20%.



(a) **(b)**

Figure 17: (a): Roughness of Ni as function of nitrogen gas fraction for all SL's produced in phase 2. (b): Dynamic deposition rate as function of nitrogen gas fraction for all SL's in phase 2.

In Figure 18 the XXR results of the ML's in phase 2 are plotted. ML's with aimed D-spacings at 35 [Å] are plotted together in (a) and ML's with aimed D-spacings at 70 [Å] are plotted in (b).



(a) **(b)**

Figure 18: (a): Reflectance of ML's with aimed D-spacing equal to 35 Å as function of incident gracing angle in phase 2. (b): Reflectance of ML's with aimed D-spacing equal to 70 Å as function of incident gracing angle in phase 2.

It is clear that the ML's produced with 15% N_2 deviates from the rest. Due to this behaviour the coating log was consolidated and for unknown reasons, a power outage accrued in the middle of the coating as seen in Figure 19. Because of this, the dimensions of ML si6475 and si6474 are not known and henceforth they will not be included in the analysis.

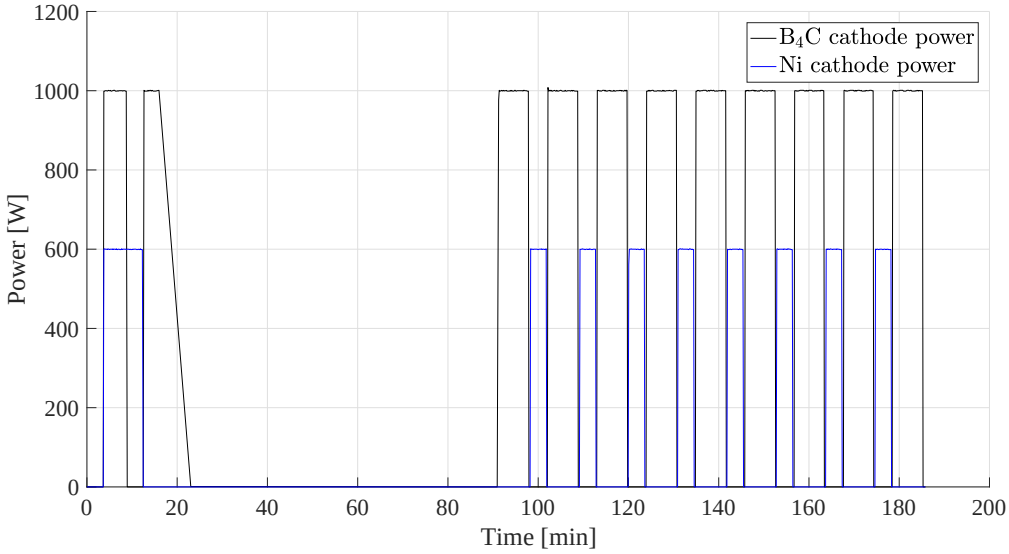


Figure 19: Cathode powers during the coating session of the ML's si6474 and si6475.

The remaining ML's in Figure 18 displays clearly that the optimal ML's are produced at 20% nitrogen. For the ML's with D-spacings aimed at 35 [Å], an obvious higher Bragg peak is present for the 20% coating and the same apply for the ML's with aimed D-spacings equal to 70 [Å]. When the roughness of the ML's are plotted, see Figure 20 (a), these findings are even more obvious. Both types of ML's have a clear decreased roughness when produced in gas with 20% nitrogen. Especially the B_4C seem to smoothed when coated by reactive sputtering.

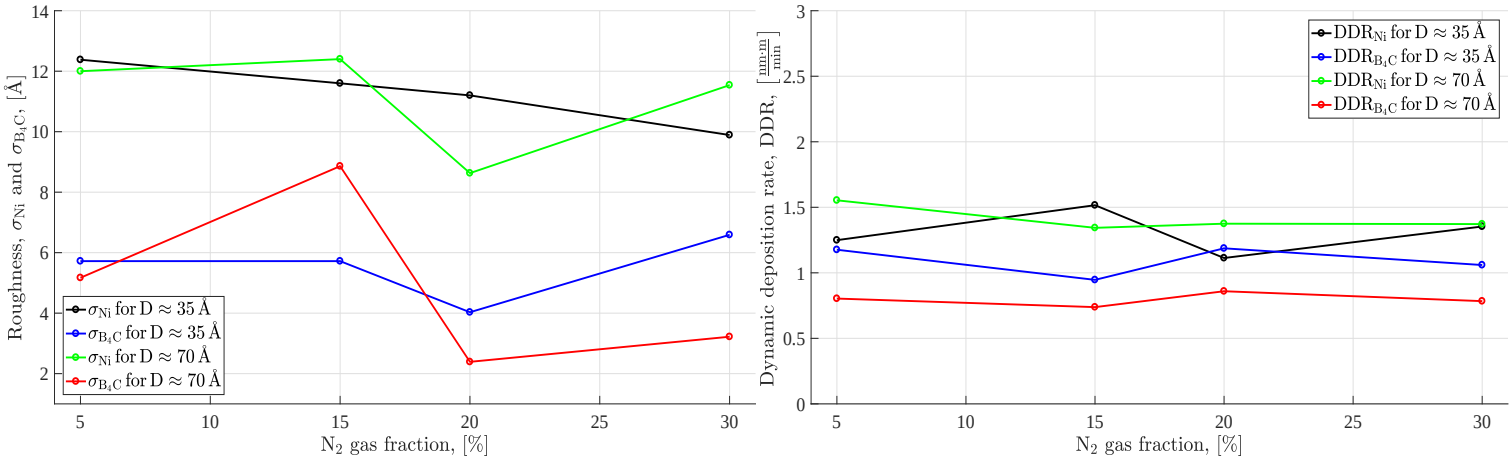


Figure 20: (a): Roughnesses of Ni and B_4C as function of nitrogen gas fraction for the ML's in phase 2. (b): Dynamic deposition rate as function of nitrogen gas fraction for ML's in phase 2.

4.3 Phase 3 - Reactive sputtering for different pressures

Phase 3 of the project studies the properties of coatings produced in gas of 80% argon and 20% nitrogen at different pressures. Four SL's with thicknesses as in the previous phases are produced at the same pressures as in phase 1. ML's with aimed D-spacings at 30, 60, 90 and 120 [Å] are produced at 3.0 [mTorr].

The parameters of each film in phase 3, are listed in Table 3. The parameters are the same as for phase 1 and 2. All fits of the reflectance curves in phase 3 can be found in Appendix A.9 to A.12.

Table 3: Table of thin films produced in phase 2 along with their respective parameters. ID lists the thin films identification number. Type note if the thin film is a single layer (SL) or a multilayer (ML). Phase indicate which phase the film belongs to. Gas indicate in which gas/gasses the film is produced. The gas fraction, GF, specify the gas fractions of the chamber while the thin film was produced. P denotes under which coating pressure the thin film was produced. D is the thickness of the bilayers (also called D-spacing) of which the ML is composed of. Γ is the bilayer thickness ratio, defined as $\Gamma = z_{B_4C}/D$. The columns z_{Ni} and z_{B_4C} are the fitted thickness of the Ni and B_4C layers and likewise σ_{Ni} and σ_{B_4C} are the fitted roughness. The χ^2 column indicate how well the fits describes the XRR data. DDR_{Ni} and DDR_{B_4C} are the dynamic deposition rates of Ni and B_4C.

ID	Type	Phase	Gas	GF [Ar%/N%]	P [mTorr]	D [Å]	Γ	z_{Ni} [Å]	z_{B_4C} [Å]	σ_{Ni} [Å]	σ_{B_4C} [Å]	χ^2	DDR_{Ni} $\left[\frac{nm \cdot m}{min}\right]$	DDR_{B_4C} $\left[\frac{nm \cdot m}{min}\right]$
si6481	SL	3	Ar/N	80/20	3.0	-	-	33.99	-	22.10	-	489.76	1.8223	-
si6482	SL	3	Ar/N	80/20	3.0	-	-	70.00	-	13.48	-	214.62	1.8765	-
si6483	SL	3	Ar/N	80/20	3.0	-	-	136.63	-	13.14	-	148.30	1.8313	-
si6484	SL	3	Ar/N	80/20	3.0	-	-	183.33	-	13.55	-	172.24	1.8429	-
si6485	SL	3	Ar/N	80/20	2.5	-	-	40.00	-	14.97	-	195.90	2.1446	-
si6486	SL	3	Ar/N	80/20	2.5	-	-	72.00	-	13.97	-	115.35	1.9301	-
si6487	SL	3	Ar/N	80/20	2.5	-	-	142.00	-	14.27	-	341.37	1.9033	-
si6488	SL	3	Ar/N	80/20	2.5	-	-	192.17	-	15.45	-	288.41	1.9318	-
si6489	SL	3	Ar/N	80/20	4.0	-	-	36.45	-	15.44	-	117.04	1.9542	-
si6490	SL	3	Ar/N	80/20	4.0	-	-	68.00	-	14.45	-	347.58	1.8229	-
si6491	SL	3	Ar/N	80/20	4.0	-	-	130.77	-	16.13	-	200.64	1.7528	-
si6492	SL	3	Ar/N	80/20	4.0	-	-	174.76	-	17.77	-	268.80	1.7568	-
si6493	SL	3	Ar/N	80/20	5.0	-	-	33.00	-	19.04	-	197.54	1.7693	-
si6494	SL	3	Ar/N	80/20	5.0	-	-	61.00	-	15.20	-	138.20	1.6352	-
si6495	SL	3	Ar/N	80/20	5.0	-	-	121.02	-	18.08	-	83.66	1.6221	-
si6496	SL	3	Ar/N	80/20	5.0	-	-	171.90	-	23.90	-	335.61	1.7280	-
si6497	ML	3	Ar/N	80/20	3.0	35.00	0.5411	16.06	18.94	7.79	3.74	628.10	1.5705	1.2353
si6498	ML	3	Ar/N	80/20	3.0	57.73	0.5200	27.71	30.02	11.76	2.23	3207.83	1.3549	0.9786
si6499	ML	3	Ar/N	80/20	3.0	84.90	0.4284	48.53	36.37	14.67	3.56	4132.78	1.5819	0.7907
si6500	ML	3	Ar/N	80/20	3.0	113.50	0.4346	64.17	49.33	20.00	3.22	6235.76	1.5688	0.8040

In Figure 21 the reflectance from all SL's in phase 3 are plotted. SL's produced at the same coating rates are plotted together. In Figure 21 (a) SL's with aimed thicknesses at 50 [Å] are shown and Figure 21 (b), (c) and (d) show the SL's with aimed thicknesses at 100, 200 and 300 [Å] respectively.

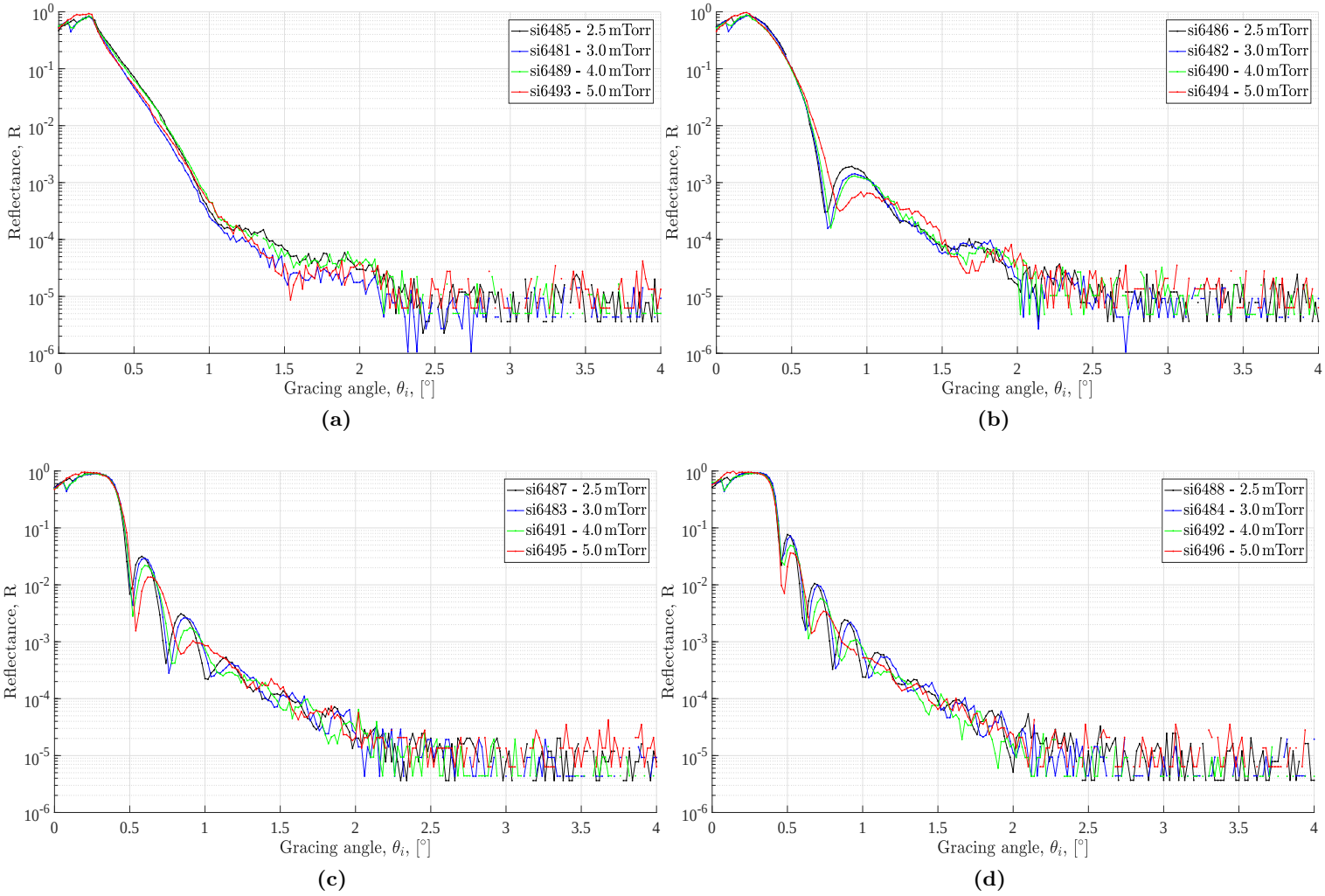
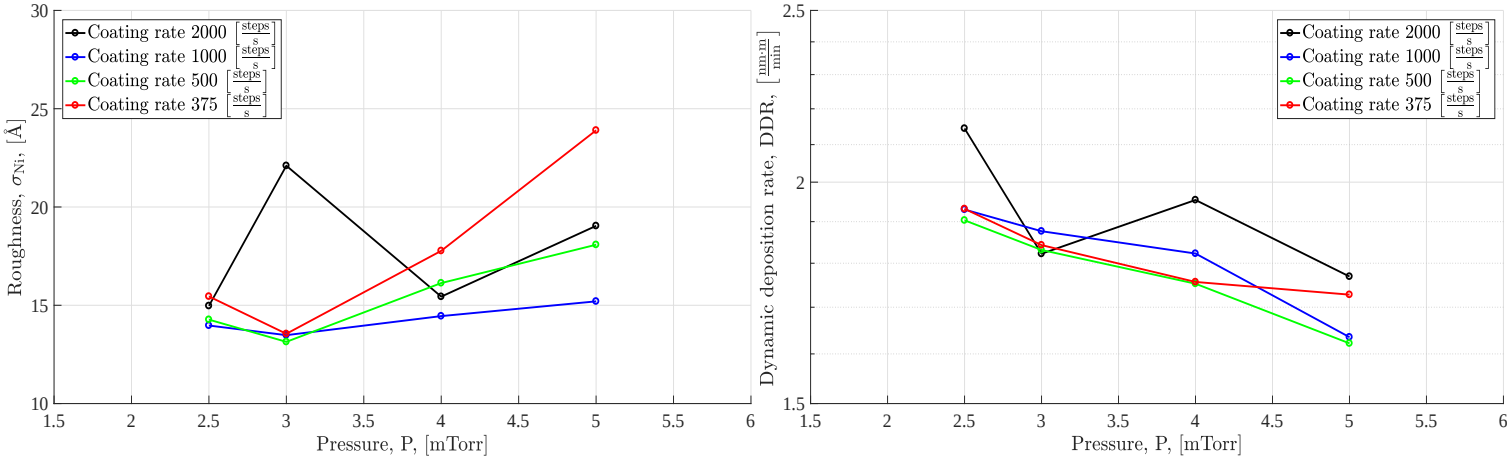


Figure 21: (a): SL coatings produced with a coating rate at 2000 [steps $\cdot$ s^{-1}] in a gas of 80% Ar and 20% N_2 for different pressures. (b): SL coatings produced with a coating rate at 1000 [steps $\cdot$ s^{-1}] in a gas of 80% Ar and 20% N_2 for different pressures. (c): SL coatings produced with a coating rate at 500 [steps $\cdot$ s^{-1}] in a gas of 80% Ar and 20% N_2 for different pressures. (d): SL coatings produced with a coating rate at 375 [steps $\cdot$ s^{-1}] in a gas of 80% Ar and 20% N_2 for different pressures.

From Figure 21 the same trend as seen in phase 1 is observed. This is, that SL's produced at low pressures have the highest reflectance, and hence the coating pressure at 2.5 [mTorr] seems to be the best. However when the roughnesses of the SL's are looked upon another result is seen.
If the SL's with an aimed thickness at 50 [Å] are excluded, it is seen from Figure 22 (a) that SL's produced at 3.0 [mTorr] have the the lowest roughneses.
In Figure 22 (b) the dinamic deposition rates of the SL's in phase 3 are shown as function of coating pressure. Again is seen that the trend from phase 1 is repeated. The DDR's of the SL's decrease with the coating pressure.



(a) **(b)**

Figure 22: (a): Roughness of Ni as function of coating pressure for all SL's produced in phase 3. (b): Dynamic deposition rate as function of coating pressure for all SL's in phase 3.

In Figure 23 the produced ML's of phase 3 can be seen. Due to time constrains ML's have only been produced at 3.0 [mTorr], which was believed to be the optimal coating pressure. ML's have been produce at 4 aimed D-spacing equal to 30, 60, 90 and 120 [Å] which can be seen in Figure 23 (a), (b), (c) and (d) respectively.

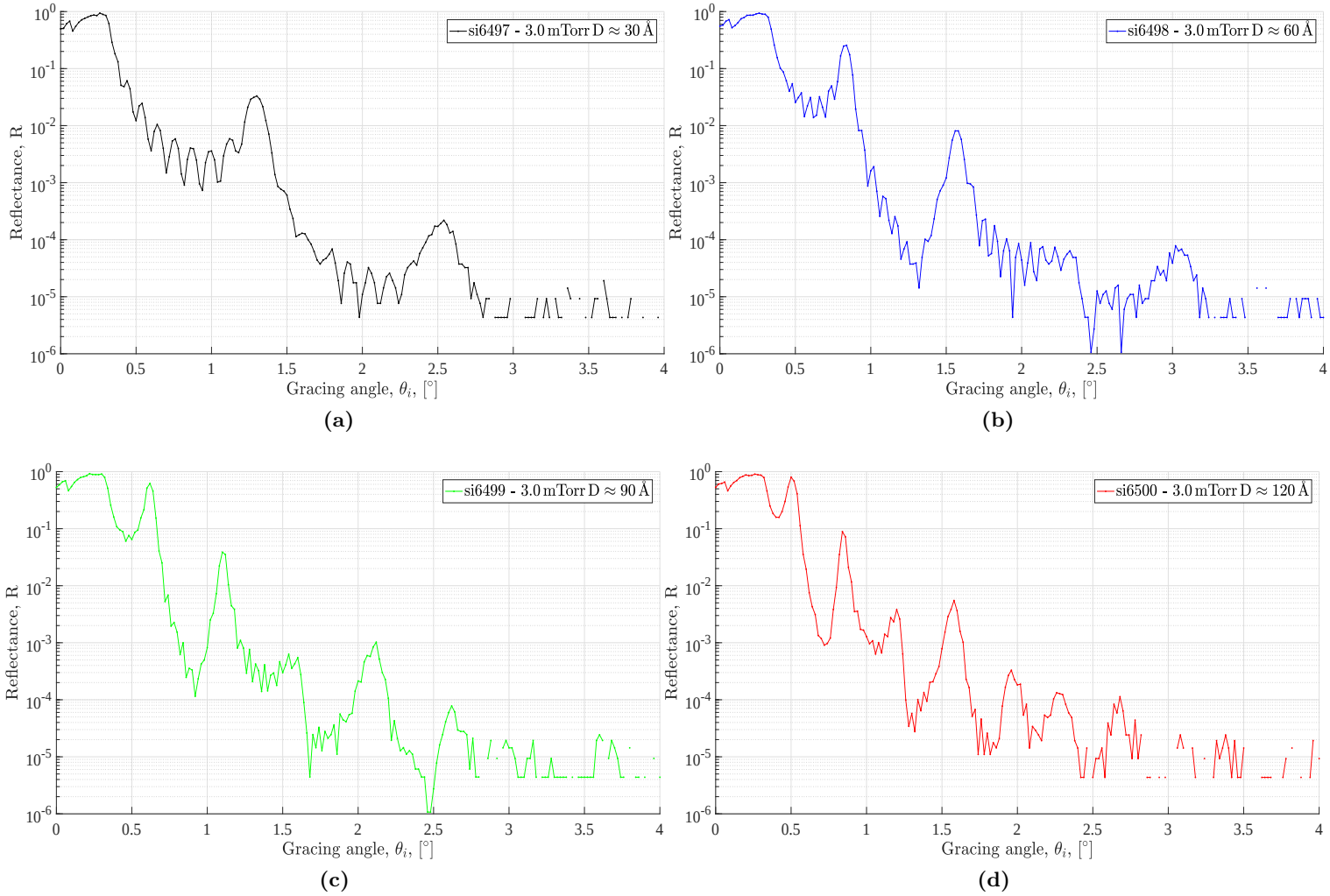
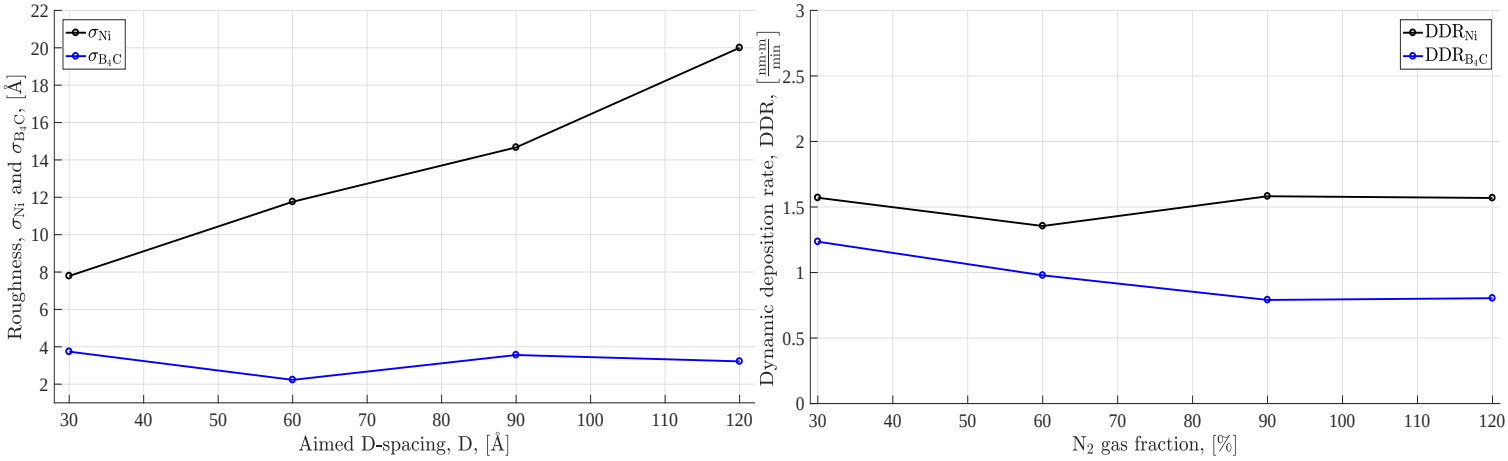


Figure 23: Reflectance as function of gracing incidence angle for all ML's in plaes 3.

In Figure 24 (a) and (b) the roughnesses and dinamic deposition rates of the ML's in phase 3 are shown. From (a) it is seen that the rhoughness of Ni increases with thickness, but the that the roughness of B_4C is almost constant.



(a) **(b)**

Figure 24:

4.4 Comparison of phases:

From this study it is evident that the roughness of nickel increases with thickness. This is found to be true when nickel is used in both SL's and ML's. Furthermore it is found that single layers of nickel have high roughness at approximately 8 [Å] or higher. This certainly limits the minimum thickness at which it is possible to produce usable thin films. From this project the minimum usable thickness of nickel is found to be approximately 40 [Å].

Comparing the SL's from all the three phases it is found that the best single layers of nickel are produced in pure argon with a coating pressure at 2.5 [mTorr]. At these conditions the average roughness of the SL's is 9.76 [Å] and the average dynamic deposition rate is 2.1832 $[\text{nm} \cdot \text{m} \cdot \text{min}^{-1}]$. For SL's of nickel it found that reactive sputtering do not have any positive effects.

From the multilayer produced during this study it is evident that reactive sputtering compared to sputtering in pure argon improve the performance of the ML's. However reactive sputtering have very little effect, if any, on the behaviour of nickel. The improved properties of the ML's when produced by reactive sputtering comes from advances on the properties of boron carbide.
When ML's are produced in pure argon at 3.0 [mTorr], boron carbide have a roughness at 7.43 [Å]. When ML's are produced by reactive sputtering with 80% argon and 20% nitrogen at 3.0 [mTorr] boron carbide have an average roughness at 3.19 [Å]. This is a 57% reduction in of the B_4C roughness.

From the ML's in phase 1 it is found that the DDR's of Ni and B_4C are almost constant for coating pressures between 2.5 and 5.0 [mTorr]. For Ni the average DDR is 1.9061 $[\text{nm} \cdot \text{m} \cdot \text{min}^{-1}]$ and for B_4C the average DDR is 0.4068 $[\text{nm} \cdot \text{m} \cdot \text{min}^{-1}]$.
For the ML's in phase 3, the average DDR of Ni and B_4C are 1.5191 $[\text{nm} \cdot \text{m} \cdot \text{min}^{-1}]$ and 0.9521 $[\text{nm} \cdot \text{m} \cdot \text{min}^{-1}]$ respectively.
By this dynamic deposition rate of Ni decreases with approximately 20% and the dynamic deposition rate of B_4C increases with proximately 134% when ML's are produced by reactive sputtering compared to sputtering in pure argon.

5 Conclusion

This thesis have studied the properties and behaviour of nickel and boron carbide for the assessment of their applicability as materials for high energy X-ray mirrors. Under different coating pressures and gas compositions, thin films of single layers and thin films of multilayers have been produced and compared.

It is found that the roughness of nickel increases with the thickness of the sputtered layer. This is evident for both single and multilayers. Furthermore, the results indicate that single layers of nickel have a high degree of roughness at approximately 8 [Å] or higher. This limits the minimum degree of thickness of which it is possible to produce usable thin films. From this project the minimum thickness of nickel, for which the sputtered layer is usable, is found to be approximately 40 [Å].

Comparing the single layers from all three phases, it is evident that the best single layers of nickel are produced in pure argon with a coating pressure at 2,5 [mTorr]. Contrary to the expected, reactive sputtering does not have any positive effects for single layers of nickel.

Reactive sputtering compared to sputtering in pure argon does, however, improve the performance of multilayers. It is found that the best multilayer of Ni and B_4C is produced with a gas fraction of argon and nitrogen at 80/20 at a coating pressure equal to 3.0 [mTorr]. Reactive sputtering has very little effect on the behaviour of nickel. The improved properties of the multi-layers when produced by reactive sputtering comes from advantages on the properties of boron carbide.

References

[1] Alexander I. Lvovsky. "Fresnel equations". In: *Encyclopedia of Optical Engineering* (2013).

[2] Richard Fitzpatrick. "Reflection at a dielectric boundary". In: (2014). URL: `http://farside.ph.utexas.edu/teaching/em/lectures/node104.html`.

[3] David L. Windt. "IMD - Software for modeling the optical properties of multilayer films". In: *Computers in physics* (1998).

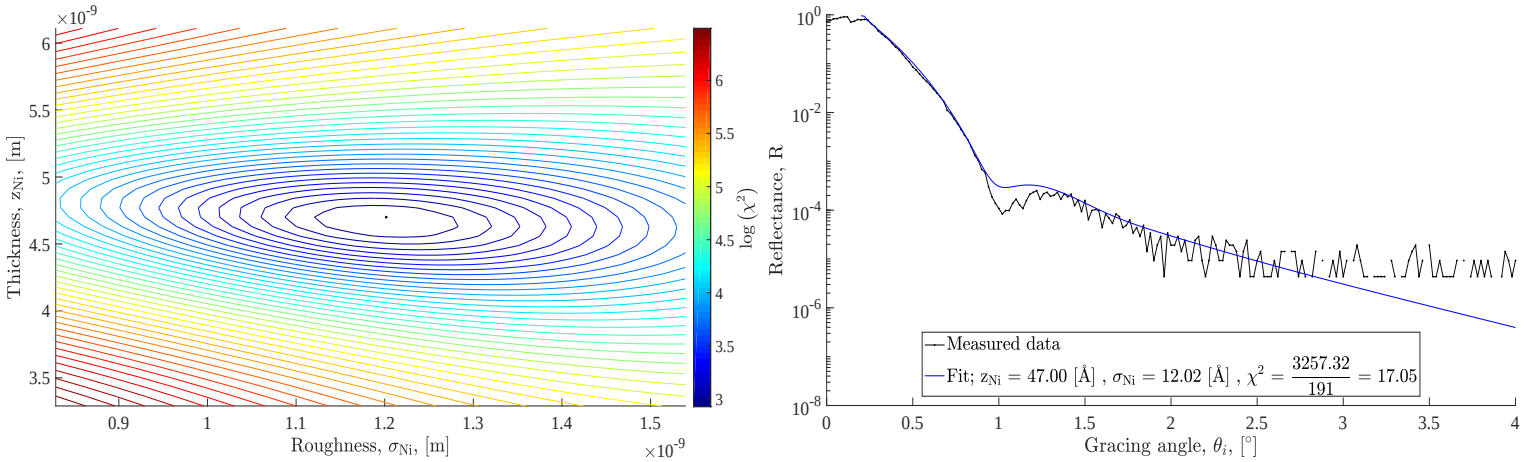
[4] David Girou. "Investigation of Ni-based multilayers for hard X-ray mirrors for high energy astrophysics telescopes". In: (2015).

[5] Anders C. Jacobsen. "X-ray optics in new insturments for astro- and astroparticle physics". In: (2016).

Appendices

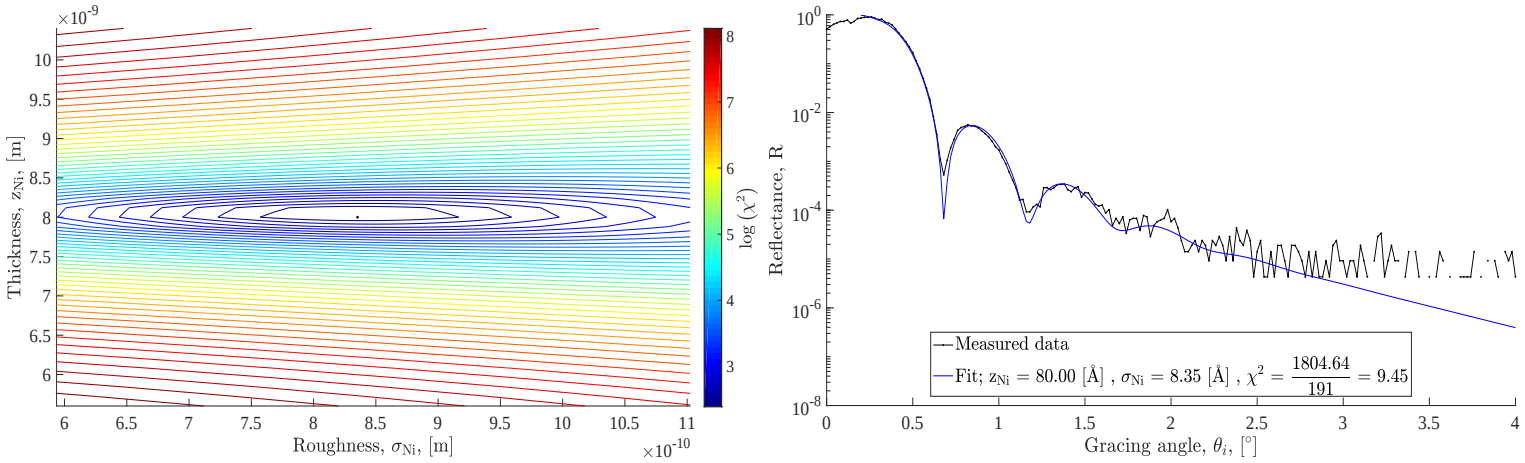
A Fits of reflections

A.1 Phase 1 - Coatings in 100% Ar at 2.5 mTorr



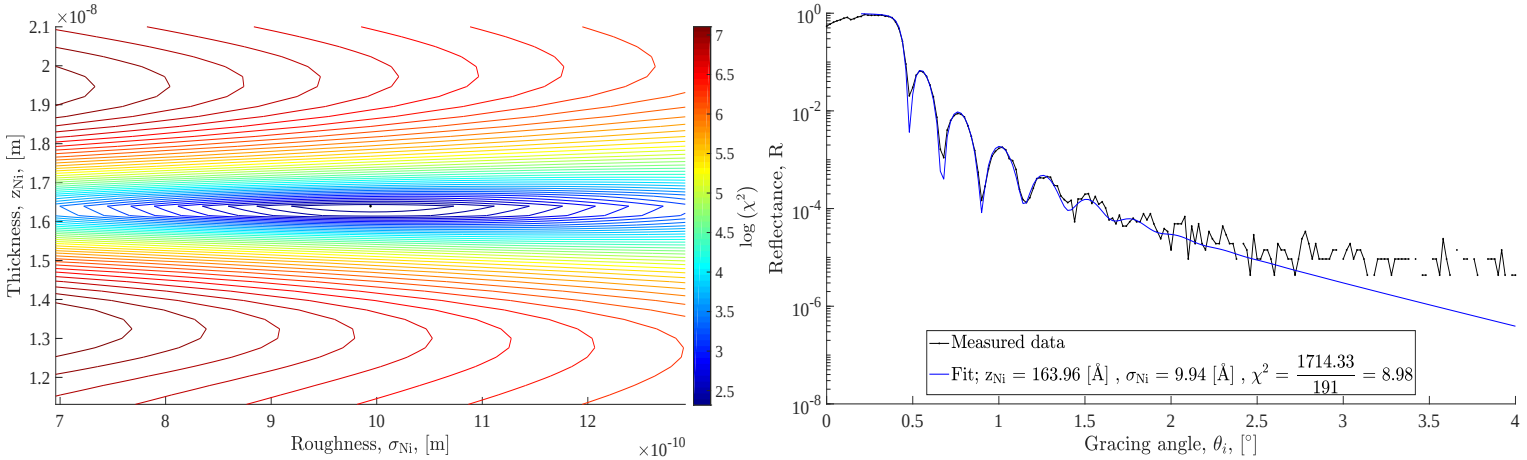
(a) (b)

Figure 25: (a): si6433_3_contour. (b): si6433_3_plot.



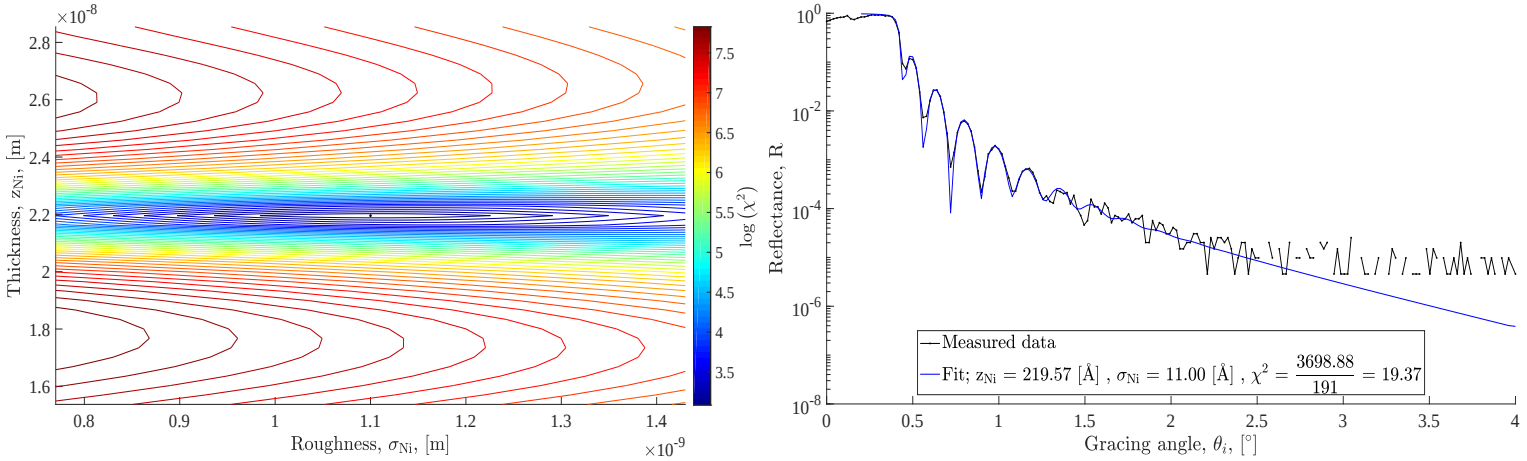
(a) (b)

Figure 26: (a): si6434_6_contour. (b): si6434_6_plot.



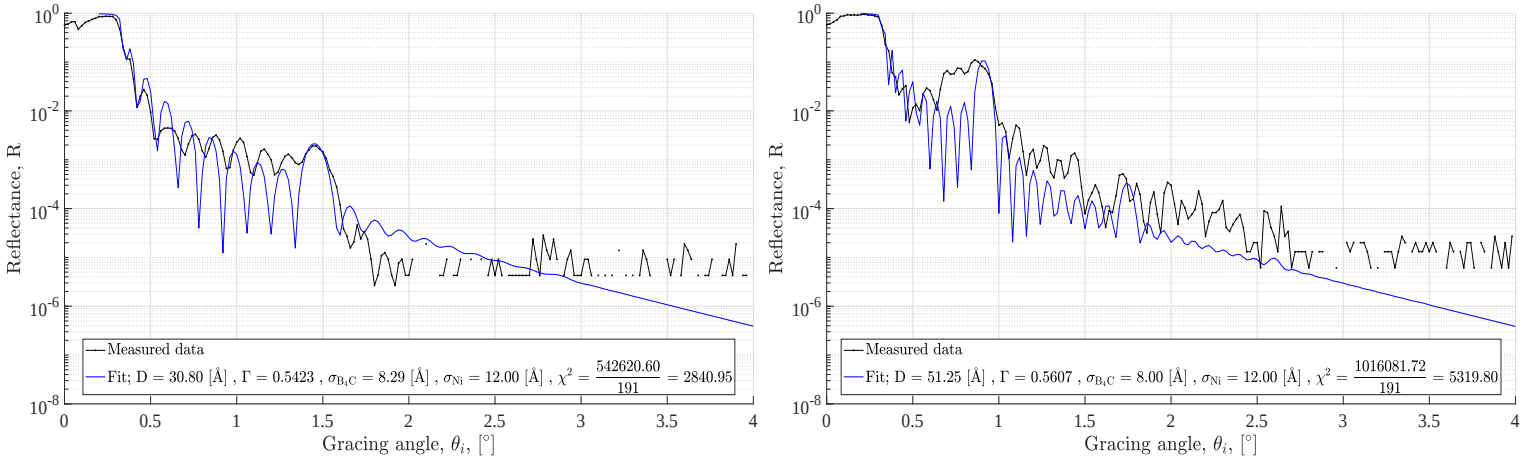
(a) **(b)**

Figure 27: (a): si6435_7_contour. (b): si6435_7_plot.



(a) **(b)**

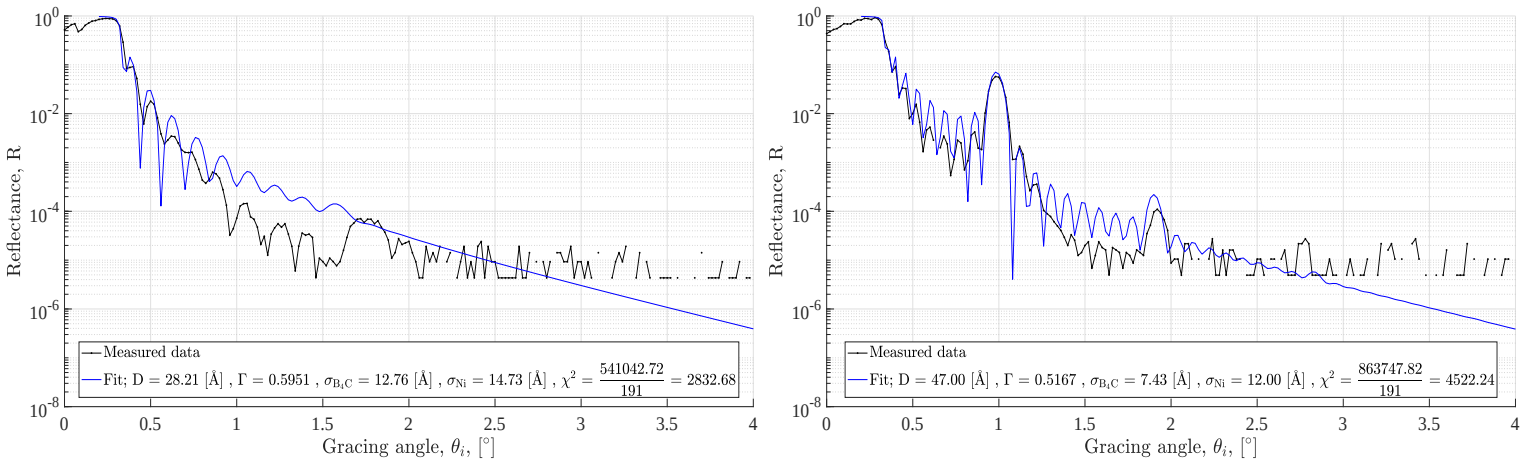
Figure 28: (a): si6436_3_contour. (b): si6436_3_plot.



(a) **(b)**

Figure 29: (a): si6452_3_UnCorPlot. (b): si6449_5_UnCorPlot.

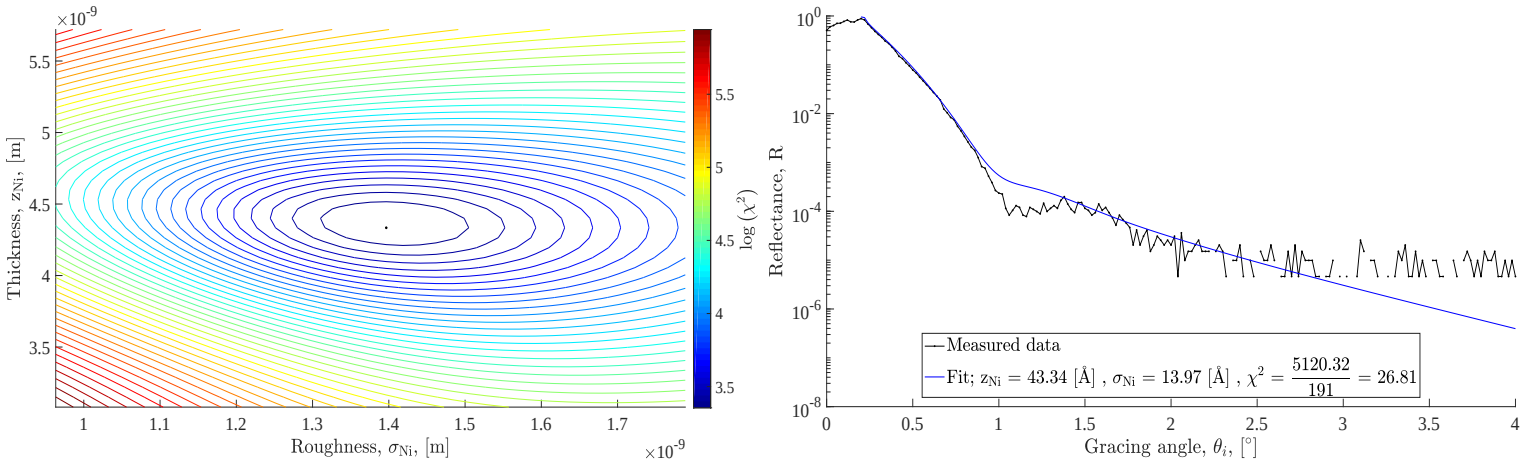
A.2 Phase 1 - Coatings in 100% Ar at 3.0 mTorr



(a) **(b)**

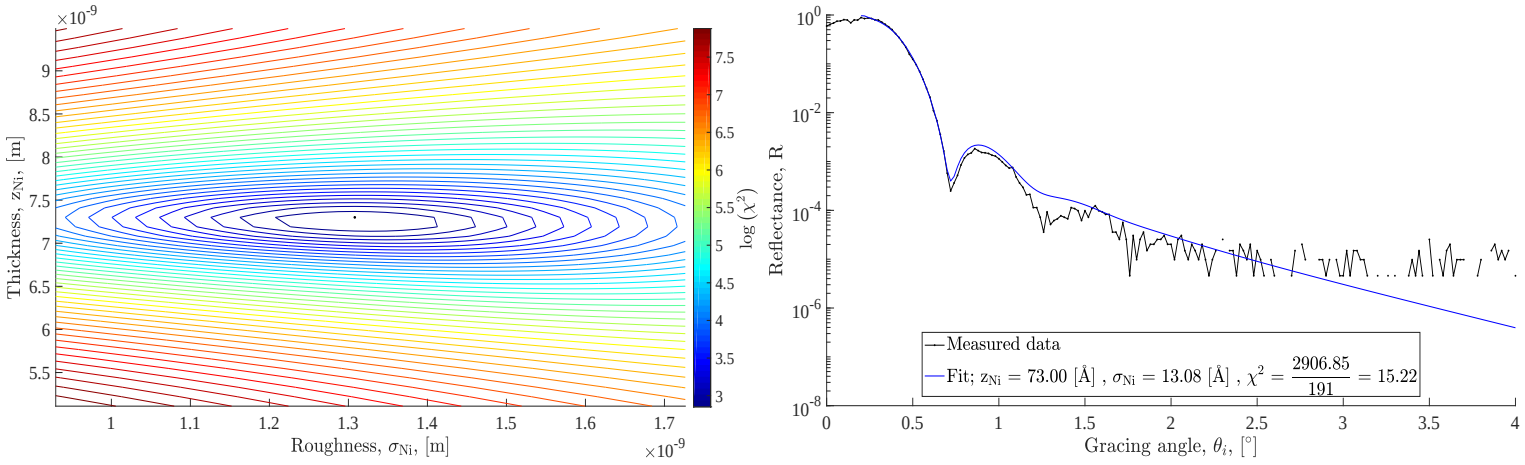
Figure 30: (a): si6450_2_UnCorPlot. (b): si6451_2_UnCorPlot.

A.3 Phase 1 - Coatings in 100% Ar at 4.0 mTorr



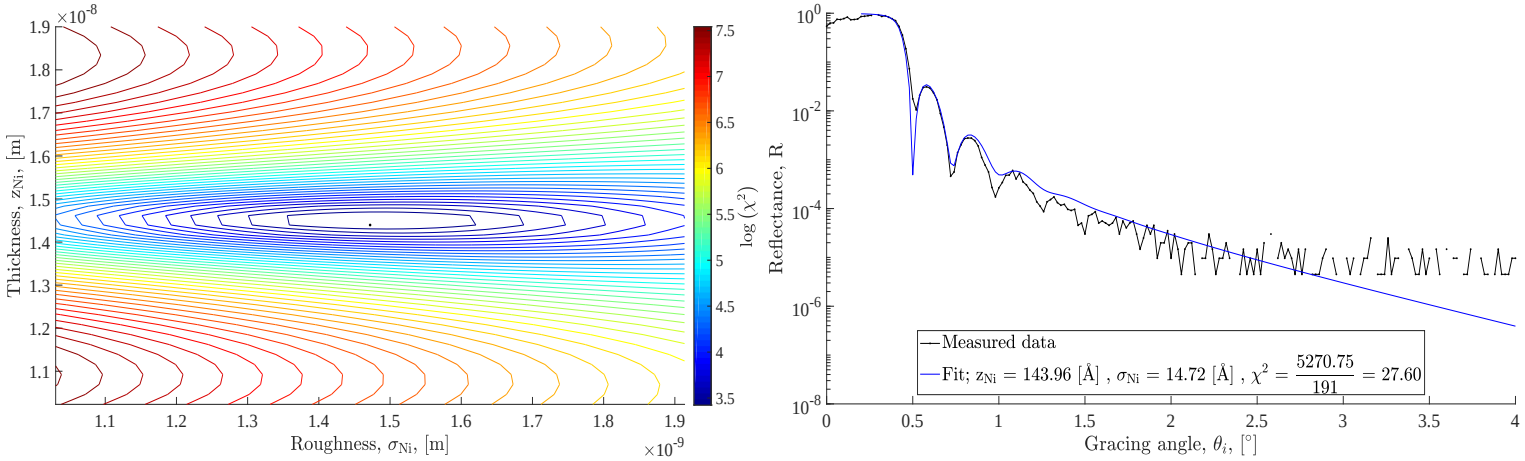
(a) **(b)**

Figure 31: (a): si6437_5_contour. (b): si6437_5_plot.



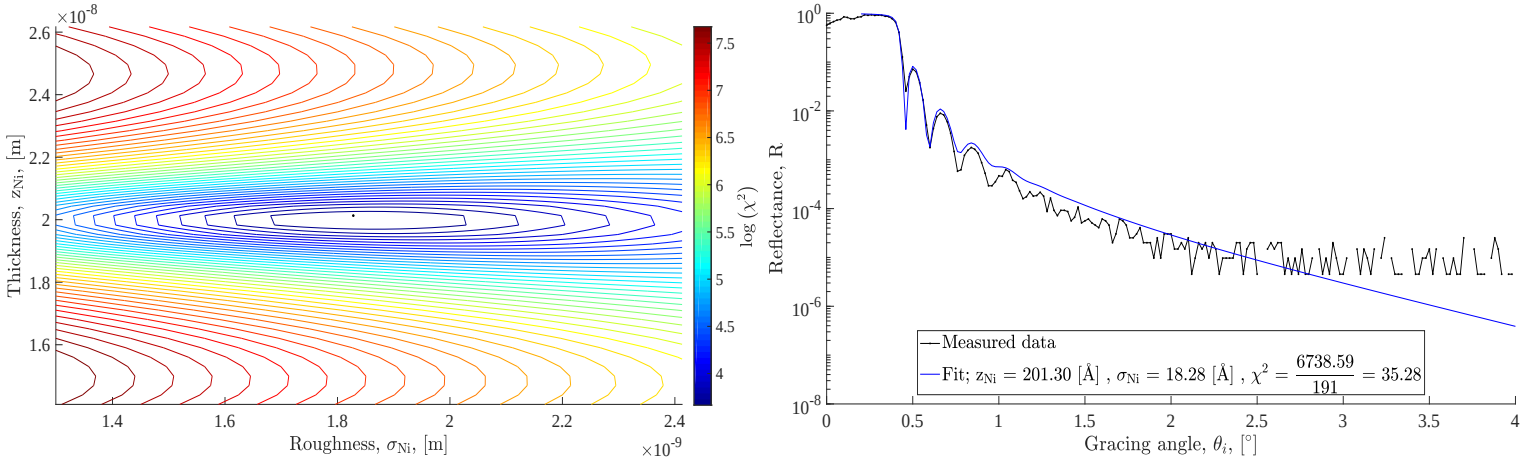
(a) **(b)**

Figure 32: (a): si6438_2_contour. (b): si6438_2_plot.



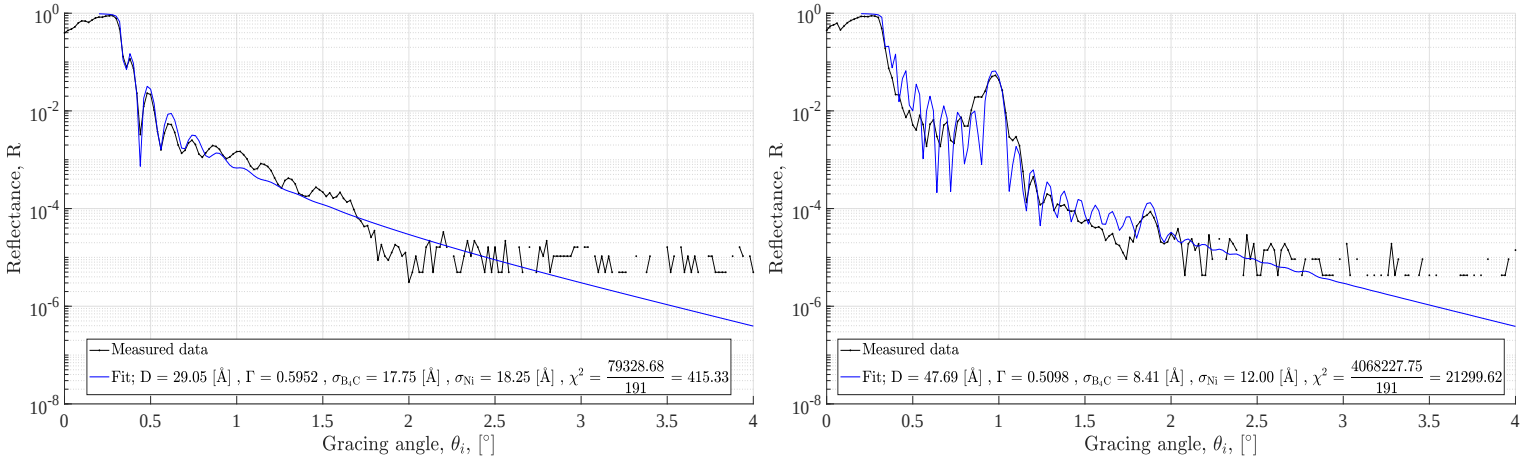
(a) **(b)**

Figure 33: (a): si6439_4_contour. (b): si6439_4_plot.



(a) **(b)**

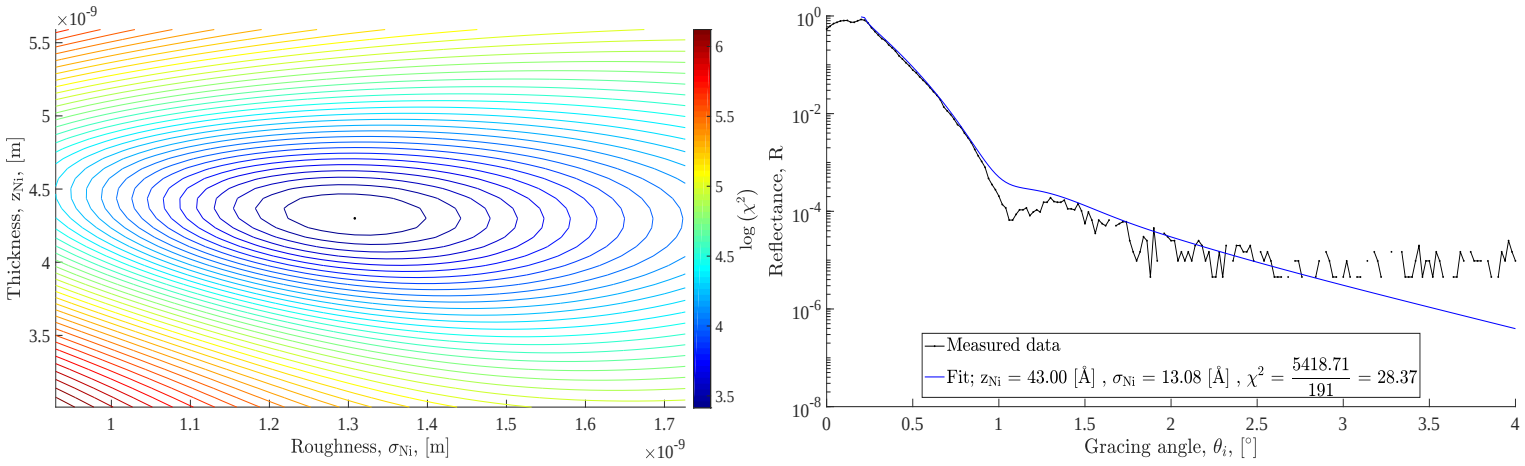
Figure 34: (a): si6440_4_contour. (b): si6440_4_plot.



(a) **(b)**

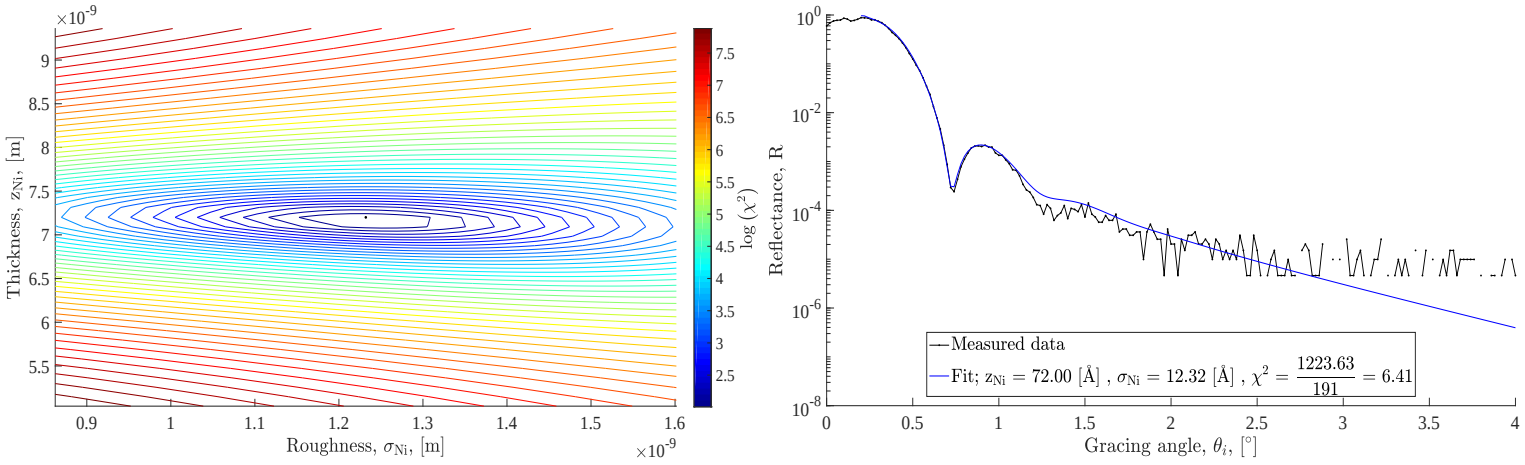
Figure 35: (a): si6453_2_UnCorPlot. (b): si6456_2_UnCorPlot.

A.4 Phase 1 - Coatings in 100% Ar at 5.0 mTorr



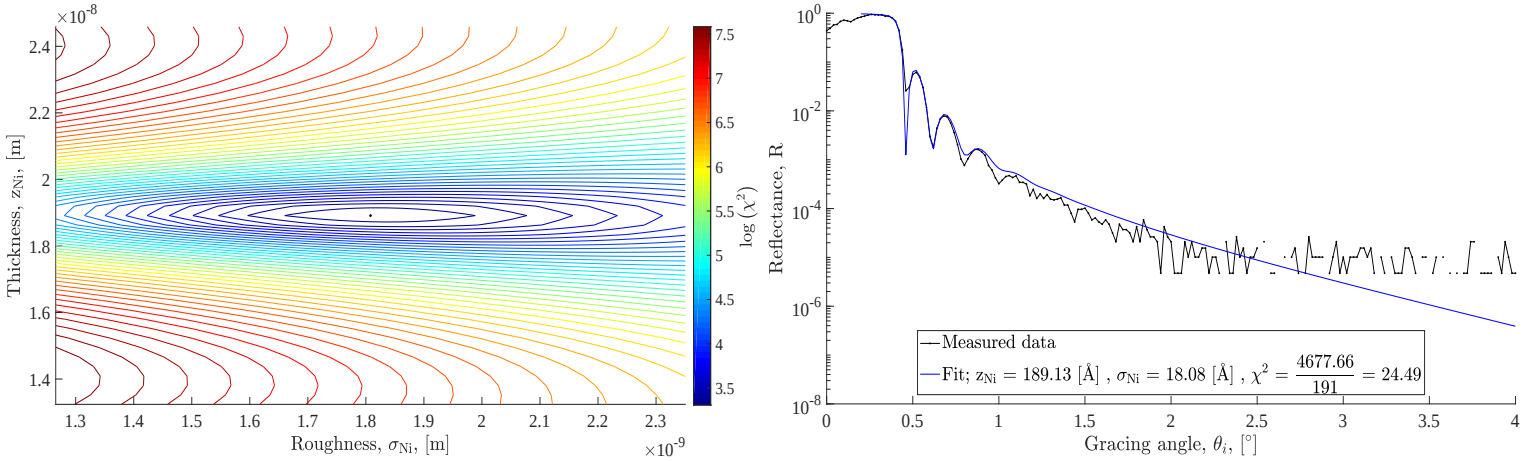
(a) (b)

Figure 36: (a): si6441_1_contour. (b): si6441_1_plot.



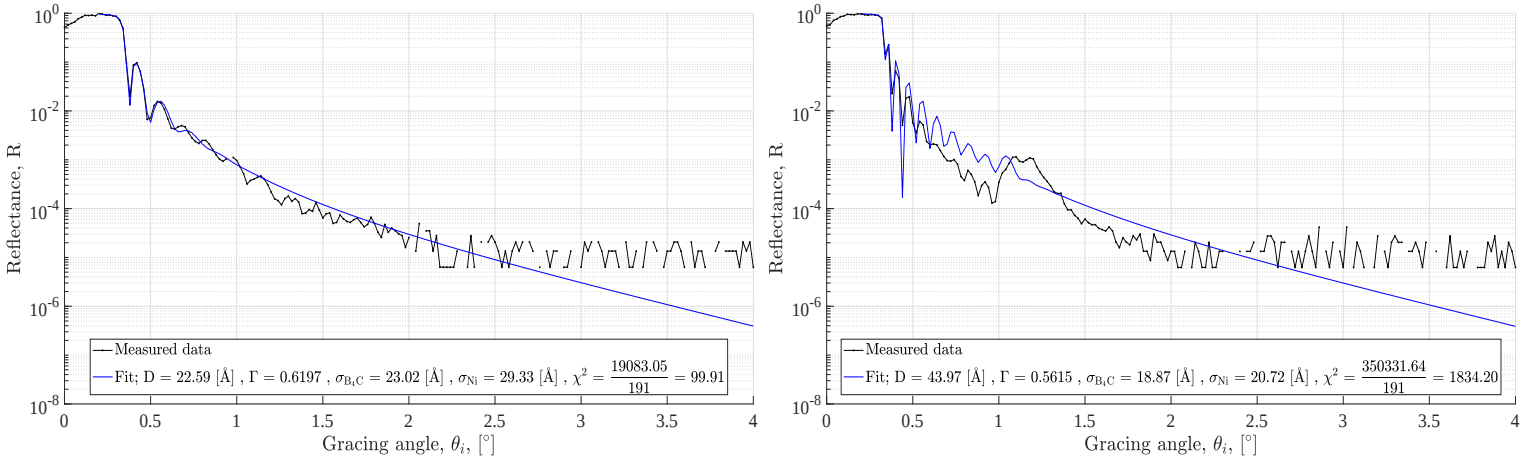
(a) (b)

Figure 37: (a): si6442_2_contour. (b): si6442_2_plot.



(a) **(b)**

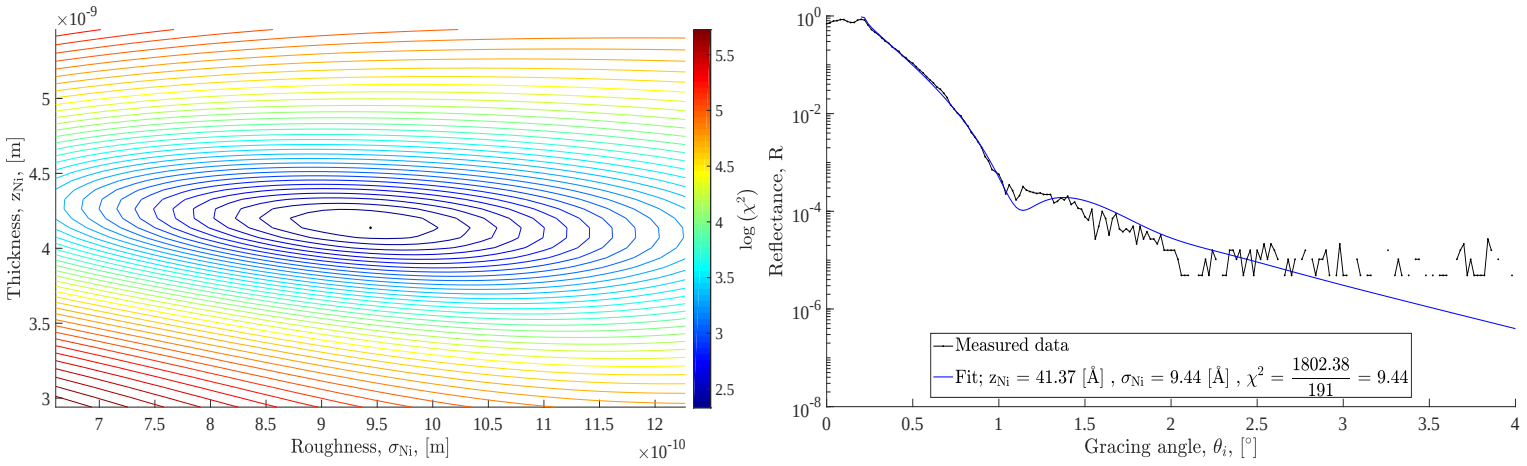
Figure 38: (a): si6444_3_contour. (b): si6444_3_plot.



(a) **(b)**

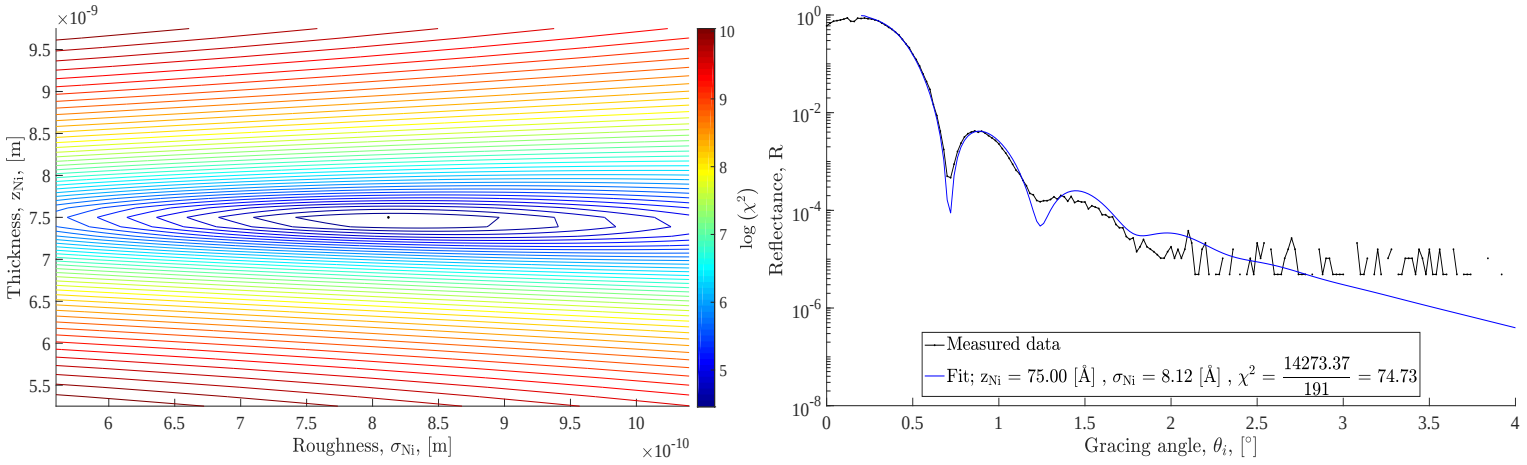
Figure 39: (a): si6444_3_contour. (b): si6444_3_plot.

A.5 Phase 2 - Coatings in $5\%\,N_2$ at 3.0 mTorr



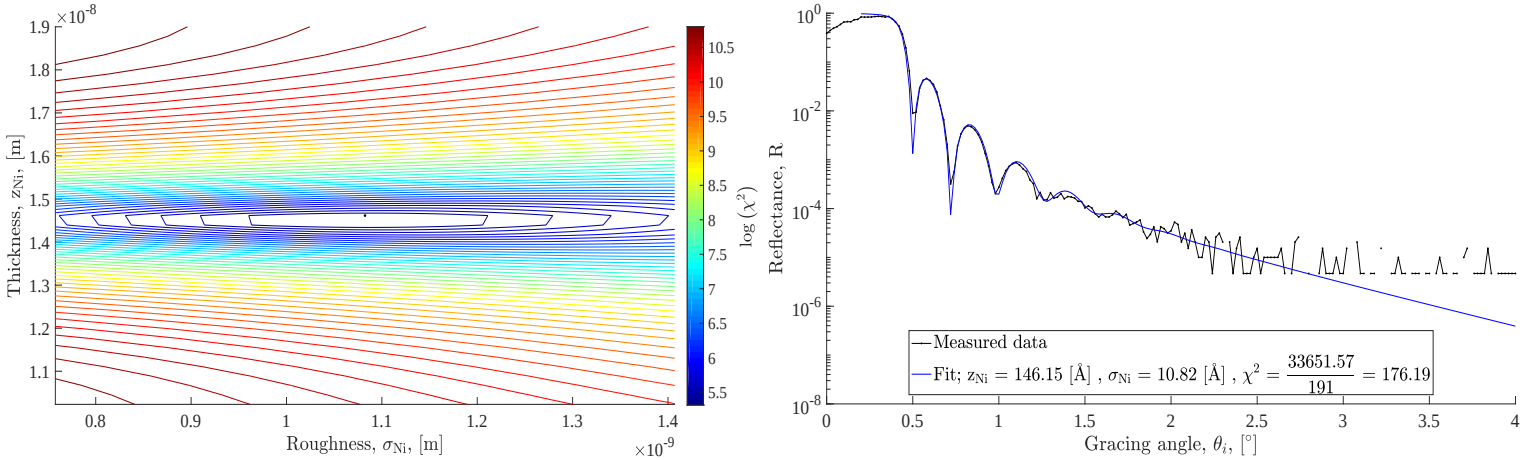
(a) **(b)**

Figure 40: (a): si6457_2_contourZS. (b): si6457_2_plot.



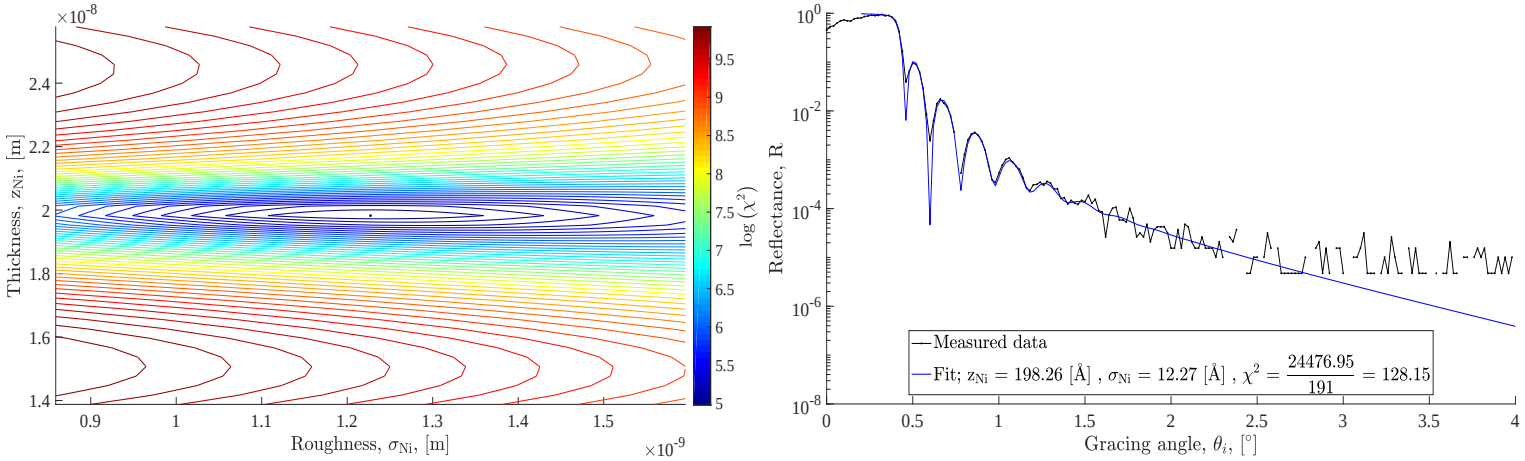
(a) **(b)**

Figure 41: (a): si6460_3_contourZS. (b): si6460_3_plot.



(a) **(b)**

Figure 42: (a): si6459_2_contourZS. (b): si6459_2_plot.



(a) **(b)**

Figure 43: (a): si6458_2_contourZS. (b): si6458_2_plot.

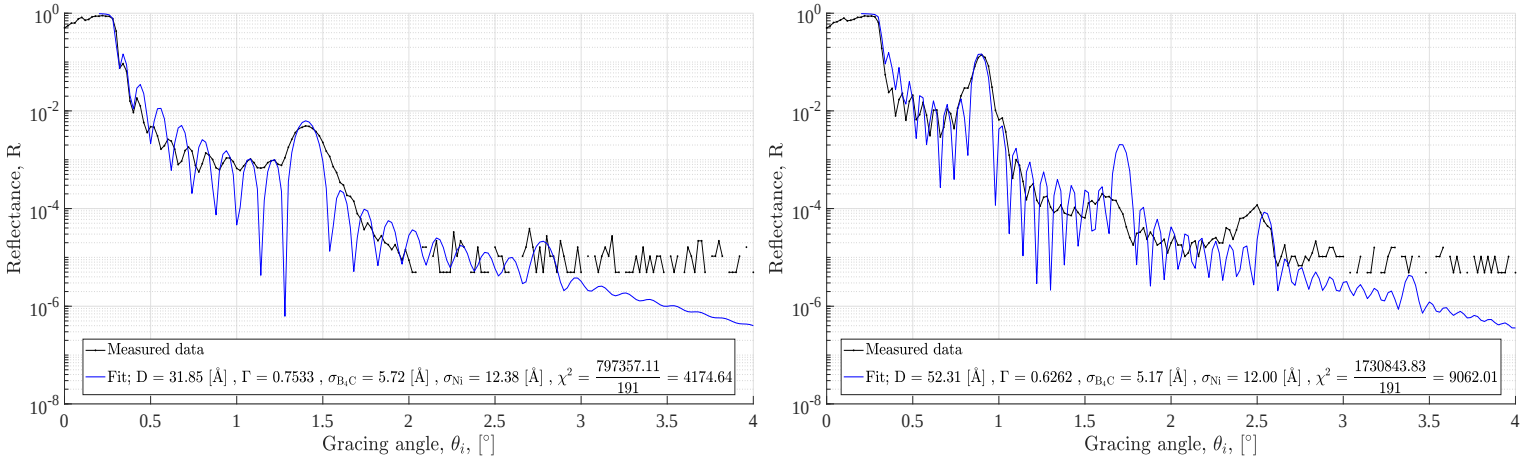
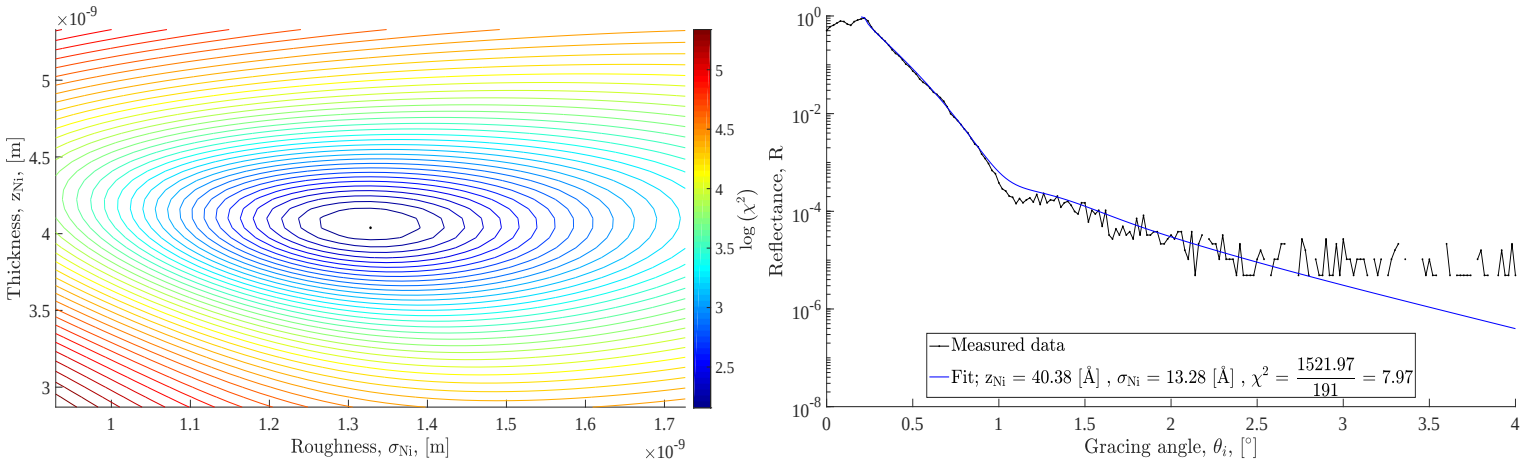


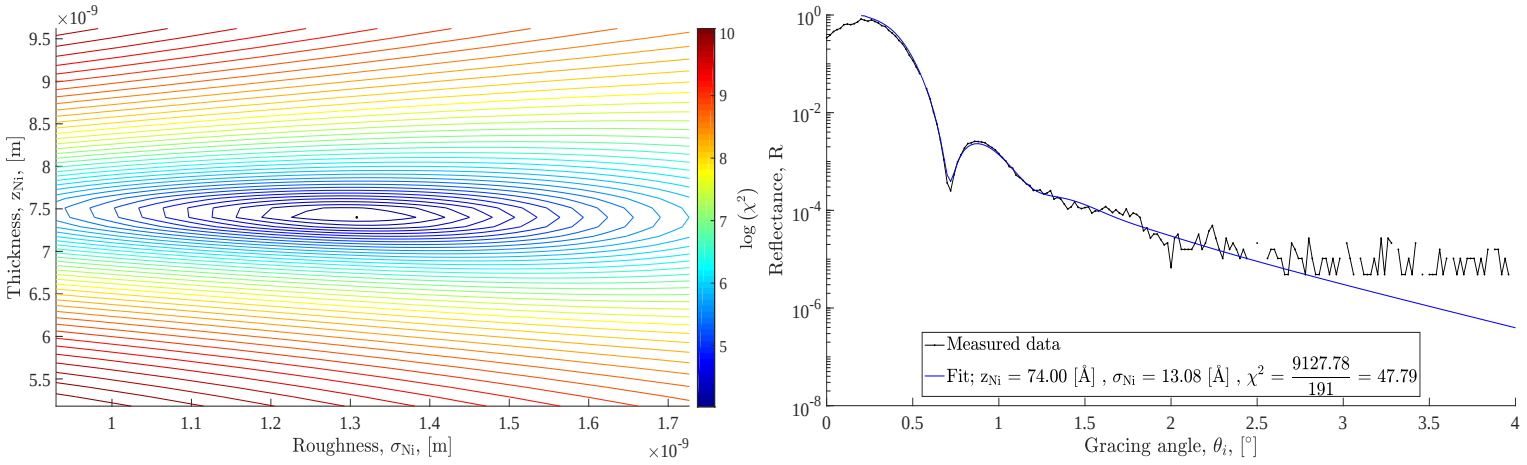
Figure 44: (a): si6473_2_UnCorPlot. (b): si6476_4_UnCorPlot.

A.6 Phase 2 - Coatings in $15\%\,N_2$ at 3.0 mTorr



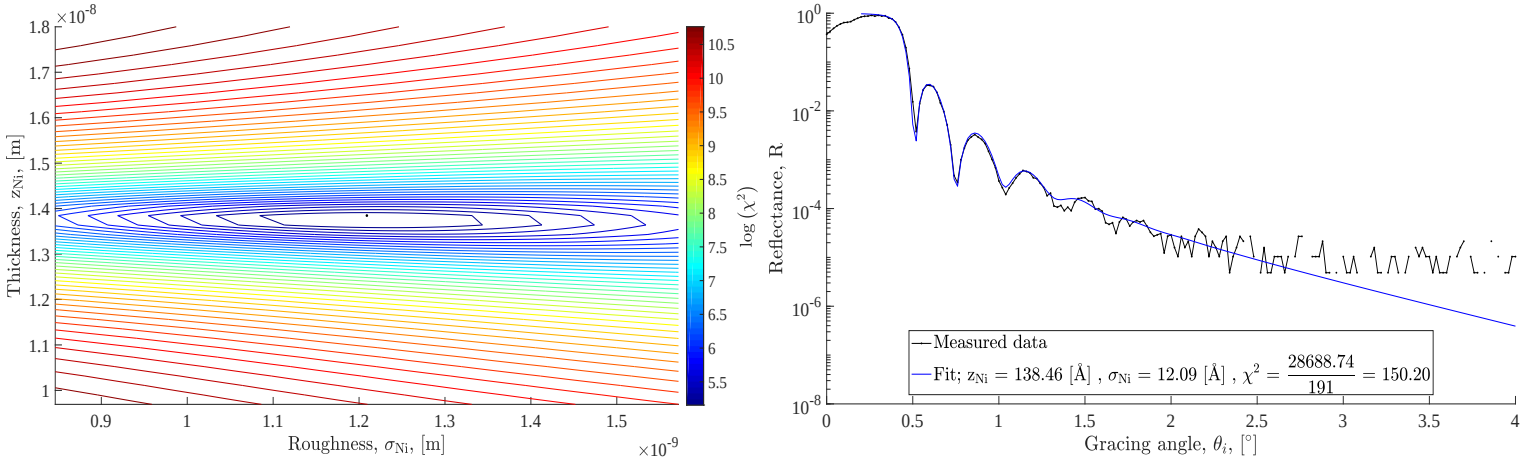
(a) **(b)**

Figure 45: (a): si6461_1_contourZS. (b): si6461_1_plot.



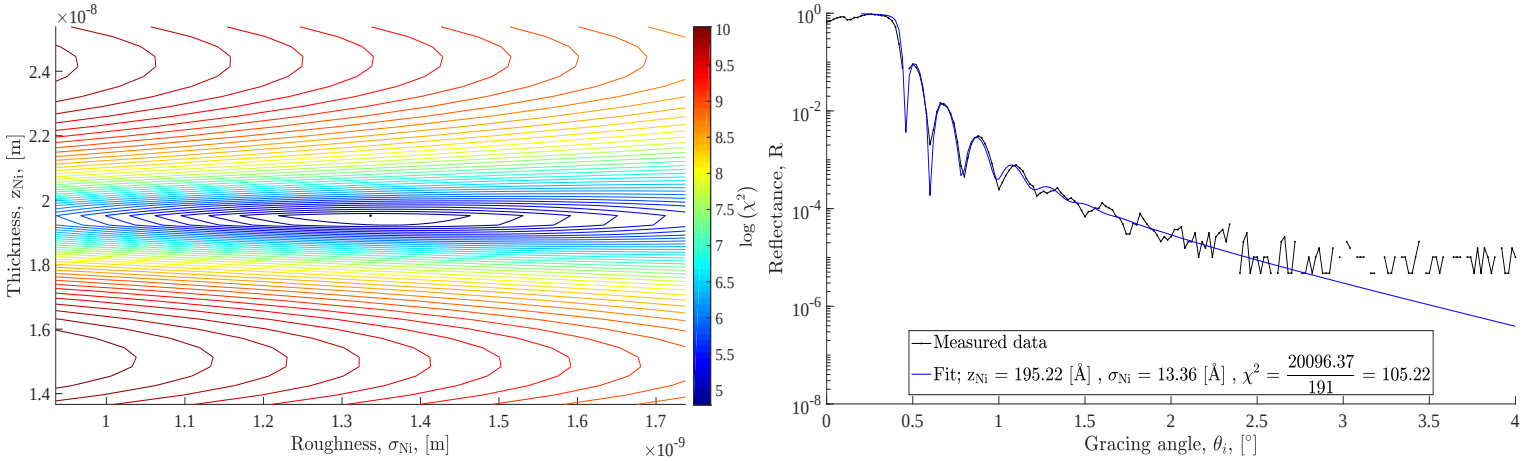
(a) **(b)**

Figure 46: (a): si6464_2_contourZS. (b): si6464_2_plot.



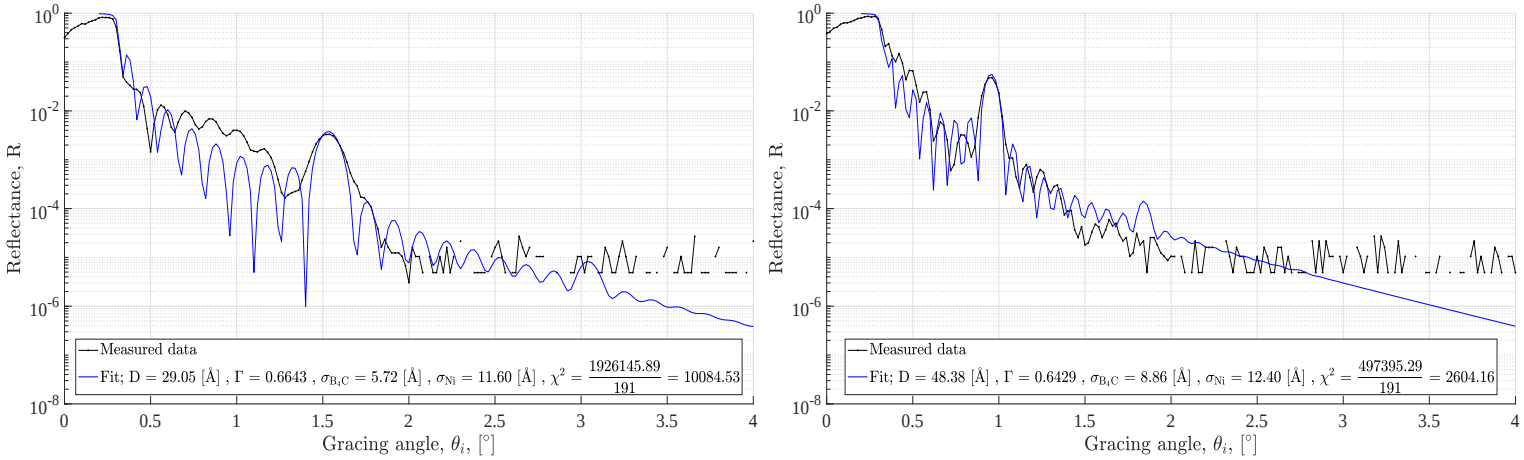
(a) **(b)**

Figure 47: (a): si6463_2_contourZS. (b): si6463_2_plot.



(a) **(b)**

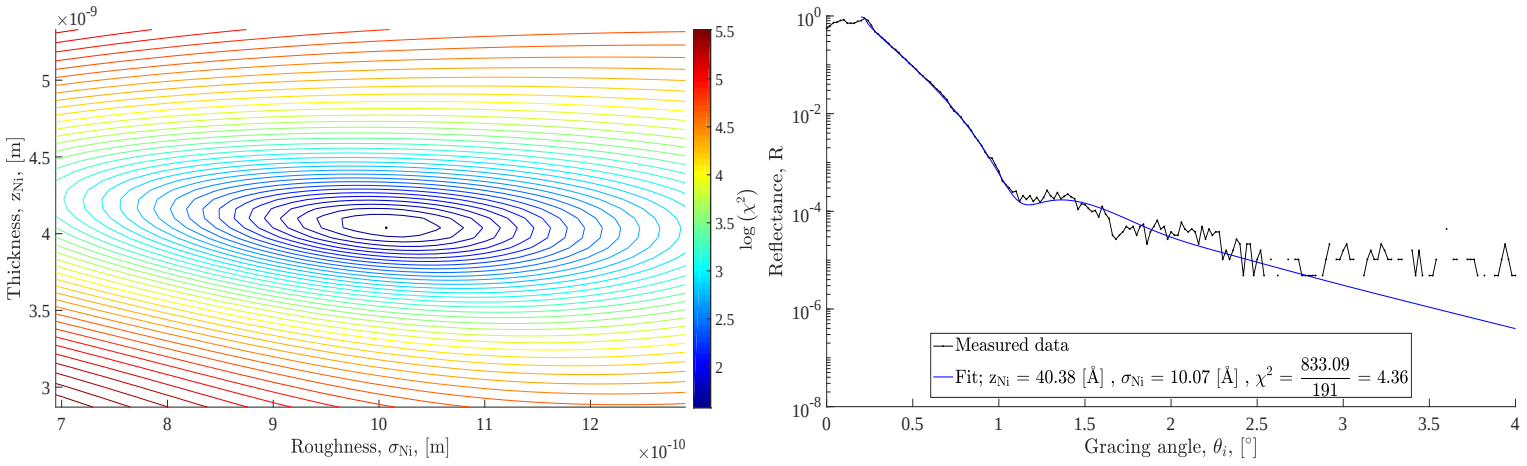
Figure 48: (a): si6462_1_contourZS. (b): si6462_1_plot.



(a) **(b)**

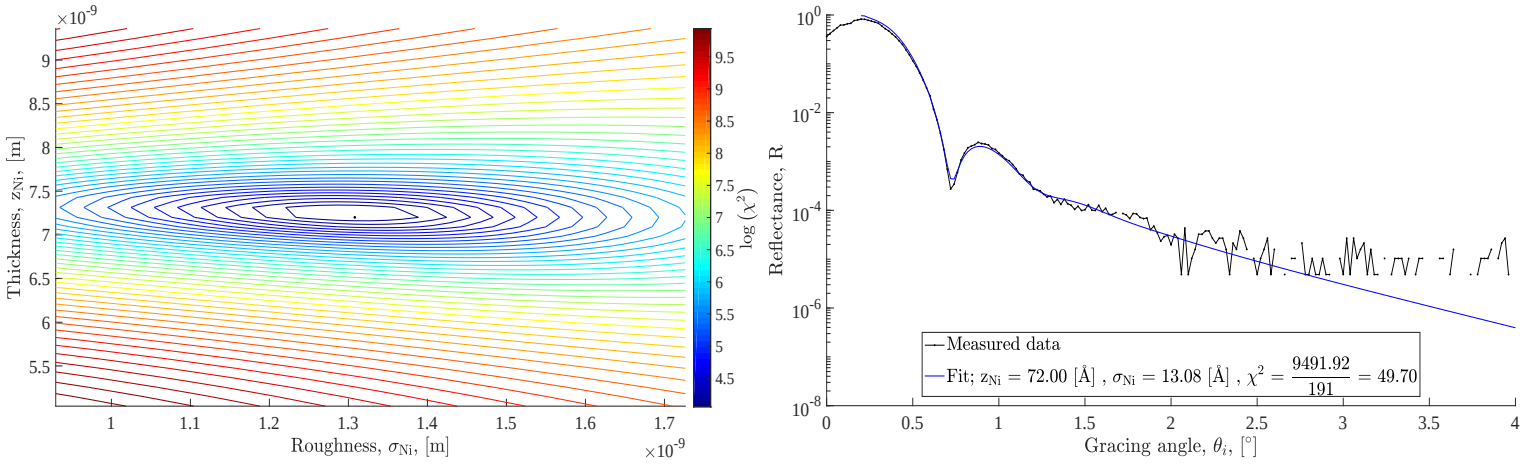
Figure 49: (a): si6475_2_UnCorPlot. (b): si6474_2_UnCorPlot.

A.7 Phase 2 - Coatings in $20\%\,N_2$ at 3.0 mTorr



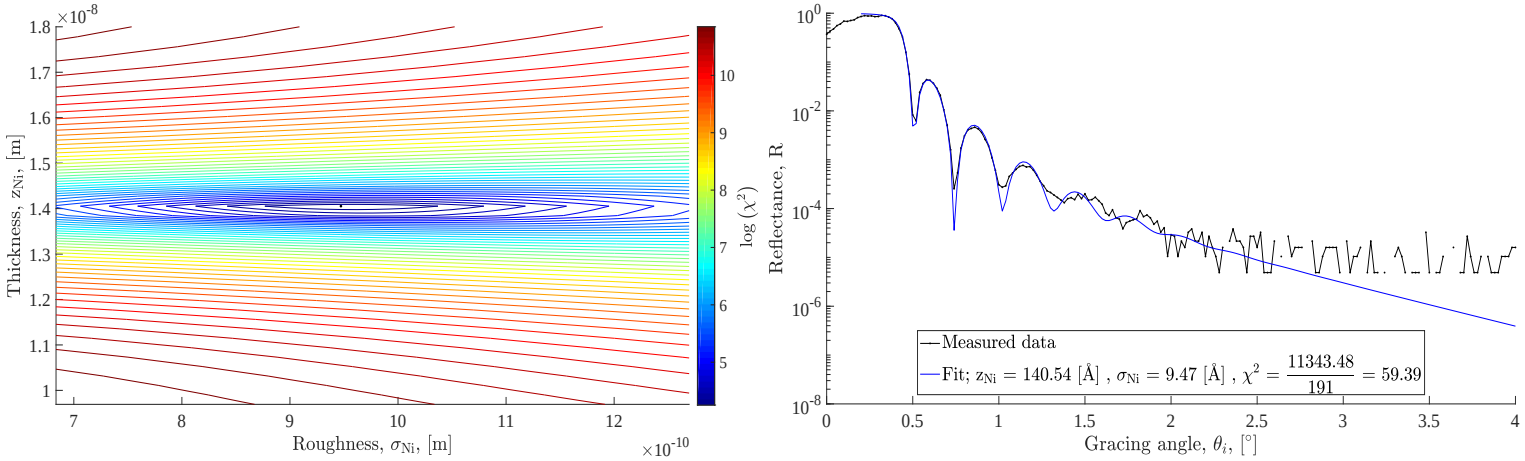
(a) **(b)**

Figure 50: (a): si6465_1_contourZS. (b): si6465_1_plot.



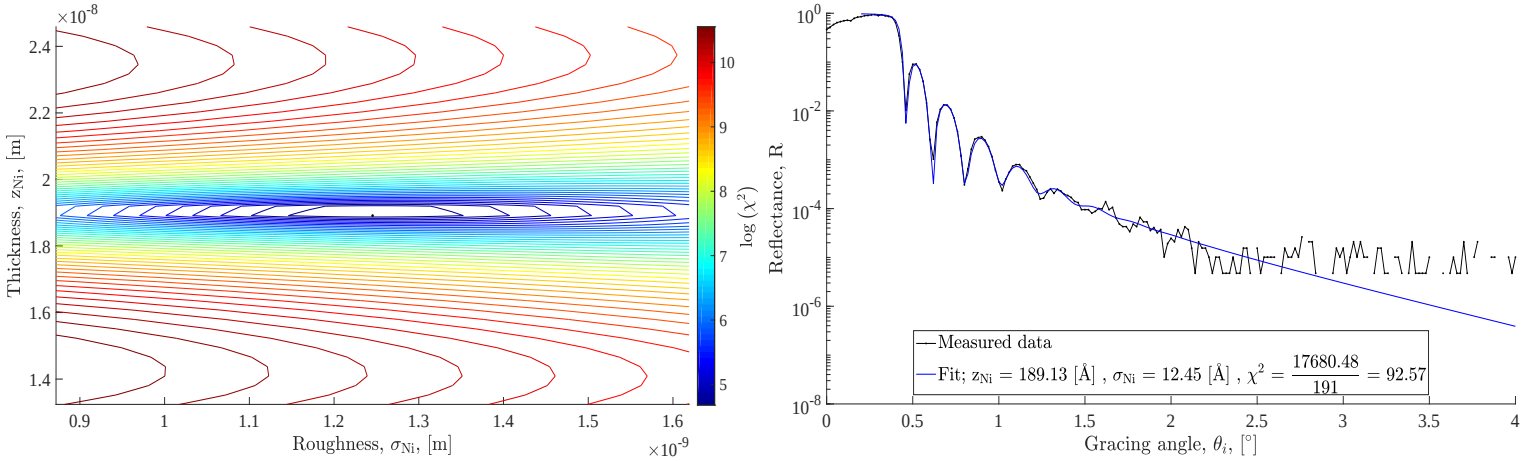
(a) **(b)**

Figure 51: (a): si6468_2_contourZS. (b): si6468_2_plot.



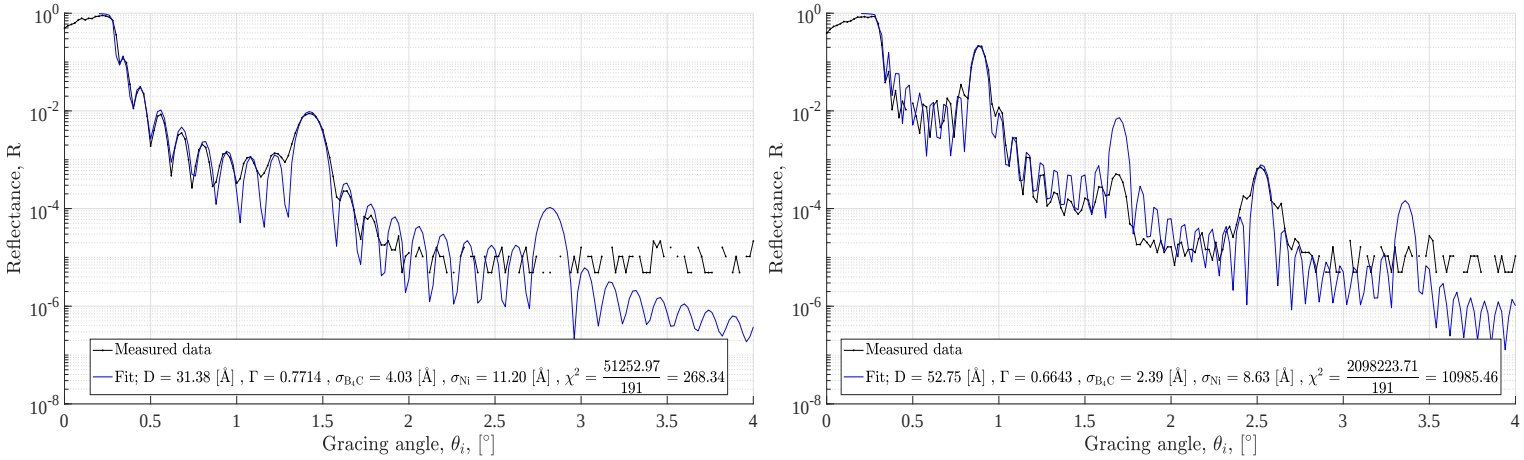
(a) **(b)**

Figure 52: (a): si6467_2_contourZS. (b): si6467_2_plot.



(a) **(b)**

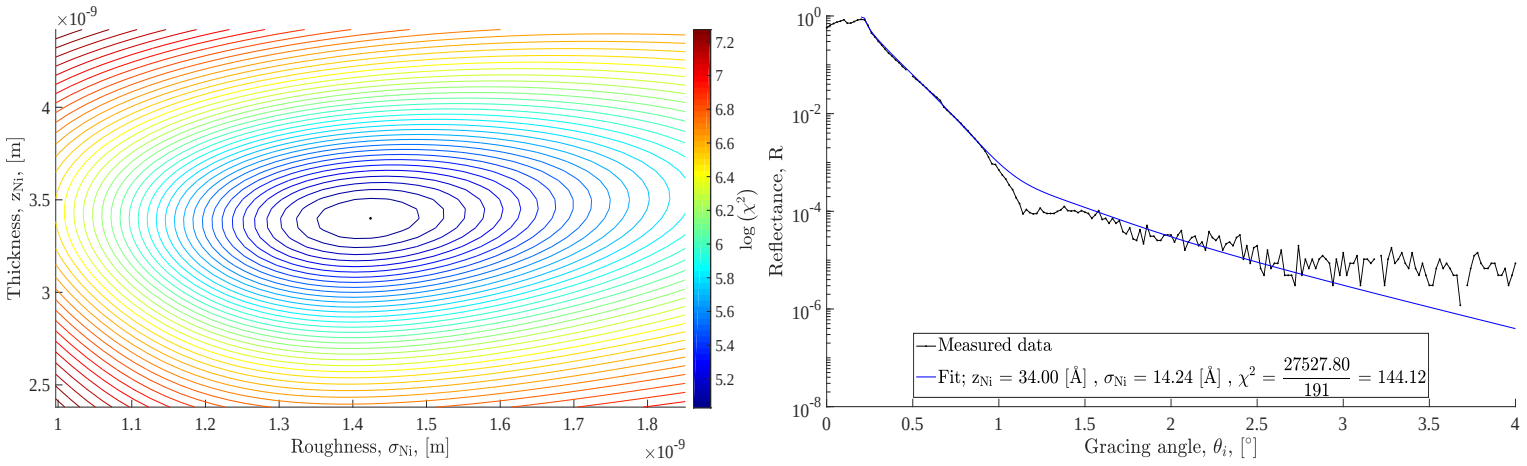
Figure 53: (a): si6466_2_contourZS. (b): si6466_2_plot.



(a) **(b)**

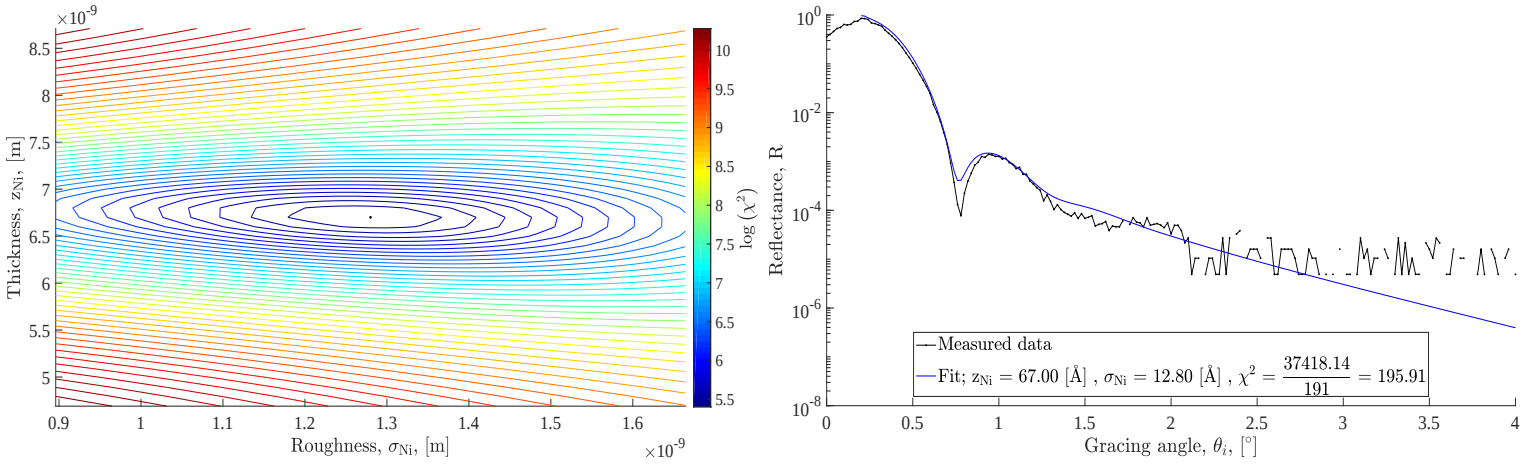
Figure 54: (a): si6477_2_UnCorPlot. (b): si6478_3_UnCorPlot.

A.8 Phase 2 - Coatings in $30\%\,N_2$ at 3.0 mTorr



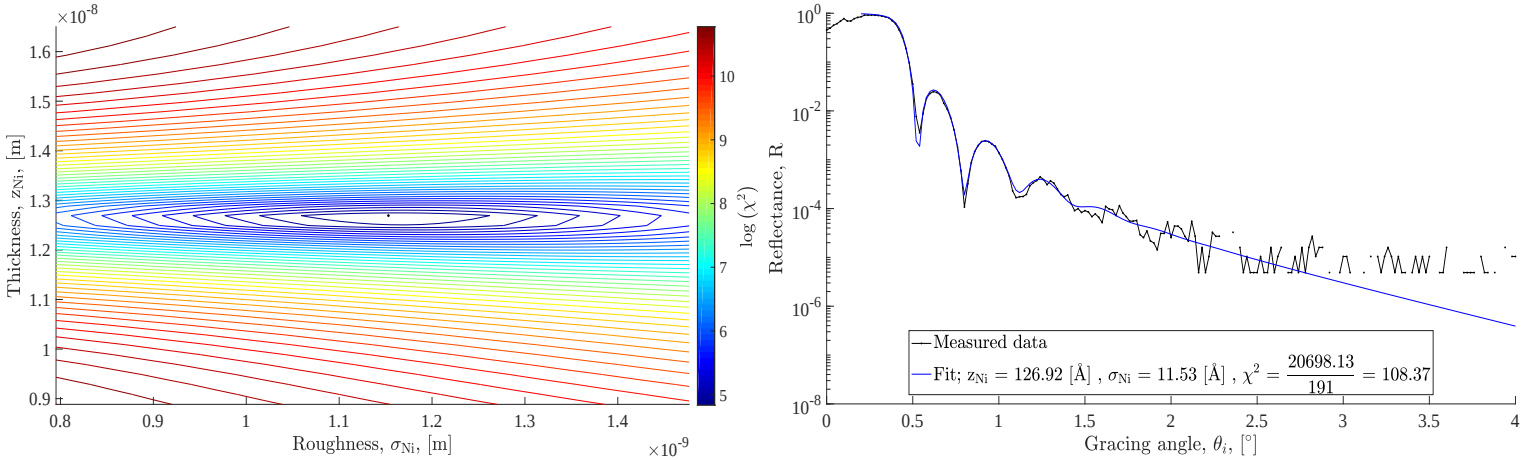
(a) **(b)**

Figure 55: (a): si6469_3_contourZS. (b): si6469_3_plot.



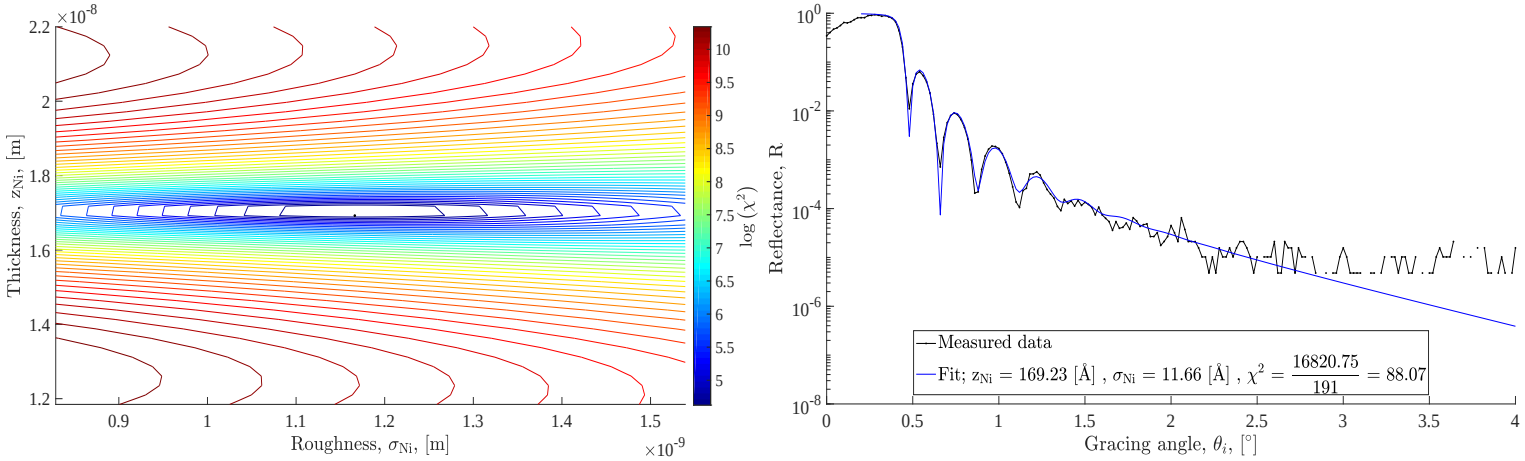
(a) **(b)**

Figure 56: (a): si6472_2_contourZS. (b): si6472_2_plot.



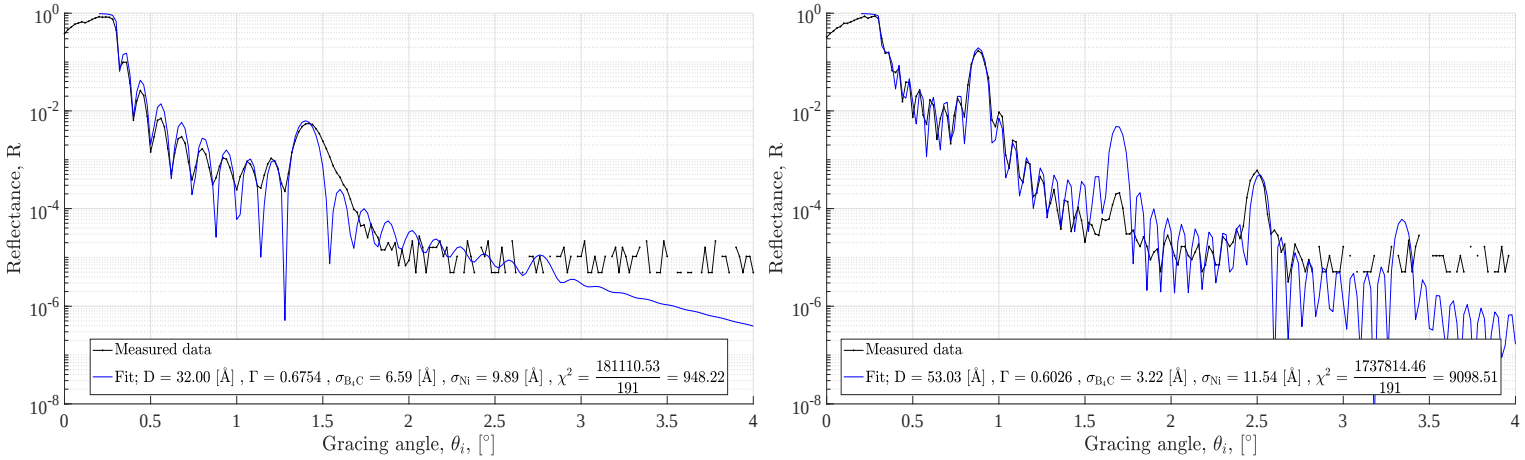
(a) **(b)**

Figure 57: (a): si6471_2_contourZS. (b): si6471_2_plot.



(a) **(b)**

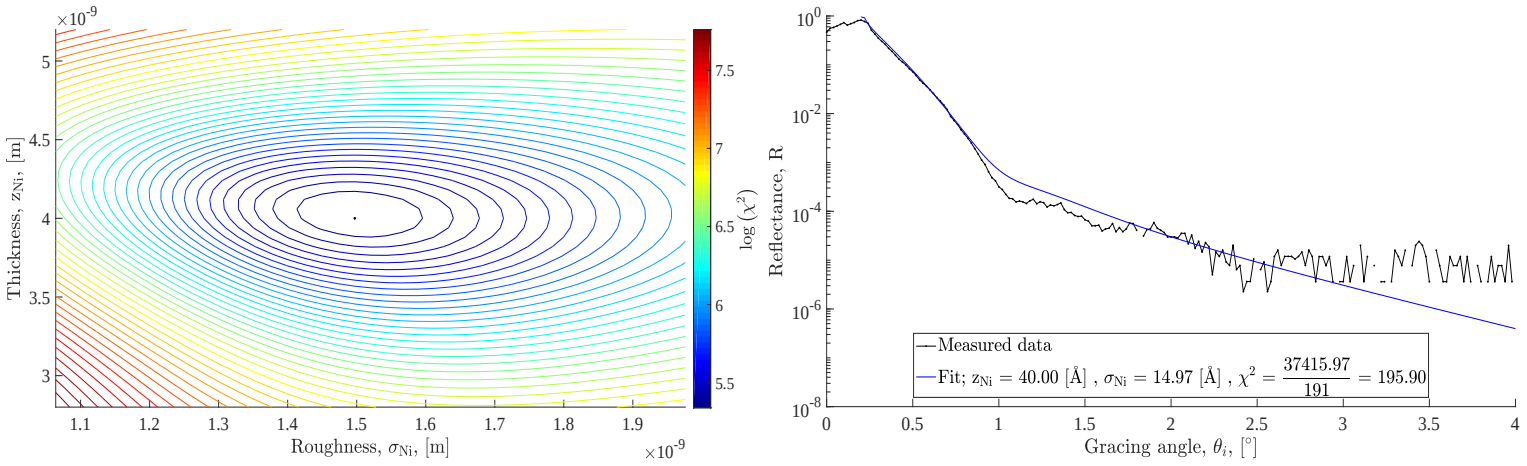
Figure 58: (a): si6470_2_contourZS. (b): si6470_2_plot.



(a) **(b)**

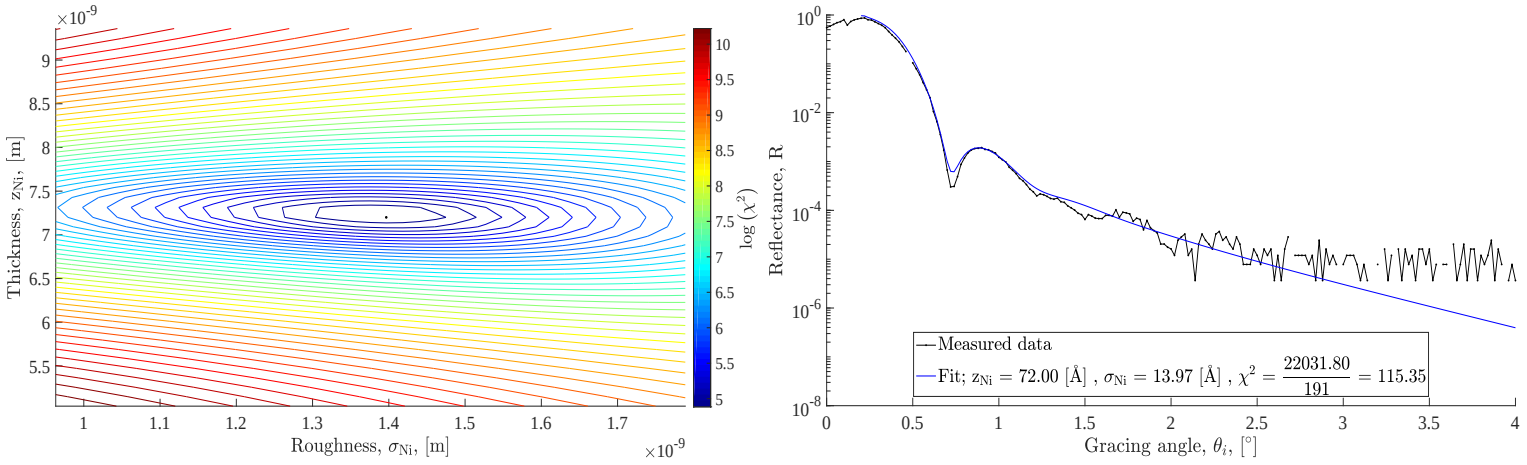
Figure 59: (a): si6479_2_UnCorPlot. (b): si6480_2_UnCorPlot.

A.9 Phase 3 - Coatings in $20\%\,N_2$ at 2.5 mTorr



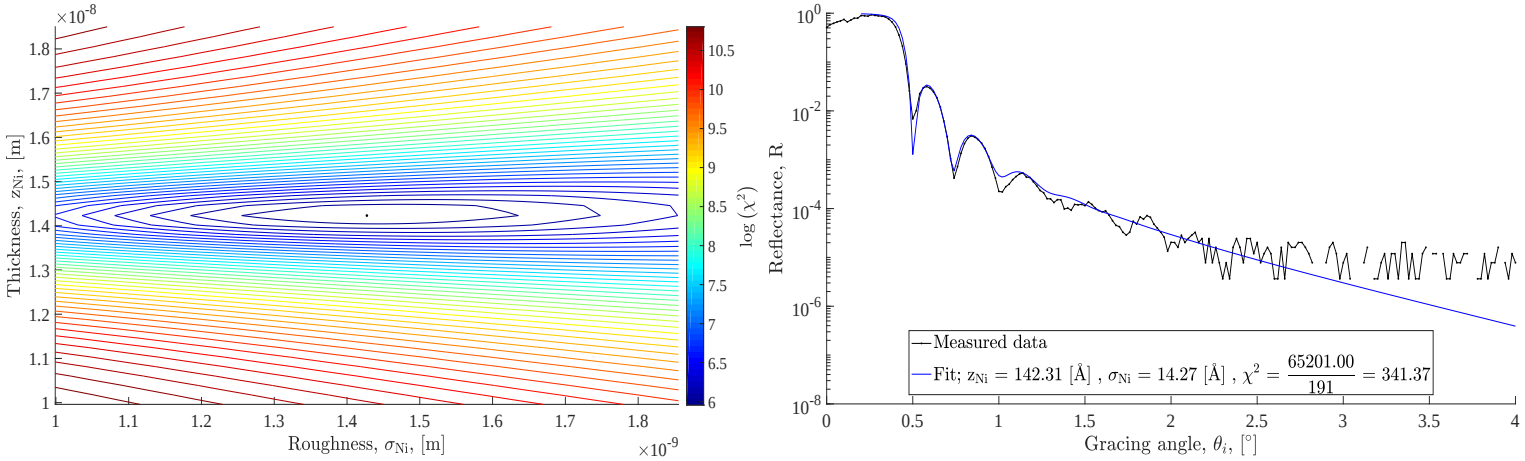
(a) **(b)**

Figure 60: (a): si6485_2_contourZS. (b): si6485_2_plot.



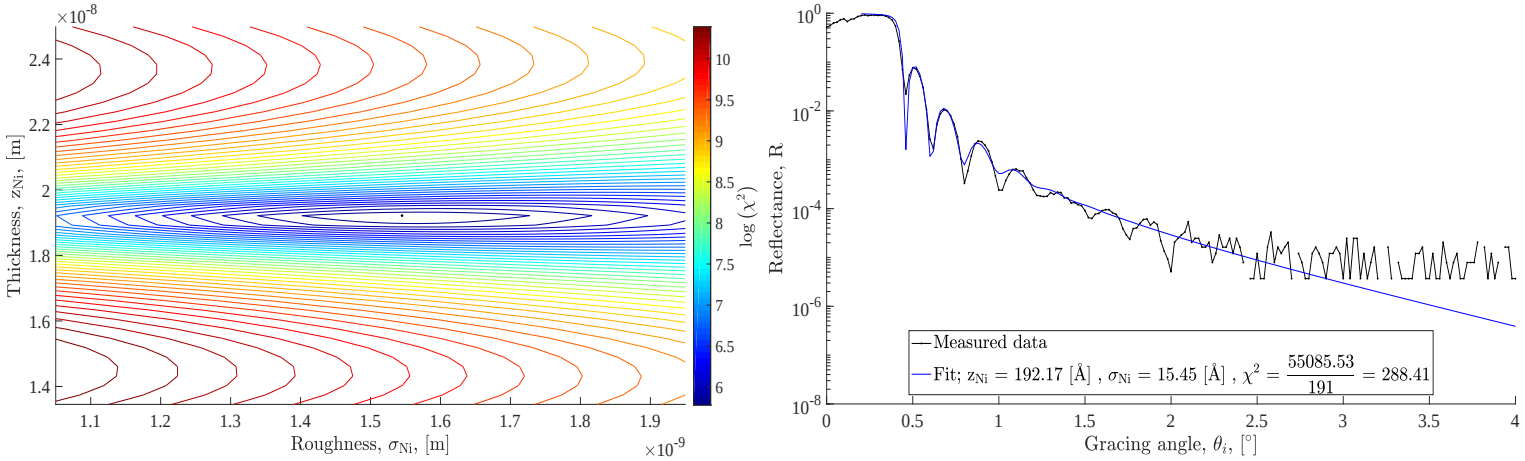
(a) **(b)**

Figure 61: (a): si6486_3_contourZS. (b): si6486_3_plot.



(a) **(b)**

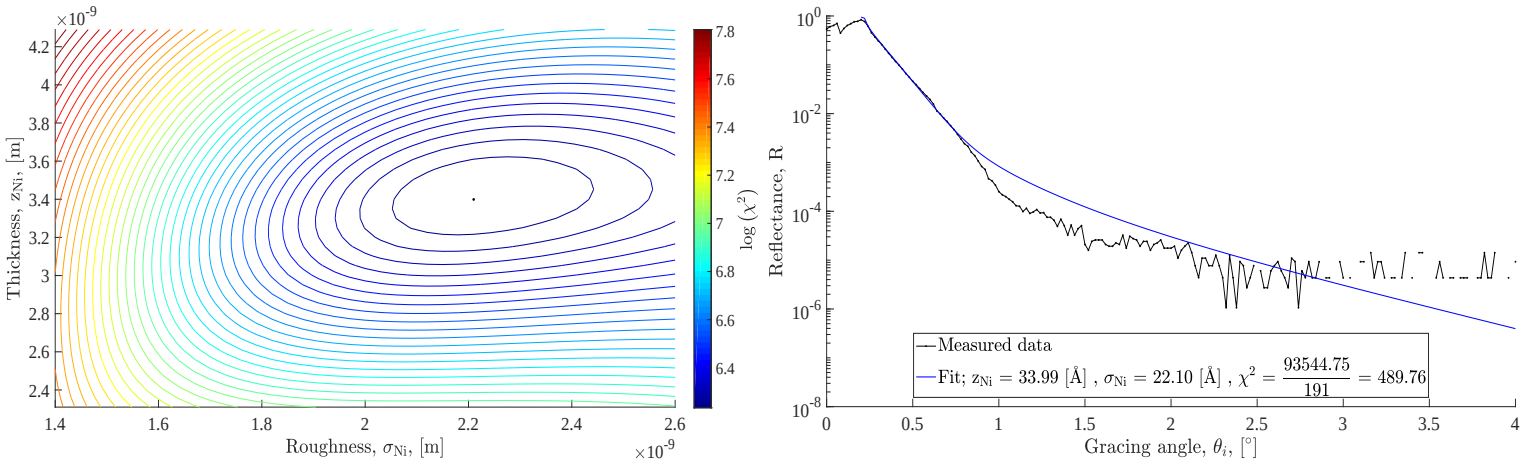
Figure 62: (a): si6487_2_contourZS. (b): si6487_2_plot.



(a) **(b)**

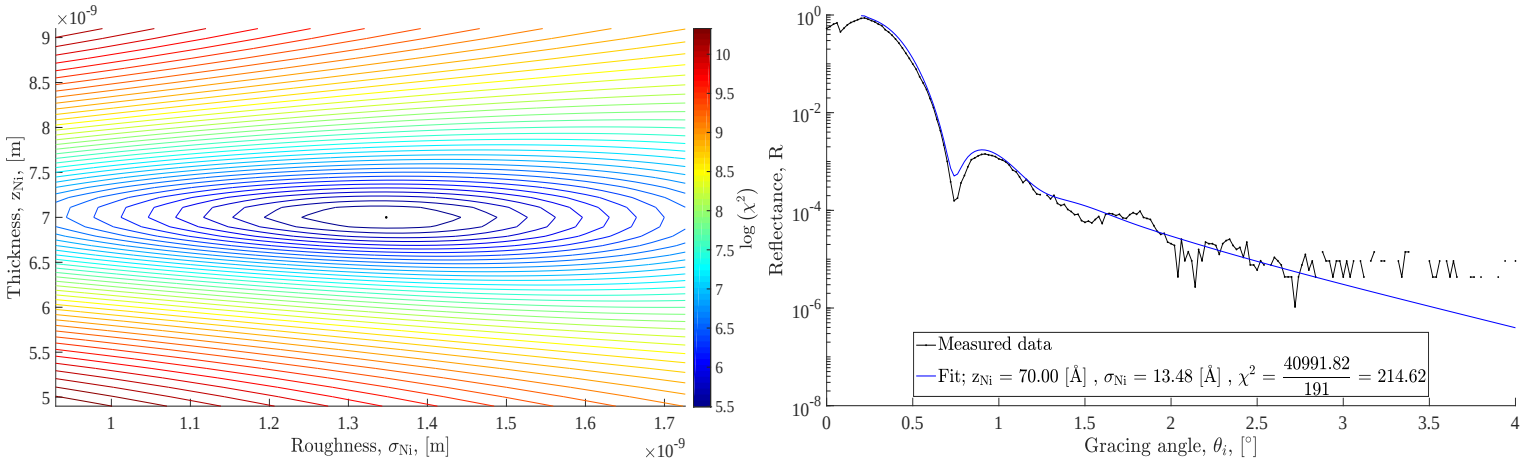
Figure 63: (a): si6488_2_contourZS. (b): si6488_2_plot.

A.10 Phase 3 - Coatings in $20\%\,N_2$ at 3.0 mTorr



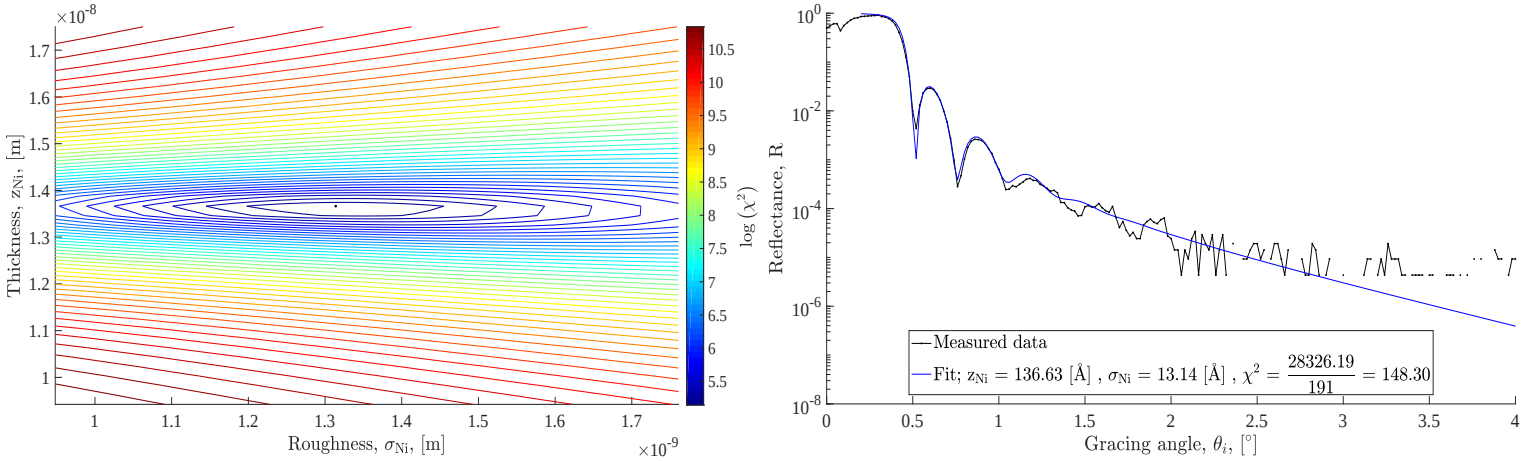
(a) **(b)**

Figure 64: (a): si6481_3_contourZS. (b): si6481_3_plot.



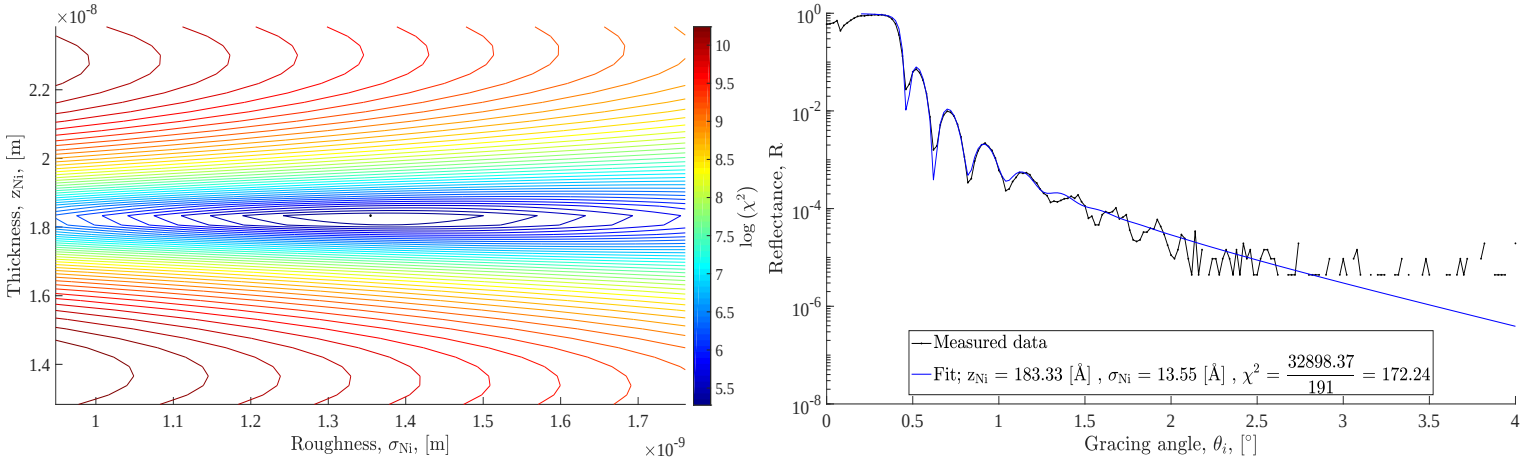
(a) **(b)**

Figure 65: (a): si6482_2_contourZS. (b): si6482_2_plot.



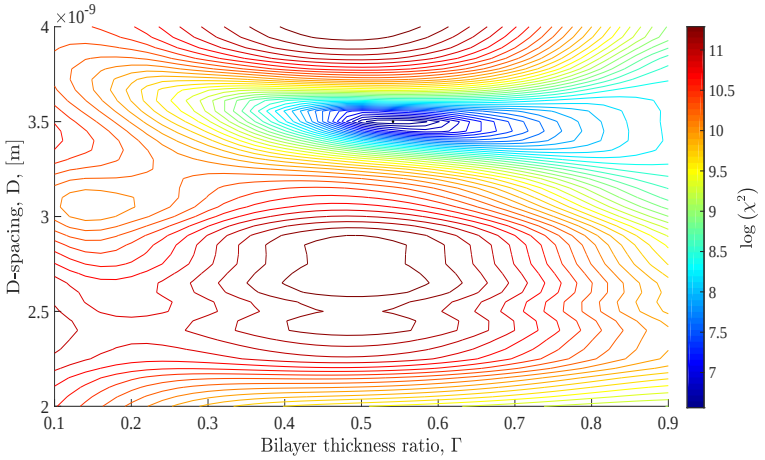
(a) **(b)**

Figure 66: (a): si6483_2_contourZS. (b): si6483_2_plot.

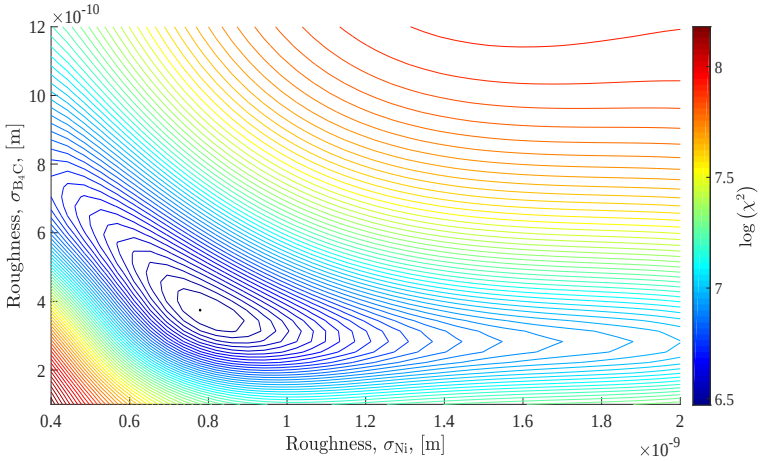


(a) **(b)**

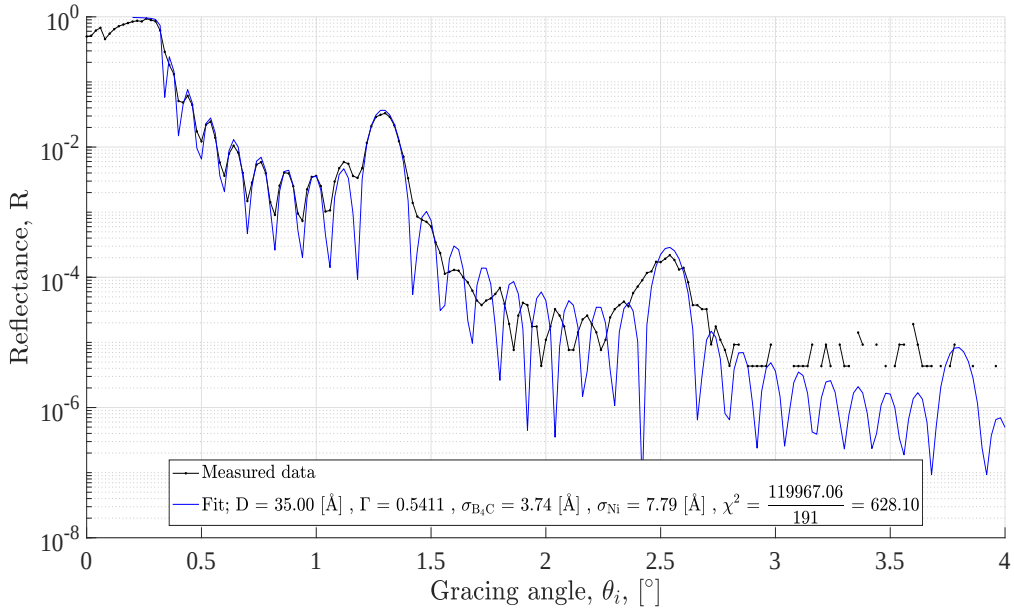
Figure 67: (a): si6484_2_contourZS. (b): si6484_2_plot.


×10^-9
D-spacing, D, [m]
Bilayer thickness ratio, Γ
log (χ²)

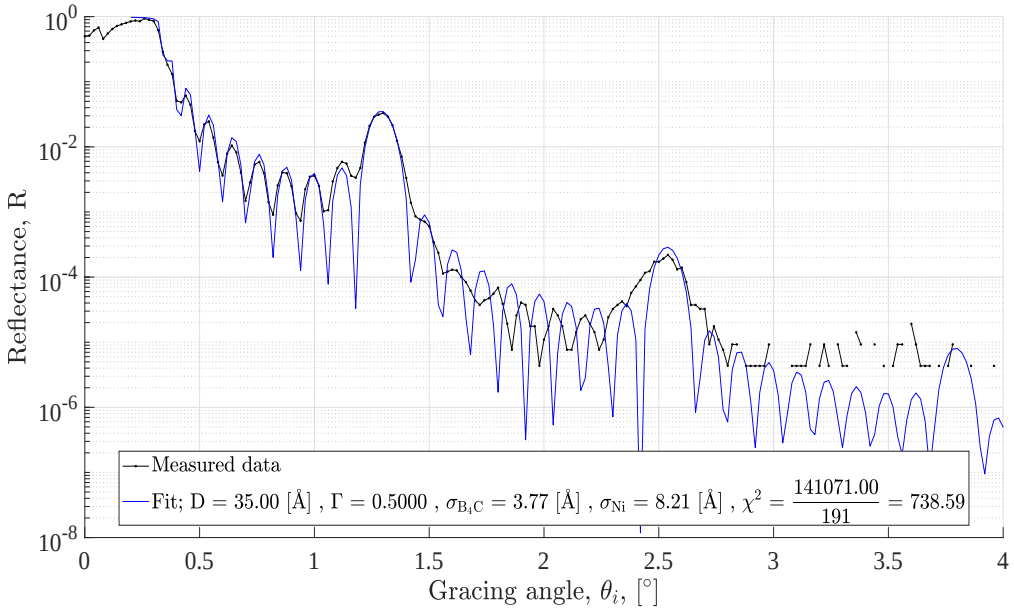
(a)


×10^-10
Roughness, σ_B4C, [m]
Roughness, σ_Ni, [m]
×10^-9
log (χ²)

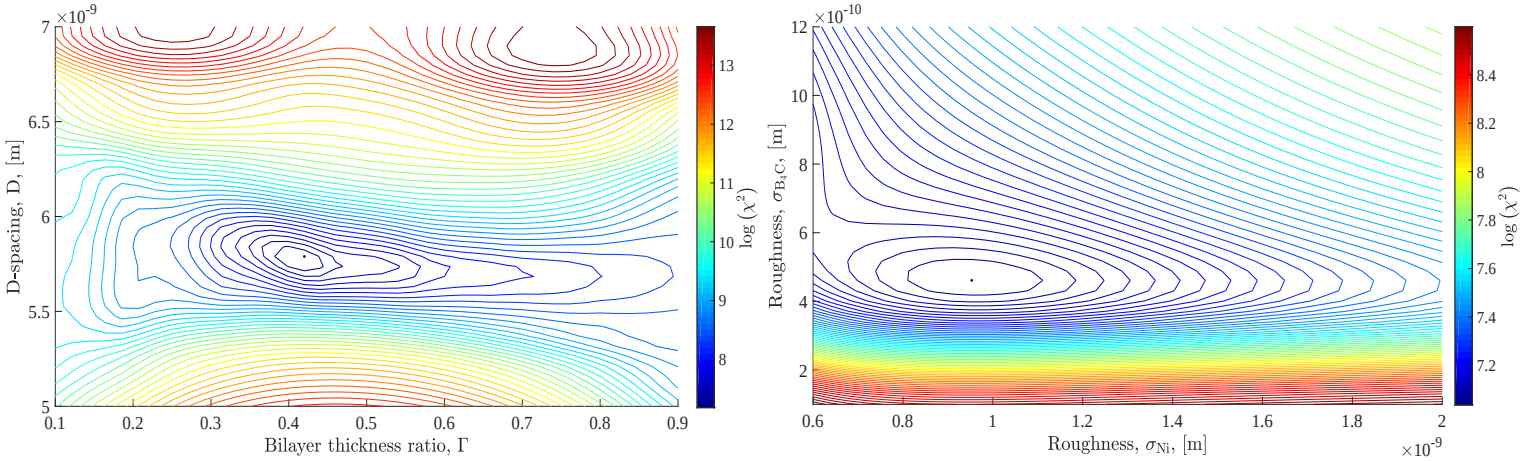
(b)


Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 35.00 [Å] , Γ = 0.5411 , σ_B4C = 3.74 [Å] , σ_Ni = 7.79 [Å] , χ² = 119967.06/191 = 628.10

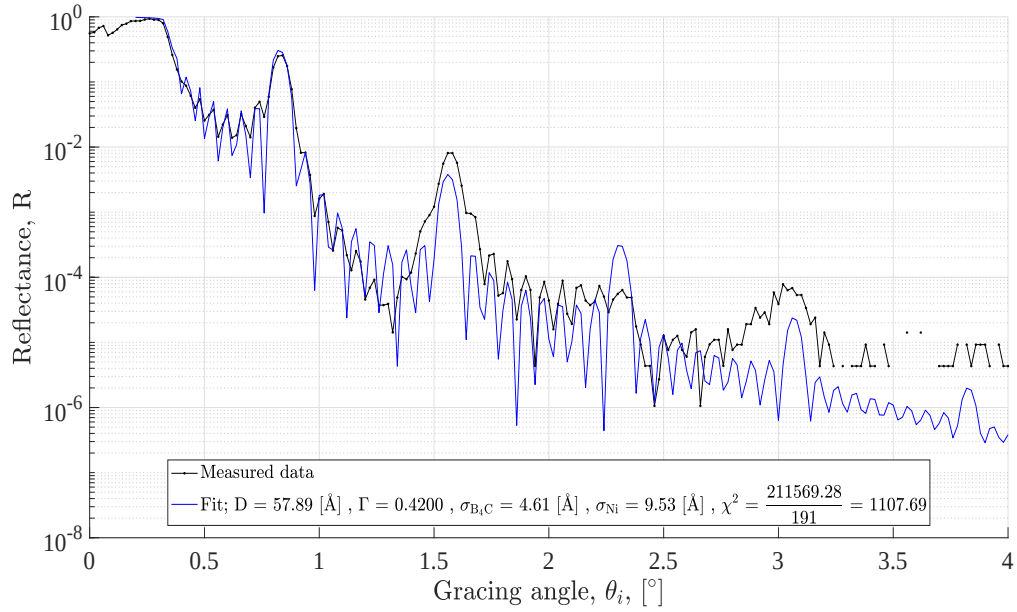
(c)


Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 35.00 [Å] , Γ = 0.5000 , σ_B4C = 3.77 [Å] , σ_Ni = 8.21 [Å] , χ² = 141071.00/191 = 738.59

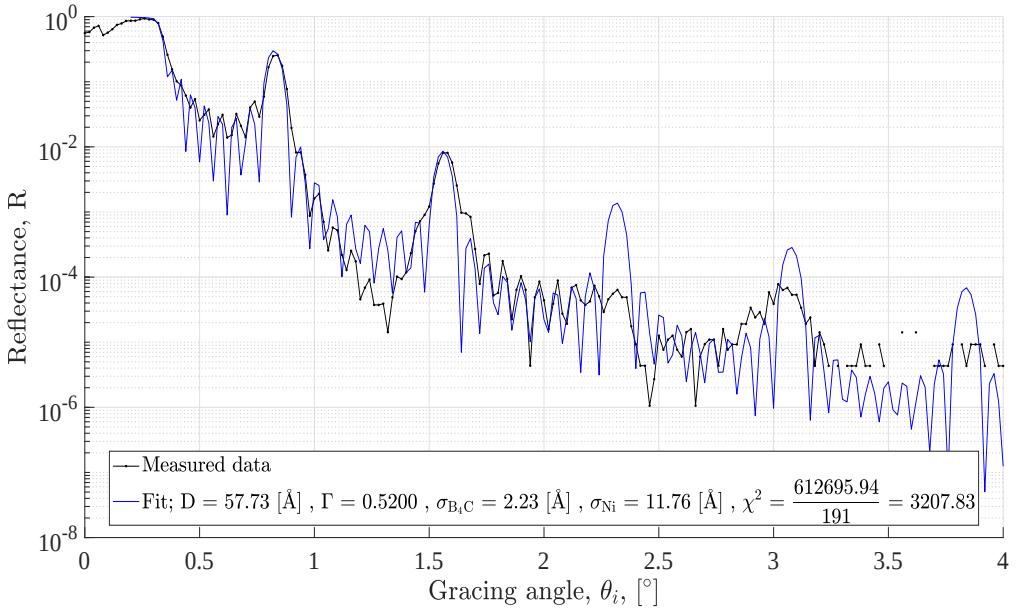
(d)


×10^-9
D-spacing, D, [m]
Bilayer thickness ratio, Γ
log (χ²)
×10^-10
Roughness, σ_B4C, [m]
Roughness, σ_Ni, [m]
×10^-9
log (χ²)

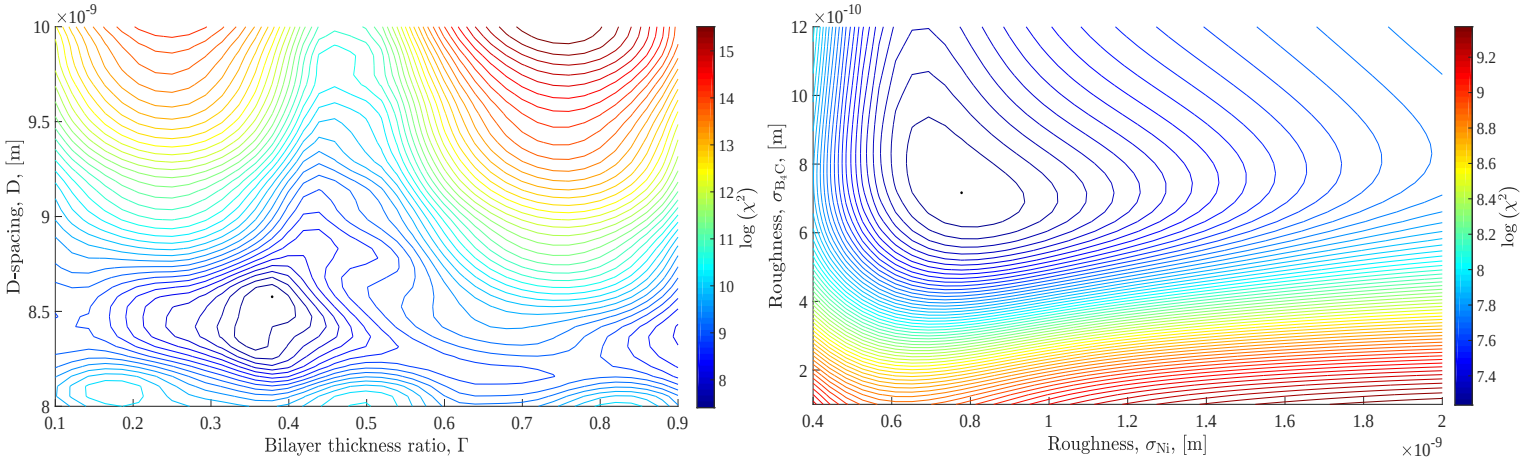
(a) **(b)**


Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 57.89 [Å] , Γ = 0.4200 , σ_B4C = 4.61 [Å] , σ_Ni = 9.53 [Å] , χ² = 211569.28/191 = 1107.69

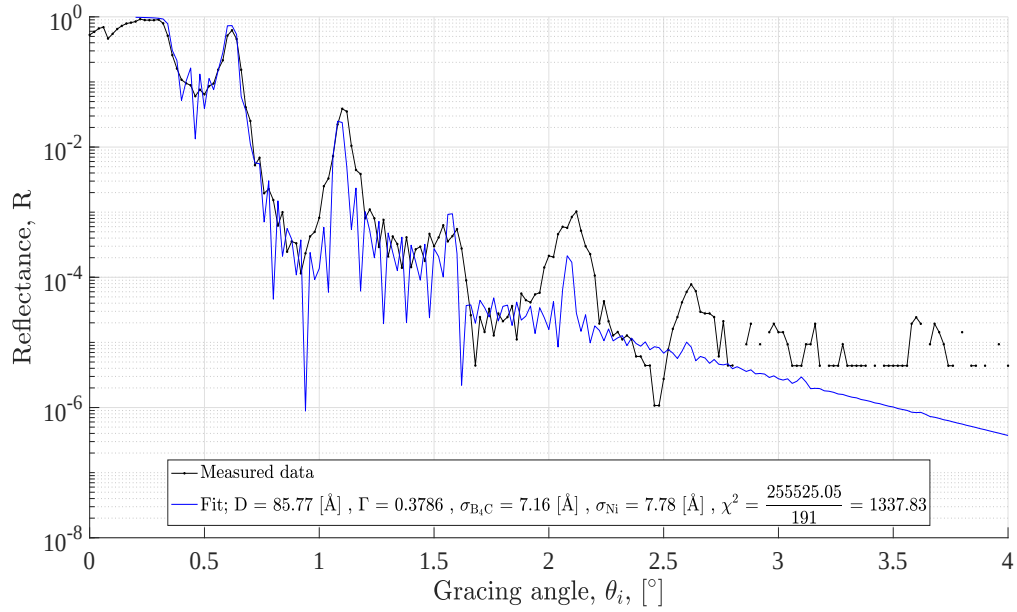
(c)


Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 57.73 [Å] , Γ = 0.5200 , σ_B4C = 2.23 [Å] , σ_Ni = 11.76 [Å] , χ² = 612695.94/191 = 3207.83

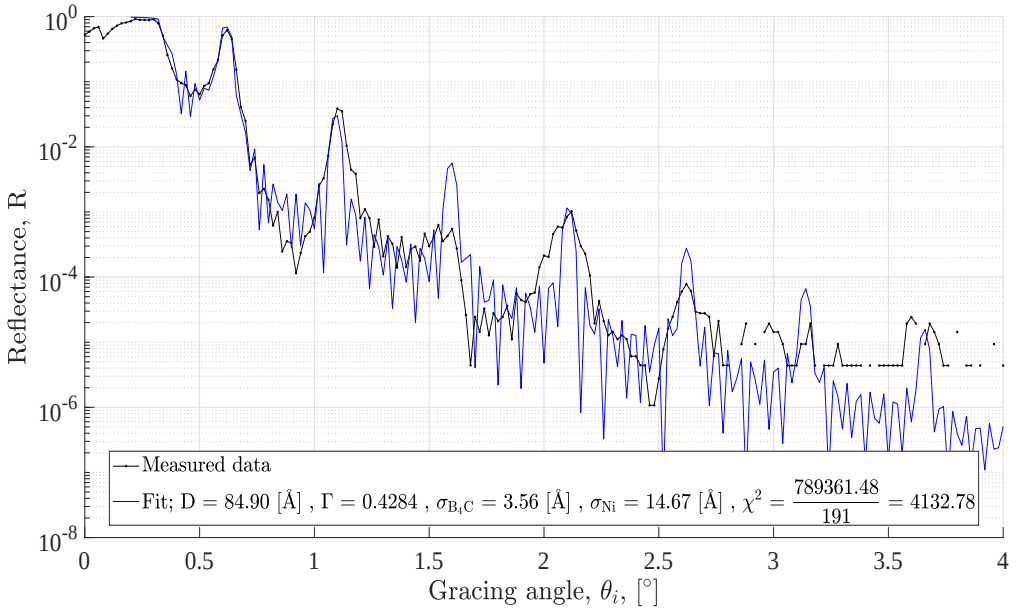
(d)


×10^-9
D-spacing, D, [m]
Bilayer thickness ratio, Γ
log (χ²)
×10^-10
Roughness, σ_B4C, [m]
Roughness, σ_Ni, [m]
×10^-9
log (χ²)

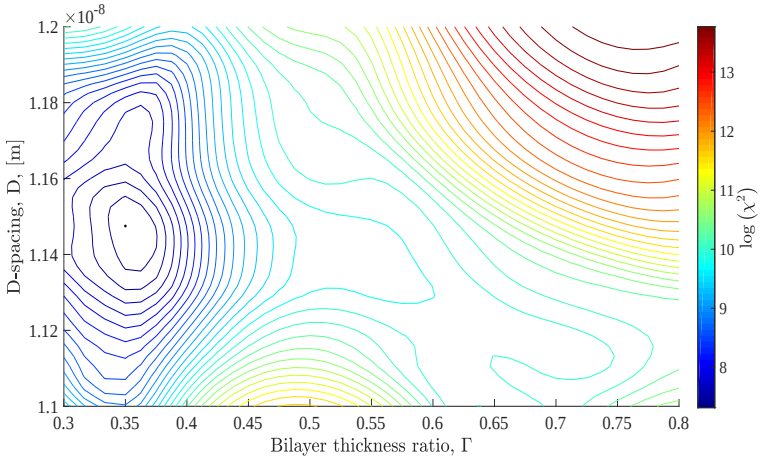
(a) **(b)**


Reflectance, R
Measured data
Fit; D = 85.77 [Å] , Γ = 0.3786 , σ_B4C = 7.16 [Å] , σ_Ni = 7.78 [Å] , χ² = 255525.05/191 = 1337.83
Gracing angle, θ_i, [°]

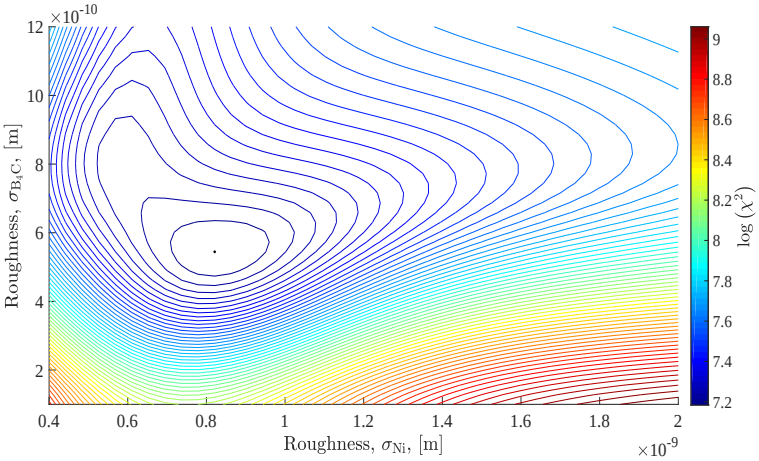
(c)


Reflectance, R
Measured data
Fit; D = 84.90 [Å] , Γ = 0.4284 , σ_B4C = 3.56 [Å] , σ_Ni = 14.67 [Å] , χ² = 789361.48/191 = 4132.78
Gracing angle, θ_i, [°]

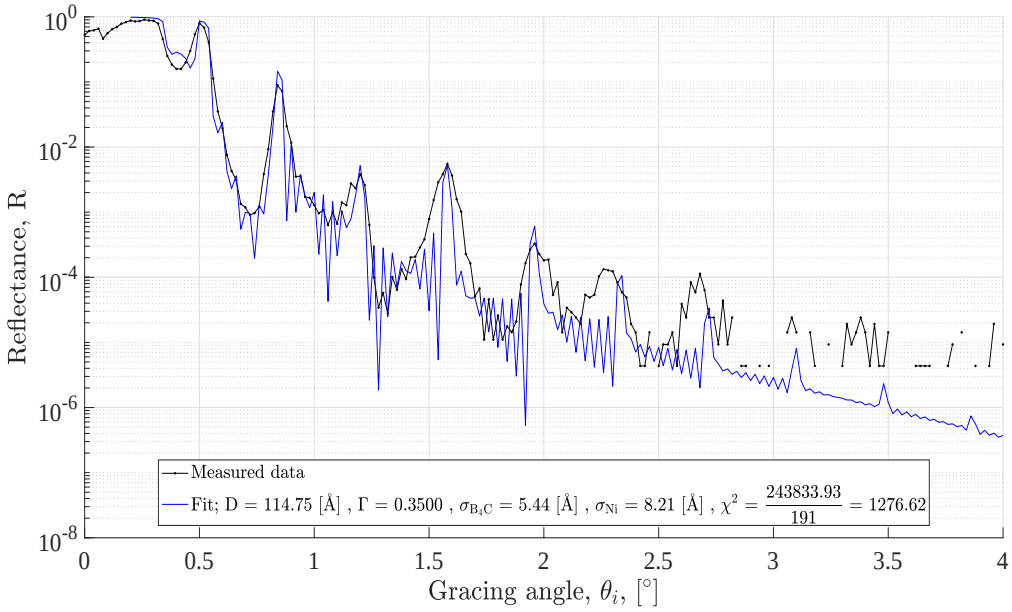
(d)


×10^-8
D-spacing, D, [m]
Bilayer thickness ratio, Γ
log (χ²)

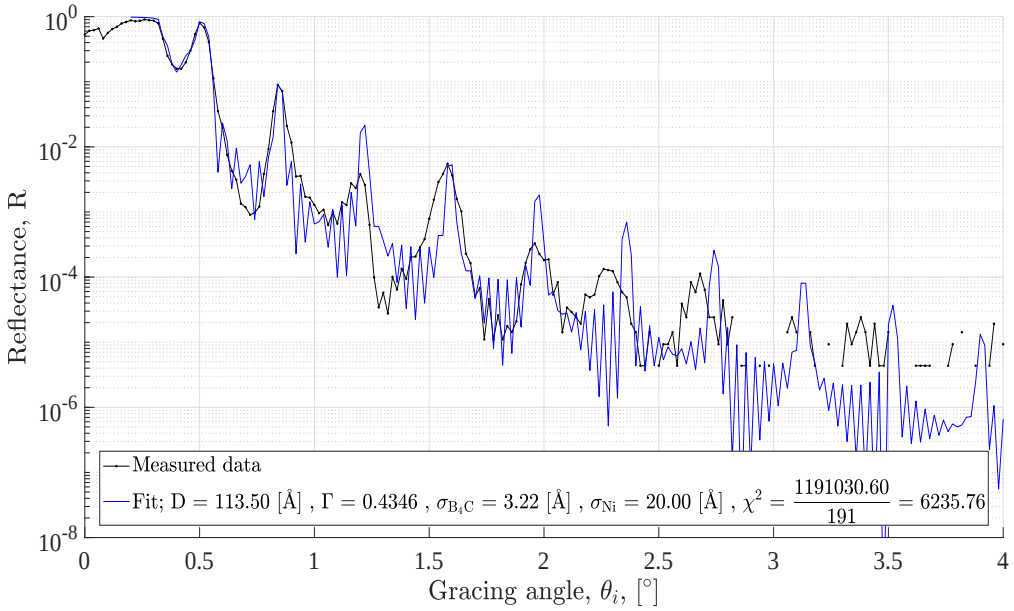
(a)


×10^-10
Roughness, σ_B4C, [m]
Roughness, σ_Ni, [m]
×10^-9
log (χ²)

(b)

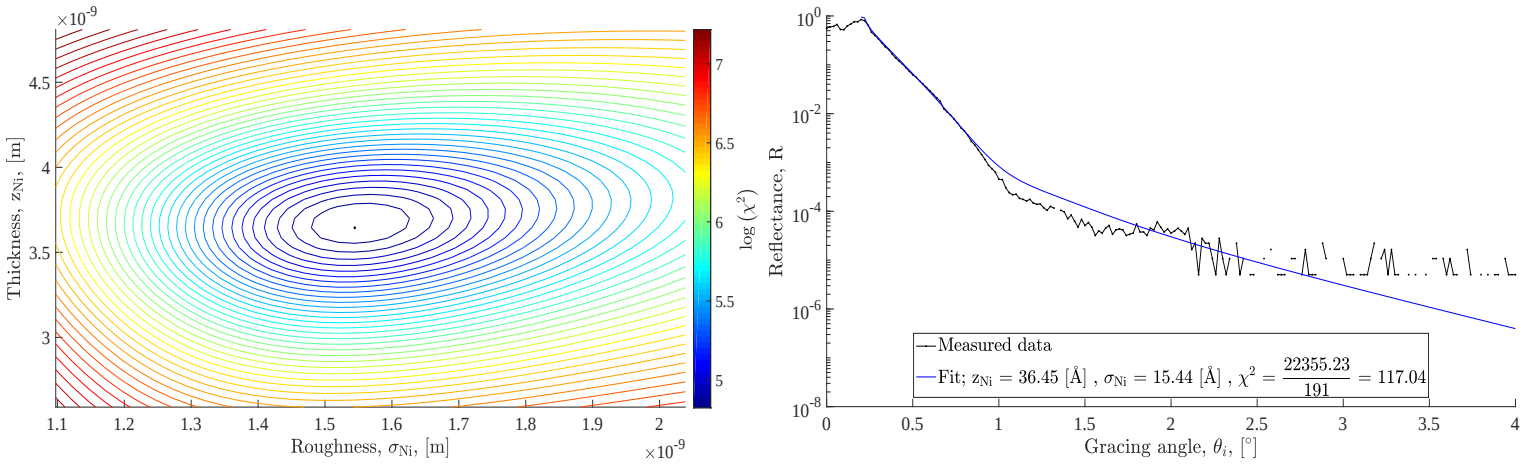

Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 114.75 [Å] , Γ = 0.3500 , σ_B4C = 5.44 [Å] , σ_Ni = 8.21 [Å] , χ² = 243833.93/191 = 1276.62

(c)


Reflectance, R
Gracing angle, θ_i, [°]
Measured data
Fit; D = 113.50 [Å] , Γ = 0.4346 , σ_B4C = 3.22 [Å] , σ_Ni = 20.00 [Å] , χ² = 1191030.60/191 = 6235.76

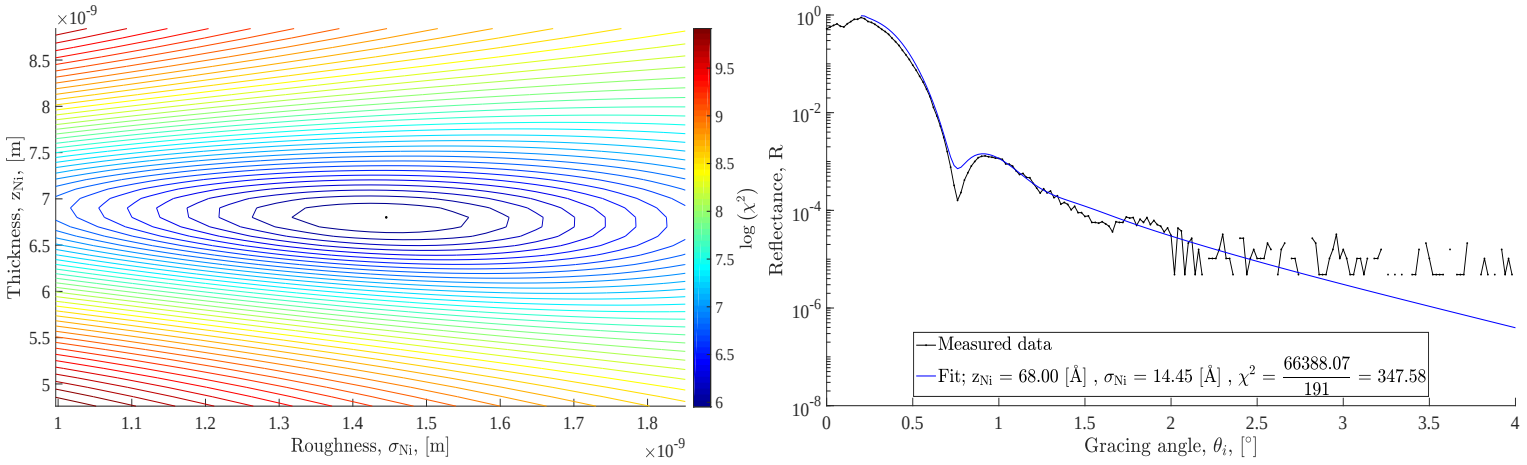
(d)

A.11 Phase 3 - Coatings in $20\%\,N_2$ at 4.0 mTorr



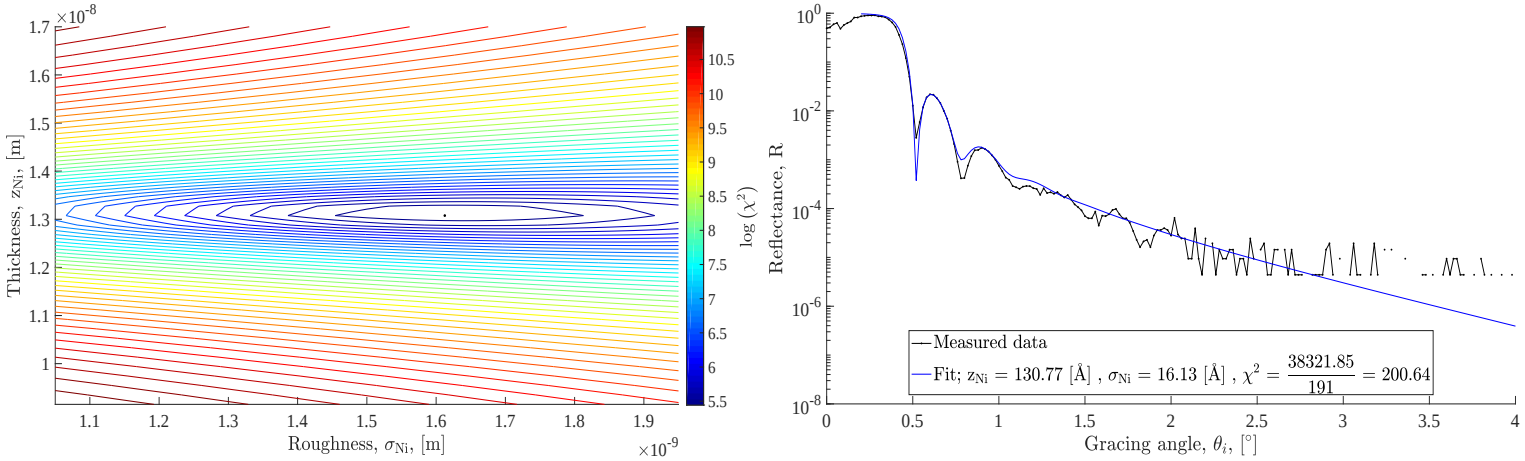
(a) **(b)**

Figure 72: (a): si6489_2_contourZS. (b): si6489_2_plot.



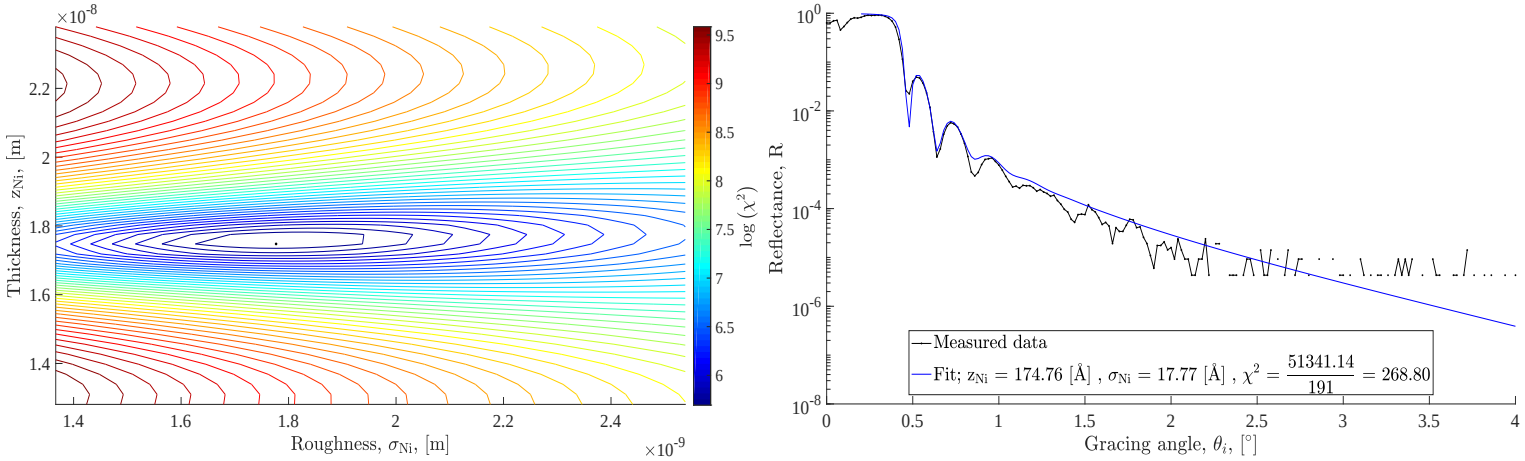
(a) **(b)**

Figure 73: (a): si6490_2_contourZS. (b): si6490_2_plot.



(a) **(b)**

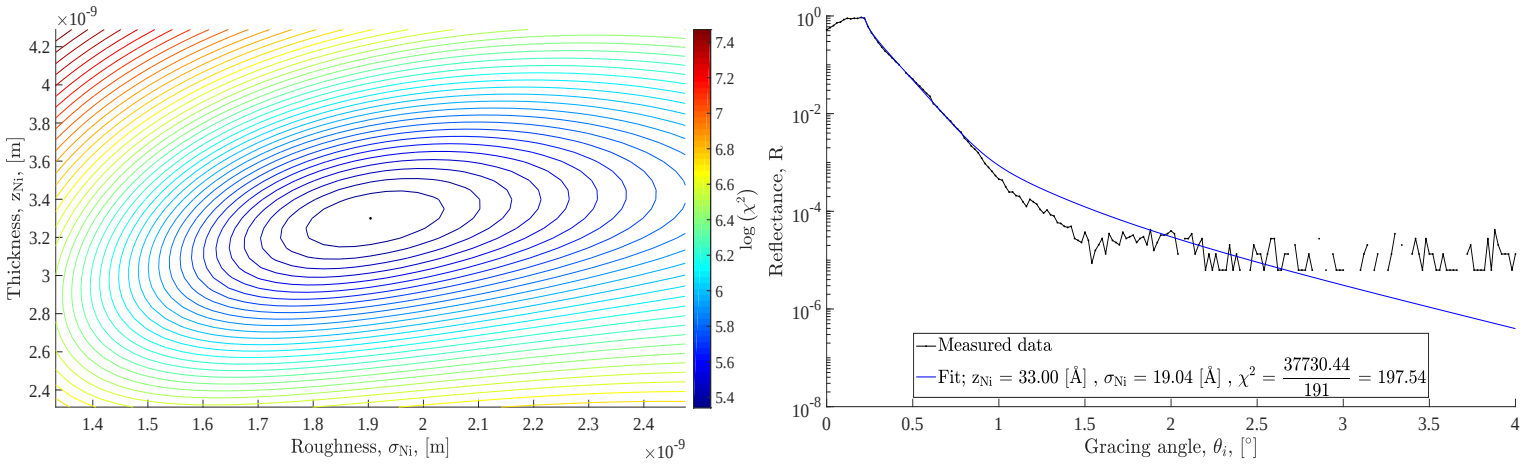
Figure 74: (a): si6491_2_contourZS. (b): si6491_2_plot.



(a) **(b)**

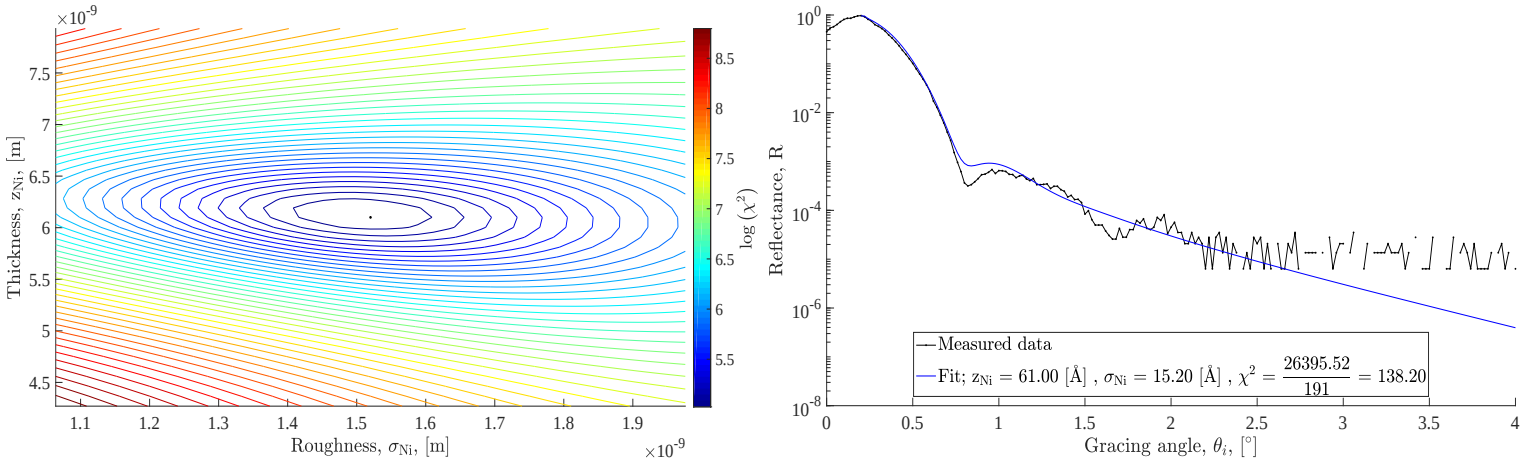
Figure 75: (a): si6492_2_contourZS. (b): si6492_2_plot.

A.12 Phase 3 - Coatings in $20\%\,N_2$ at 5.0 mTorr



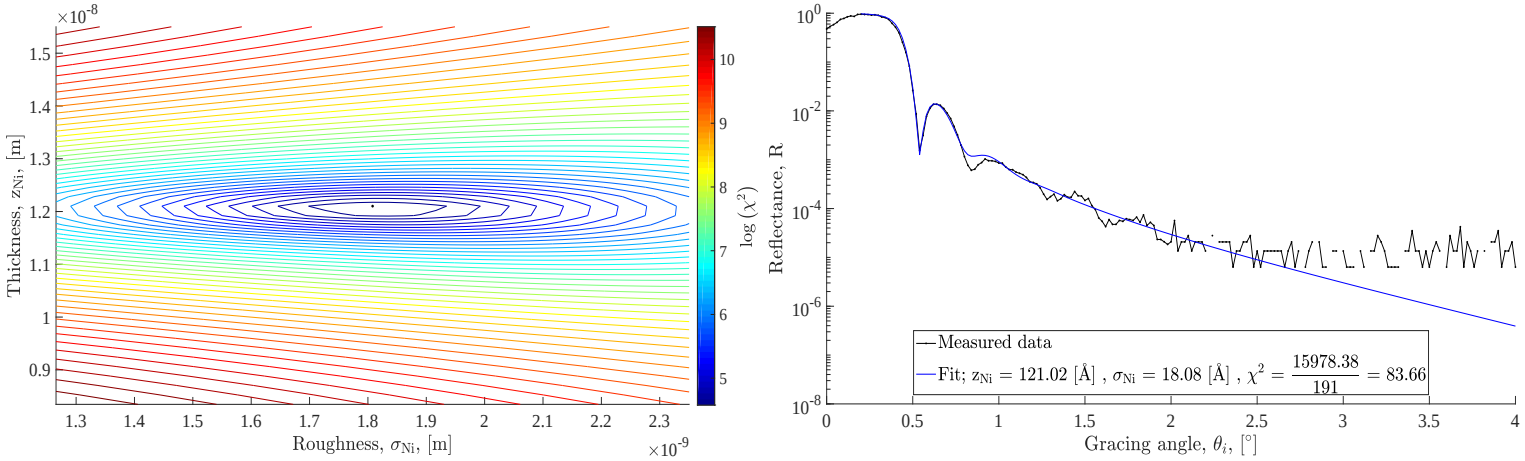
(a) **(b)**

Figure 76: (a): si6493_2_contourZS. (b): si6493Z_2_plot.



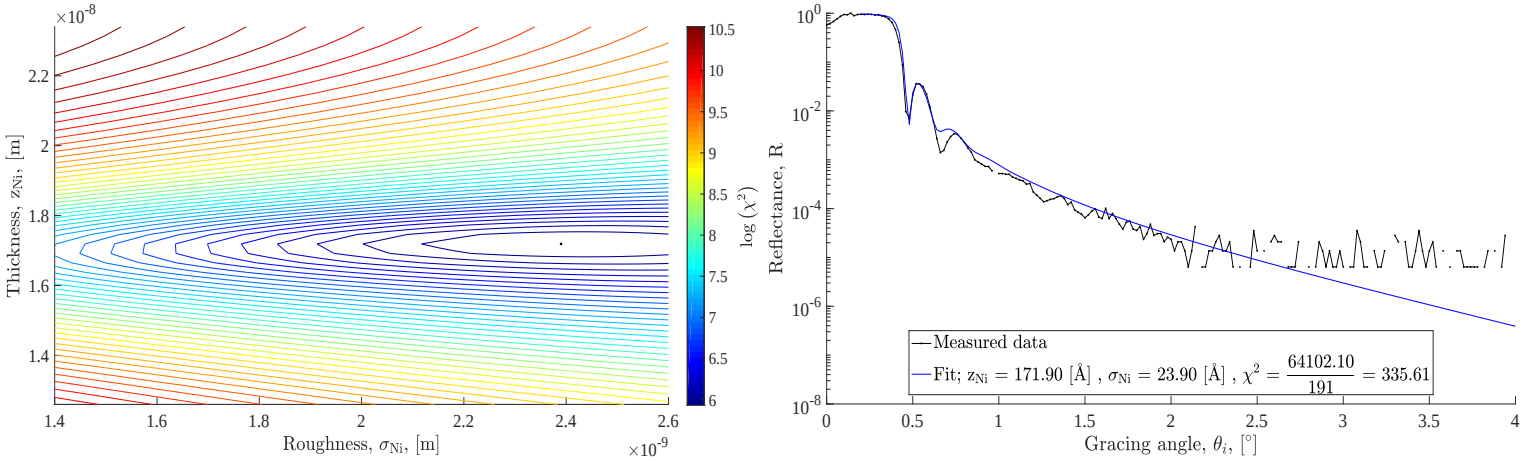
(a) **(b)**

Figure 77: (a): si6494_2_contourZS. (b): si6494_2_plot.



(a) **(b)**

Figure 78: (a): si6495_2_contourZS. (b): si6495_2_plot.



(a) **(b)**

Figure 79: (a): si6496_2_contourZS. (b): si6496_2_plot.

B Source codes

B.1 MainScript

```
%% Initializing:
clear all
close all
clc
tic

%% Defining single layer:
Layer1Material = 'Ni';
Layer1Thickness = [];
Layer1Roughness = [];
SubstrateMaterial = 'Si';
SubstrateRoughness = 2.5*10^(-10);
DegreeLimit = [0.2 4 nan nan]';
vSL = {SubstrateMaterial,SubstrateRoughness,Layer1Material,Layer1Thickness,...
       Layer1Roughness,DegreeLimit};

%% Defining multilayer:
Layer1Material = 'B4C';
Layer1Roughness = [];
Layer2Material = 'Ni';
Layer2Roughness = [];
SubstrateMaterial = 'Si';
SubstrateRoughness = 2.5*10^(-10);
Dspacing = [];
Gamma = [];
NumberOfBilayers = 10;
DegreeLimit = [0.2 4 nan nan]';
vML = {SubstrateMaterial,SubstrateRoughness,Layer1Material,Layer1Roughness,...
       Layer2Material,Layer2Roughness,Dspacing,Gamma,NumberOfBilayers,DegreeLimit};

%% Defining file:
SampleName = 'si6442';
Extension = 2;
Path = strcat('D:\Dropbox\Master\XRRData\',SampleName,'\');
FileName = strcat(SampleName,'_',num2str(Extension));

% Printing witness sample specifications:
fid = fopen('FileToBeFitted.txt','wt');
fprintf(fid,'%s\n%s',FileName,Path);
fclose(fid);

%% SL:
% lb = [20*10^(-10) 1*10^(-10) ]';
% m  = [25*10^(-10) 8*10^(-10)]';
% ub = [60*10^(-10) 20*10^(-10)]';
% BraggNo = [nan nan ; nan nan ; nan nan ]';
% [Out] = UncorrelatedLooping(m,lb,ub,vSL,BraggNo);

ProgramTime = toc
```

B.2 IMD

```
function [R] = IMD(ThetaI,NumberOfLayers,n,d,sigma,lambda)

NumberOfInterfaces = NumberOfLayers+1;

for i = 1:NumberOfInterfaces
    j = i+1;

    ThetaT(:,i) = asind(n(i).*sind(ThetaI(:,i))./n(j));
    ThetaI(:,j) = ThetaT(:,i);

    s(:,i) = 4*pi.*sqrt(cosd(ThetaI(:,i)).*cosd(ThetaT(:,i)))./lambda;

    beta(:,i) = 2*pi.*d(i,1).*n(i,1).*cosd(ThetaI(:,i))./lambda;

    w(:,i) = exp(-s(:,i).^(2).*sigma(i,1).^(2)./2);

    % For s polarization:
    rijs(:,i) = (n(i).*cosd(ThetaI(:,i))-n(j).*cosd(ThetaT(:,i)))./(n(i)...
                .*cosd(ThetaI(:,i))+n(j).*cosd(ThetaT(:,i)));
    tijs(:,i) = 2.*n(i).*cosd(ThetaI(:,i))./(n(i).*cosd(ThetaI(:,i))+n(j)...
                .*cosd(ThetaT(:,i)));

    % For p polarization:
    rijp(:,i) = (n(i).*cosd(ThetaT(:,i))-n(j).*cosd(ThetaI(:,i)))./(n(i)...
                .*cosd(ThetaT(:,i))+n(j).*cosd(ThetaI(:,i)));
    tijp(:,i) = 2.*n(i).*cosd(ThetaI(:,i))./(n(i).*cosd(ThetaT(:,i))+n(j)...
                .*cosd(ThetaI(:,i)));

    Modifiedrijs(:,i) = rijs(:,i).*w(:,i);
    Modifiedrijp(:,i) = rijp(:,i).*w(:,i);
end

Netris(:,NumberOfInterfaces+1) = zeros(size(ThetaI,1),1);
Netrip(:,NumberOfInterfaces+1) = zeros(size(ThetaI,1),1);


for x = 1:NumberOfInterfaces
    y = NumberOfInterfaces+1-x;
    z = y+1;
beta(:,y) = 2*pi.*d(y).*n(z).*cosd(ThetaI(:,z))./lambda;

Netris(:,y) = (Modifiedrijs(:,y)+Netris(:,z).*exp(2.*1i.*beta(:,y)))...
               ./(1+Modifiedrijs(:,y).*Netris(:,z).*exp(2.*1i.*beta(:,y)));
Netrip(:,y) = (Modifiedrijp(:,y)+Netrip(:,z).*exp(2.*1i.*beta(:,y)))...
               ./(1+Modifiedrijp(:,y).*Netrip(:,z).*exp(2.*1i.*beta(:,y)));
end

Rs = (abs(Netris(:,:))).^2;
Rp = (abs(Netrip(:,:))).^2;

R = (Rs+Rp)/2;
R = R(:,1);
end
```

B.3 FilmDefinitions

```
function [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(varargin)
%% FilmDefinitions is a function that designs a thin film with the parameters need to as
% input to for IMD.

% INPUT:
% varargin: Same as vXL. Cell of size(1,6) for SL and size(1,10) for ML. Contains the
%           film specifications, such as materials, number of layers and gracing angle
%           fitting interval. For specific instructions see Fittingscript.
%
% OUTPUT:
% ThetaI:         Vector of incident angles.
% NumberOfLayers: Selfexplanatory!
% n:              Complex refractive indexes of the materials in the film.
% d:              Thicknesses of each layer in the film.
% sigma:          Roughnesses of each layer in the film.
% lambda:         Wavelength of the light.
%
% Author:   Christoffer Raad (christoffer.raad@gmail.com)
% Date:     15/06/2017

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Loading refractive index data:
NIDATA = importdata('NIDATA.mat');
SIDATA = importdata('SIDATA.mat');
B4CDATA = importdata('B4CDATA.mat');
IRDATA = importdata('IRDATA.mat');
NIDATA(:,1)=NIDATA(:,1)*10^(-10);
SIDATA(:,1)=SIDATA(:,1)*10^(-10);
B4CDATA(:,1)=B4CDATA(:,1)*10^(-10);
IRDATA(:,1)=IRDATA(:,1)*10^(-10);

% Defining wavelength:
eV = 1.60217662*10^(-19); % [J]
E = 8047*eV; % [keV]
h = 6.62607004*10^(-34); % [m^2*kg/s]
c = 299792458; % [m/s]
lambda = h*c/E; %[m]

% Interpolate to find n and k at wavelength:
[~,idx] = min(abs(NIDATA(:,1)-lambda));
g(:,1) = NIDATA(idx,1):(NIDATA(idx+1,1)-NIDATA(idx,1))/1000:NIDATA(idx+1,1);
interpNIn = interp1(NIDATA(idx:idx+1,1),NIDATA(idx:idx+1,2),g);
interpNIk = interp1(NIDATA(idx:idx+1,1),NIDATA(idx:idx+1,3),g);
[~,idx] = min(abs(g-lambda));
n_Ni = interpNIn(idx);
k_Ni = interpNIk(idx);

[~,idx] = min(abs(SIDATA(:,1)-lambda));
g(:,1) = SIDATA(idx,1):(SIDATA(idx+1,1)-SIDATA(idx,1))/1000:SIDATA(idx+1,1);
interpSIn = interp1(SIDATA(idx:idx+1,1),SIDATA(idx:idx+1,2),g);
interpSIk = interp1(SIDATA(idx:idx+1,1),SIDATA(idx:idx+1,3),g);
[~,idx] = min(abs(g-lambda));
n_Si = interpSIn(idx);
k_Si = interpSIk(idx);
```

```

[~,idx] = min(abs(B4CDATA(:,1)-lambda));
g(:,1) = B4CDATA(idx,1):(B4CDATA(idx+1,1)-B4CDATA(idx,1))/1000:B4CDATA(idx+1,1);
interpB4Cn = interp1(B4CDATA(idx:idx+1,1),B4CDATA(idx:idx+1,2),g);
interpB4Ck = interp1(B4CDATA(idx:idx+1,1),B4CDATA(idx:idx+1,3),g);
[~,idx] = min(abs(g-lambda));
n_B4C = interpB4Cn(idx);
k_B4C = interpB4Ck(idx);

[~,idx] = min(abs(IRDATA(:,1)-lambda));
g(:,1) = IRDATA(idx,1):(IRDATA(idx+1,1)-IRDATA(idx,1))/1000:IRDATA(idx+1,1);
interpIRn = interp1(IRDATA(idx:idx+1,1),IRDATA(idx:idx+1,2),g);
interpIRk = interp1(IRDATA(idx:idx+1,1),IRDATA(idx:idx+1,3),g);
[~,idx] = min(abs(g-lambda));
n_Ir = interpIRn(idx);
k_Ir = interpIRk(idx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SL case:
if length(varargin{1}) == 6

    if length(varargin{1}{6}) ~= 4
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) and ',...
            'DegreeLimit(2) are mandatory and have to be set to some degree ',...
            'values other than nan. DegreeLimit(3) and DegreeLimit(4) have the ',...
            'option to be set to nans. DegreeLimit(3) and DegreeLimit(4) should ',...
            'be defined to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])

    elseif ~isnan(varargin{1}{6}(1)) && ~isnan(varargin{1}{6}(2)) && ...
            ~isnan(varargin{1}{6}(3)) && ~isnan(varargin{1}{6}(4))
        ThetaI1 = varargin{1}{6}(1):0.02:varargin{1}{6}(2);
        ThetaI2 = varargin{1}{6}(3):0.02:varargin{1}{6}(4);
        ThetaI1 = 90-ThetaI1';
        ThetaI2 = 90-ThetaI2';
        ThetaI = vertcat(ThetaI1,ThetaI2);

    elseif isnan(varargin{1}{6}(3)) && isnan(varargin{1}{6}(4))
        ThetaI = varargin{1}{6}(1):0.02:varargin{1}{6}(2);
        ThetaI = 90-ThetaI';
    else
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) and ',...
            'DegreeLimit(2) are mandatory and have to be set to some degree ',...
            'values other than nan. DegreeLimit(3) and DegreeLimit(4) have the ',...
            'option to be set to nans. DegreeLimit(3) and DegreeLimit(4) should ',...
            'be defined to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])
    end

    NumberOfLayers = 1;

    for i = 1:length(varargin{1}{4})
        d(1,i) = varargin{1}{4}(i);
        d(2,i) = 10^(-3);
    end

    for i = 1:length(varargin{1}{5})
        sigma(1,i) = varargin{1}{5}(i);
        sigma(2,i) = varargin{1}{2};
```

```
    end

    tmp1 = eval(cell2mat(strcat('n_',varargin{1}(3))));
    tmp2 = eval(cell2mat(strcat('k_',varargin{1}(3))));
    tmp3 = eval(cell2mat(strcat('n_',varargin{1}(1))));
    tmp4 = eval(cell2mat(strcat('k_',varargin{1}(1))));

    n = nan(3,1);
    n(1,1) = 1;
    n(2,1) = tmp1+1i*tmp2;
    n(3,1) = tmp3+1i*tmp4;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ML casw:
elseif length(varargin{1}) == 10

    if length(varargin{1}{10}) ~= 4
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) and ',...
            'DegreeLimit(2) are mandatory and have to be set to some degree ',...
            'values other than nan. DegreeLimit(3) and DegreeLimit(4) have the ',...
            'option to be set to nans. DegreeLimit(3) and DegreeLimit(4) should ',...
            'be defined to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])

    elseif ~isnan(varargin{1}{10}(1)) && ~isnan(varargin{1}{10}(2)) && ...
            ~isnan(varargin{1}{10}(3)) && ~isnan(varargin{1}{10}(4))
        ThetaI1 = varargin{1}{10}(1):0.02:varargin{1}{10}(2);
        ThetaI2 = varargin{1}{10}(3):0.02:varargin{1}{10}(4);
        ThetaI1 = 90-ThetaI1';
        ThetaI2 = 90-ThetaI2';
        ThetaI = vertcat(ThetaI1,ThetaI2);

    elseif isnan(varargin{1}{10}(3)) && isnan(varargin{1}{10}(4))
        ThetaI = varargin{1}{10}(1):0.02:varargin{1}{10}(2);
        ThetaI = 90-ThetaI';
    else
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) and ',...
            'DegreeLimit(2) are mandatory and have to be set to some degree ',...
            'values other than nan. DegreeLimit(3) and DegreeLimit(4) have the ',...
            'option to be set to nans. DegreeLimit(3) and DegreeLimit(4) should ',...
            'be defined to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])
    end


    NumberOfLayers = 2*varargin{1}{9};

    for i = 1:length(varargin{1}{7})
        for j = 1:length(varargin{1}{8})
            d(1,i,j) = varargin{1}{7}(i)*varargin{1}{8}(j);
            d(2,i,j) = varargin{1}{7}(i)*(1-varargin{1}{8}(j));
        end
    end
    d = repmat(d,varargin{1}{9},1);
    d(end+1,:,:) = 10^(-3);

    for i = 1:length(varargin{1}{4})
        for j = 1:length(varargin{1}{6})
            sigma(1,i,j) = varargin{1}{4}(i);
```

```
            sigma(2,i,j) = varargin{1}{6}(j);
        end
    end
    sigma = repmat(sigma,varargin{1}{9},1);
    sigma(end+1,:,:) = varargin{1}{2};

    tmp1 = eval(strcat('n_',varargin{1}{1}));
    tmp2 = eval(strcat('k_',varargin{1}{1}));
    tmp3 = eval(strcat('n_',varargin{1}{3}));
    tmp4 = eval(strcat('k_',varargin{1}{3}));
    tmp5 = eval(strcat('n_',varargin{1}{5}));
    tmp6 = eval(strcat('k_',varargin{1}{5}));

    tmpn(1,1) = tmp3+1i*tmp4;
    tmpn(2,1) = tmp5+1i*tmp6;
    tmpn = repmat(tmpn,cell2mat(varargin{1}(9)),1);
    n = vertcat(1,tmpn,tmp1+1i*tmp2);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
    error(['Number of input variables must be 6 for a single layer or 10 for ',...
        'multilayers'])
end
end
```

B.4 LoadXRRData

```
function [Degree, R, E, D] = LoadXRRData(FileName, Path)
%% LoadXRRData is a function that extracts all the data from a 'FOURC' file
% produced during X-ray reflection (XRR) measurements.
% The function does background substraction and normalizes and and returns
% the normalised reflection and corresponding angles.
% LoadXRRData do also construct and saves a plot of the reflectance as
% function of incidence angle.
%
% INPUT:
% Filename:    Name of the FOURC-file generated during XRR measurement.
% Path:        Full path to where the FileName is located. For windows and
%              example could be D:\Dropbox\Master\XRRData\.
% PathOut:     Full path to where eps plot are to be saved.
% FileNameSIM: Name of the text file containing the simulated data.
% PathSIM:     Full path to where the simulated data file is located.
%
% OUTPUT:
% Degree:   Vector containing angles (in degrees) corresponding to every
%           data point.
% R:        Vector containing the normalised reflectans.
% Plot:     Plot of normalised reflectance vs. angle.
%
%
% Author:   Christoffer Raad (christoffer.raad@gmail.com)
% Date:     13/03/2017

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Importing and compute measured data:

addpath(Path);

% Defining baground (read out from a bagground file):
bg_0 = 0.000297529;

% Loading data as text:
FID = fopen(strcat(Path,FileName));
DataCell = textscan(FID,'%s','Delimiter','\n');
fclose(FID);

% Finding scan flags:
ScanFlags = find(contains(DataCell{1},'#S'));

% Pre-allocating physical memory:
LengthOfScans = nan(length(ScanFlags)-1,1);
CumLengthOfScans = nan(length(ScanFlags)-1,1);

% Reading out angels, double angles, counts, scantime, filter position and
% rejected channels of the measurements. Also reading channelnumber,
% channelcounts, filter position and scantime of the direct beam:
for i = 1:length(ScanFlags)
    FID = fopen(strcat(Path,FileName));
    Data = textscan(FID, '%f%f%f%f', 'HeaderLines', ScanFlags(i)+12,...
                    'CollectOutput', 1);
    fclose(FID);

```

```
56      if i == 1
57          LengthOfScans(i) = (ScanFlags(i+1)-2)-(ScanFlags(i)+12);
58          CumLengthOfScans(i) = sum(LengthOfScans(1:i,1));
59
60          % Angle:
61          D(1:CumLengthOfScans(i),1) = Data{1}(:,1);
62
63          % Double angle:
64          D(1:CumLengthOfScans(i),2) = Data{1}(:,2);
65
66          % Counts:
67          D(1:CumLengthOfScans(i),3) = Data{1}(:,3);
68
69          % Time:
70          D(1:CumLengthOfScans(i),4) = Data{1}(:,4)*10^(-3);
71
72          % Filter:
73          D(1:CumLengthOfScans(i),5) = str2double(DataCell{1}...
74                                       {ScanFlags(i)+8}(5:7));
75          % Cannels (Xrejection):
76          D(1:CumLengthOfScans(i),6) = str2double(DataCell{1}...
77                                       {ScanFlags(i)+11}(16:19));
78
79          tmp = length(DataCell{1}{ScanFlags(i)+11});
80          D(1:CumLengthOfScans(i),7) = str2double(DataCell{1}...
81                                       {ScanFlags(i)+11}(20:tmp));
82
83      elseif i == length(ScanFlags)
84          FID = fopen(strcat(Path,FileName));
85          Total = textscan(FID, '%f%f', 'HeaderLines', ScanFlags(end)+12,...
86                           'CollectOutput', 1);
87          fclose(FID);
88
89          % Channel:
90          B(1:length(Total{1}),1) = Total{1}(:,1);
91
92          % Counts:
93          B(1:length(Total{1}),2) = Total{1}(:,2);
94
95          % Filter:
96          B(1:length(Total{1}),3) = str2double(DataCell{1}...
97                                    {ScanFlags(end)+8}(5:7));
98
99          % Time:
100         B(1:length(Total{1}),4) = str2double(DataCell{1}...
101                                   {ScanFlags(end)+2}(3:4));
102
103     else
104         LengthOfScans(i) = (ScanFlags(i+1)-2)-(ScanFlags(i)+12);
105         CumLengthOfScans(i) = sum(LengthOfScans(1:i,1));
106
107         % Angle:
108         D(CumLengthOfScans(i-1)+1:CumLengthOfScans(i),1) = Data{1}(:,1);
109
110         % Double angle:
111         D(CumLengthOfScans(i-1)+1:CumLengthOfScans(i),2) = Data{1}(:,2);
112
113         % Counts:
114         D(CumLengthOfScans(i-1)+1:CumLengthOfScans(i),3) = Data{1}(:,3);
```

```
115
116         % Time:
117         D(CumLengthOfScans(i-1)+1:CumLengthOfScans(i),4) = ...
118         Data{1}(:,4)*10^(-3);
119
120         % Filter:
121         D(CumLengthOfScans(i-1)+1:CumLengthOfScans(i),5) = ...
122         str2double(DataCell{1}{ScanFlags(i)+8}(5:7));
123
124         % Channels:
125         D(1:CumLengthOfScans(i),6) = str2double(DataCell{1}...
126         {ScanFlags(i)+11}(16:19));
127         tmp = length(DataCell{1}{ScanFlags(i)+11});
128         D(1:CumLengthOfScans(i),7) = str2double(DataCell{1}...
129         {ScanFlags(i)+11}(20:tmp));
130     end
131 end
132
133 % Calculating counts in each channel from direct beam:
134 for i = 1:length(B(:,2))
135     lspc(i,1) = (B(i,2)./B(i,4)-bg_0)*104.8;
136     if lspc(i,1) < 0
137         lspc(i,1) = 0;
138     end
139 end
140
141 % Pre-allocating physical memory
142 outd = nan(length(D),1);
143 R = nan(length(D),1);
144
145 % Calculating counts per second at each angle and the corresponding direct
146 % photon count (for normalising) and compute the normalised refectance:
147 for i = 1:length(D)
148
149     switch D(i,5)
150
151         case -24
152             outd(i,1) = (D(i,3)/D(i,4)-bg_0*(D(i,7)-D(i,6)))*104.8;
153             io = sum(lspc(D(i,6):D(i,7),1));
154             if outd(i,1) < 0
155                 outd(i,1) = 0;
156             end
157             sigma(i,1) = sqrt(D(i,3)/D(i,4))/D(i,4)*104.8;
158             iosigma = sqrt(io)/B(i,4);
159             E(i,1) = outd(i,1)./io.*sqrt((sigma(i,1)./outd(i,1)).^2+(iosigma./io).^2);
160             R(i,1) = outd(i,1)/io;
161
162         case -28
163             outd(i,1) = (D(i,3)/D(i,4)-bg_0*(D(i,7)-D(i,6)))*34.99;
164             io = sum(lspc(D(i,6):D(i,7),1));
165             if outd(i,1) < 0
166                 outd(i,1) = 0;
167             end
168             sigma(i,1) = sqrt(D(i,3)/D(i,4))/D(i,4)*34.99;
169             iosigma = sqrt(io)/B(i,4);
170             E(i,1) = outd(i,1)./io.*sqrt((sigma(i,1)./outd(i,1)).^2+(iosigma./io).^2);
171             R(i,1) = outd(i,1)/io;
172
173         case -32
```

```
            outd(i,1) = (D(i,3)/D(i,4)-bg_0*(D(i,7)-D(i,6)))*11.04;
            io = sum(lspc(D(i,6):D(i,7),1));
            if outd(i,1) < 0
                outd(i,1) = 0;
            end
            sigma(i,1) = sqrt(D(i,3)/D(i,4))/D(i,4)*11.04;
            iosigma = sqrt(io)/B(i,4);
            E(i,1) = outd(i,1)./io.*sqrt((sigma(i,1)./outd(i,1)).^2+(iosigma./io).^2);
            R(i,1) = outd(i,1)/io;

        case -36
            outd(i,1) = (D(i,3)/D(i,4)-bg_0*(D(i,7)-D(i,6)))*3.615;
            io = sum(lspc(D(i,6):D(i,7),1));
            if outd(i,1) < 0
                outd(i,1) = 0;
            end
            sigma(i,1) = sqrt(D(i,3)/D(i,4))/D(i,4)*3.615;
            iosigma = sqrt(io)/B(i,4);
            E(i,1) = outd(i,1)./io.*sqrt((sigma(i,1)./outd(i,1)).^2+(iosigma./io).^2);
            R(i,1) = outd(i,1)/io;

        case -40
            outd(i,1) = (D(i,3)/D(i,4)-bg_0*(D(i,7)-D(i,6)))*1;
            io = sum(lspc(D(i,6):D(i,7),1));
            if outd(i,1) < 0
                outd(i,1) = 0;
            end
            sigma(i,1) = sqrt(D(i,3)/D(i,4))/D(i,4);
            iosigma = sqrt(io)/B(i,4);
            E(i,1) = outd(i,1)./io.*sqrt((sigma(i,1)./outd(i,1)).^2+(iosigma./io).^2);
            R(i,1) = outd(i,1)/io;
    end
end

% Converting into degrees:
Degree = abs(D(:,1))*10^(-3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Saving to text file:

 FID = fopen(strcat(Path,FileName,'MATLAB.txt'), 'wt');
 for i = 1:length(Degree)
 fprintf(FID,'%f\t %f\t %f \n',Degree(i),R(i),E(i));
 end
 fclose(FID);
```

B.5 FitArea

```
function [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL)

% SL case:
if length(vXL) == 6

    if length(vXL{6}) ~= 4
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
            'and DegreeLimit(2) are mandatory and have to be set ',...
            'to some degree values other than nan. DegreeLimit(3)',...
            'and DegreeLimit(4) have the option to be set to nans. ',...
            'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
            'to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])

    elseif ~isnan(vXL{6}(1)) && ~isnan(vXL{6}(2)) && ...
            ~isnan(vXL{6}(3)) && ~isnan(vXL{6}(4))

        [~, LowerDegreeLimit1] = min(abs(Degree-vXL{6}(1)));
        [~, UpperDegreeLimit2] = min(abs(Degree-vXL{6}(2)));
        [~, LowerDegreeLimit3] = min(abs(Degree-vXL{6}(3)));
        [~, UpperDegreeLimit4] = min(abs(Degree-vXL{6}(4)));

        DegreeReduced1 = Degree(LowerDegreeLimit1:UpperDegreeLimit2);
        RReduced1 = R(LowerDegreeLimit1:UpperDegreeLimit2);
        EReduced1 = E(LowerDegreeLimit1:UpperDegreeLimit2);

        DegreeReduced2 = Degree(LowerDegreeLimit3:UpperDegreeLimit4);
        RReduced2 = R(LowerDegreeLimit3:UpperDegreeLimit4);
        EReduced2 = E(LowerDegreeLimit3:UpperDegreeLimit4);

        DegreeReduced = vertcat(DegreeReduced1,DegreeReduced2);
        RReduced = vertcat(RReduced1,RReduced2);
        EReduced = vertcat(EReduced1,EReduced2);

    elseif isnan(vXL{6}(3)) && isnan(vXL{6}(4))

        [~, LowerDegreeLimit] = min(abs(Degree-vXL{6}(1)));
        [~, UpperDegreeLimit] = min(abs(Degree-vXL{6}(2)));

        DegreeReduced = Degree(LowerDegreeLimit:UpperDegreeLimit);
        RReduced = R(LowerDegreeLimit:UpperDegreeLimit);
        EReduced = E(LowerDegreeLimit:UpperDegreeLimit);

    else
        error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
            'and DegreeLimit(2) are mandatory and have to be set ',...
            'to some degree values other than nan. DegreeLimit(3)',...
            'and DegreeLimit(4) have the option to be set to nans. ',...
            'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
            'to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])
    end

% ML case:
elseif length(vXL) == 10
```

```
56
57      if length(vXL{10}) ~= 4
58          error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
59              'and DegreeLimit(2) are mandatory and have to be set ',...
60              'to some degree values other than nan. DegreeLimit(3)',...
61              'and DegreeLimit(4) have the option to be set to nans. ',...
62              'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
63              'to some degree values if two not continius degree ',...
64              'intervals are to be fitted and otherwise to nans.'])
65
66      elseif ~isnan(vXL{10}(1)) && ~isnan(vXL{10}(2)) && ...
67              ~isnan(vXL{10}(3)) && ~isnan(vXL{10}(4))
68
69          [~, LowerDegreeLimit1] = min(abs(Degree-vXL{10}(1)));
70          [~, UpperDegreeLimit2] = min(abs(Degree-vXL{10}(2)));
71          [~, LowerDegreeLimit3] = min(abs(Degree-vXL{10}(3)));
72          [~, UpperDegreeLimit4] = min(abs(Degree-vXL{10}(4)));
73
74          DegreeReduced1 = Degree(LowerDegreeLimit1:UpperDegreeLimit2);
75          RReduced1 = R(LowerDegreeLimit1:UpperDegreeLimit2);
76          EReduced1 = E(LowerDegreeLimit1:UpperDegreeLimit2);
77
78          DegreeReduced2 = Degree(LowerDegreeLimit3:UpperDegreeLimit4);
79          RReduced2 = R(LowerDegreeLimit3:UpperDegreeLimit4);
80          EReduced2 = E(LowerDegreeLimit3:UpperDegreeLimit4);
81
82          DegreeReduced = vertcat(DegreeReduced1,DegreeReduced2);
83          RReduced = vertcat(RReduced1,RReduced2);
84          EReduced = vertcat(EReduced1,EReduced2);
85
86      elseif isnan(vXL{10}(3)) && isnan(vXL{10}(4))
87
88          [~, LowerDegreeLimit] = min(abs(Degree-vXL{10}(1)));
89          [~, UpperDegreeLimit] = min(abs(Degree-vXL{10}(2)));
90
91          DegreeReduced = Degree(LowerDegreeLimit:UpperDegreeLimit);
92          RReduced = R(LowerDegreeLimit:UpperDegreeLimit);
93          EReduced = E(LowerDegreeLimit:UpperDegreeLimit);
94
95      else
96          error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
97              'and DegreeLimit(2) are mandatory and have to be set ',...
98              'to some degree values other than nan. DegreeLimit(3)',...
99              'and DegreeLimit(4) have the option to be set to nans. ',...
100             'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
101             'to some degree values if two not continius degree ',...
102             'intervals are to be fitted and otherwise to nans.'])
103     end
104
105 else
106     error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
107         'and DegreeLimit(2) are mandatory and have to be set ',...
108         'to some degree values other than nan. DegreeLimit(3)',...
109         'and DegreeLimit(4) have the option to be set to nans. ',...
110         'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
111         'to some degree values if two not continius degree ',...
112         'intervals are to be fitted and otherwise to nans.'])
113 end
```

B.6 Bounds

```
function [vXL,N] = Bounds(m,lb,ub,vXL)

% SL case:
if length(vXL) == 6

    % Allocating physical memory:
    flaglbedge = nan(2,1);
    flagubedge = nan(2,1);

    flaglb = isnan(lb);
    flagub = isnan(ub);

    for i = 1:2
        flaglbedge(i,1) = isequal(lb(i),m(i));
        flagubedge(i,1) = isequal(ub(i),m(i));
    end

    if flaglb(1) == 1 || flagub(1) == 1
        vXL{4} = m(1,1);
        N(1,1) = 1;
    elseif flaglbedge(1) == 1
        vXL{4} = linspace(lb(1,1),ub(1,1),40);
        N(1,1) = 1;
    elseif flagubedge(1) == 1
        vXL{4} = linspace(lb(1,1),ub(1,1),40);
        N(1,1) = 40;
    else
        eqspacedlb = linspace(lb(1,1),m(1,1),...
            round((m(1,1)-lb(1,1))/(ub(1,1)-lb(1,1))*41));
        eqspacedub = linspace(m(1,1),ub(1,1),...
            round((ub(1,1)-m(1,1))/(ub(1,1)-lb(1,1))*41));
        N1 = round((m(1,1)-lb(1,1))/(ub(1,1)-lb(1,1))*41);
        vXL{4} = [eqspacedlb,eqspacedub(2:end)];
        N(1,1) = N1;
    end

    if flaglb(2) == 1 || flagub(2) == 1
        vXL{5} = m(2,1);
        N(2,1) = 1;
    elseif flaglbedge(2) == 1
        vXL{5} = linspace(lb(2,1),ub(2,1),40);
        N(2,1) = 1;
    elseif flagubedge(2) == 1
        vXL{5} = linspace(lb(2,1),ub(2,1),40);
        N(2,1) = 40;
    else
        eqspacedlb = linspace(lb(2,1),m(2,1),...
            round((m(2,1)-lb(2,1))/(ub(2,1)-lb(2,1))*41));
        eqspacedub = linspace(m(2,1),ub(2,1),...
            round((ub(2,1)-m(2,1))/(ub(2,1)-lb(2,1))*41));
        N2 = round((m(2,1)-lb(2,1))/(ub(2,1)-lb(2,1))*41);
        vXL{5} = [eqspacedlb,eqspacedub(2:end)];
        N(2,1) = N2;
    end

```

```
    % ML case:
elseif length(vXL) == 10

    % Allocating physical memory:
    flaglbedge = nan(4,1);
    flagubedge = nan(4,1);

    flaglb = isnan(lb);
    flagub = isnan(ub);

    for i = 1:4
        flaglbedge(i,1) = isequal(lb(i),m(i));
        flagubedge(i,1) = isequal(ub(i),m(i));
    end

    if flaglb(1) == 1 || flagub(1) == 1
        vXL{7} = m(1,1);
        N(1,1) = 1;
    elseif flaglbedge(1) == 1
        vXL{7} = linspace(lb(1,1),ub(1,1),40);
        N(1,1) = 1;
    elseif flagubedge(1) == 1
        vXL{7} = linspace(lb(1,1),ub(1,1),40);
        N(1,1) = 40;
    else
        eqspacedlb = linspace(lb(1,1),m(1,1),...
            round((m(1,1)-lb(1,1))/(ub(1,1)-lb(1,1))*41));
        eqspacedub = linspace(m(1,1),ub(1,1),...
            round((ub(1,1)-m(1,1))/(ub(1,1)-lb(1,1))*41));
        N1 = round((m(1,1)-lb(1,1))/(ub(1,1)-lb(1,1))*41);
        vXL{7} = [eqspacedlb,eqspacedub(2:end)];
        N(1,1) = N1;
    end

    if flaglb(2) == 1 || flagub(2) == 1
        vXL{8} = m(2,1);
        N(2,1) = 1;
    elseif flaglbedge(2) == 1
        vXL{8} = linspace(lb(2,1),ub(2,1),40);
        N(2,1) = 1;
    elseif flagubedge(2) == 1
        vXL{8} = linspace(lb(2,1),ub(2,1),40);
        N(2,1) = 40;
    else
        eqspacedlb = linspace(lb(2,1),m(2,1),...
            round((m(2,1)-lb(2,1))/(ub(2,1)-lb(2,1))*41));
        eqspacedub = linspace(m(2,1),ub(2,1),...
            round((ub(2,1)-m(2,1))/(ub(2,1)-lb(2,1))*41));
        N2 = round((m(2,1)-lb(2,1))/(ub(2,1)-lb(2,1))*41);
        vXL{8} = [eqspacedlb,eqspacedub(2:end)];
        N(2,1) = N2;
    end

    if flaglb(3) == 1 || flagub(3) == 1
        vXL{4} = m(3,1);
        N(3,1) = 1;
    elseif flaglbedge(3) == 1
        vXL{4} = linspace(lb(3,1),ub(3,1),40);
        N(3,1) = 1;
```

```
115     elseif flagubedge(3) == 1
116         vXL{4} = linspace(lb(3,1),ub(3,1),40);
117         N(3,1) = 40;
118     else
119         eqspacedlb = linspace(lb(3,1),m(3,1),...
120             round((m(3,1)-lb(3,1))/(ub(3,1)-lb(3,1))*41));
121         eqspacedub = linspace(m(3,1),ub(3,1),...
122             round((ub(3,1)-m(3,1))/(ub(3,1)-lb(3,1))*41));
123         N3 = round((m(3,1)-lb(3,1))/(ub(3,1)-lb(3,1))*41);
124         vXL{4} = [eqspacedlb,eqspacedub(2:end)];
125         N(3,1) = N3;
126     end
127 
128     if flaglb(4) == 1 || flagub(4) == 1
129         vXL{6} = m(4,1);
130         N(4,1) = 1;
131     elseif flaglbedge(4) == 1
132         vXL{6} = linspace(lb(4,1),ub(4,1),40);
133         N(4,1) = 1;
134     elseif flagubedge(4) == 1
135         vXL{6} = linspace(lb(4,1),ub(4,1),40);
136         N(4,1) = 40;
137     else
138         eqspacedlb = linspace(lb(4,1),m(4,1),...
139             round((m(4,1)-lb(4,1))/(ub(4,1)-lb(4,1))*41));
140         eqspacedub = linspace(m(4,1),ub(4,1),...
141             round((ub(4,1)-m(4,1))/(ub(4,1)-lb(4,1))*41));
142         N4 = round((m(4,1)-lb(4,1))/(ub(4,1)-lb(4,1))*41);
143         vXL{6} = [eqspacedlb,eqspacedub(2:end)];
144         N(4,1) = N4;
145     end
146 
147     % Error with input variables:
148 else
149     error(['lb, m and ub must be of size(2,1) for SL or of size(4,1) for ML. If a ',...
150            'parameter should be fixed, lb and ub should be set to NaN and the ',...
151            'parameter takes the value specified in m.']);
152 end
153 end
```

B.7 DataReducing

```
function [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL)

% SL case:
if length(vXL) == 6

    if length(vXL{6}) ~= 4
        error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
            'and DegreeLimit(2) are mandatory and have to be set ',...
            'to some degree values other than nan. DegreeLimit(3)',...
            'and DegreeLimit(4) have the option to be set to nans. ',...
            'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
            'to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])

    elseif ~isnan(vXL{6}(1)) && ~isnan(vXL{6}(2)) && ...
            ~isnan(vXL{6}(3)) && ~isnan(vXL{6}(4))

        [~, LowerDegreeLimit1] = min(abs(Degree-vXL{6}(1)));
        [~, UpperDegreeLimit2] = min(abs(Degree-vXL{6}(2)));
        [~, LowerDegreeLimit3] = min(abs(Degree-vXL{6}(3)));
        [~, UpperDegreeLimit4] = min(abs(Degree-vXL{6}(4)));

        DegreeReduced1 = Degree(LowerDegreeLimit1:UpperDegreeLimit2);
        RReduced1 = R(LowerDegreeLimit1:UpperDegreeLimit2);
        EReduced1 = E(LowerDegreeLimit1:UpperDegreeLimit2);

        DegreeReduced2 = Degree(LowerDegreeLimit3:UpperDegreeLimit4);
        RReduced2 = R(LowerDegreeLimit3:UpperDegreeLimit4);
        EReduced2 = E(LowerDegreeLimit3:UpperDegreeLimit4);

        DegreeReduced = vertcat(DegreeReduced1,DegreeReduced2);
        RReduced = vertcat(RReduced1,RReduced2);
        EReduced = vertcat(EReduced1,EReduced2);

    elseif isnan(vXL{6}(3)) && isnan(vXL{6}(4))

        [~, LowerDegreeLimit] = min(abs(Degree-vXL{6}(1)));
        [~, UpperDegreeLimit] = min(abs(Degree-vXL{6}(2)));

        DegreeReduced = Degree(LowerDegreeLimit:UpperDegreeLimit);
        RReduced = R(LowerDegreeLimit:UpperDegreeLimit);
        EReduced = E(LowerDegreeLimit:UpperDegreeLimit);

    else
        error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
            'and DegreeLimit(2) are mandatory and have to be set ',...
            'to some degree values other than nan. DegreeLimit(3)',...
            'and DegreeLimit(4) have the option to be set to nans. ',...
            'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
            'to some degree values if two not continius degree ',...
            'intervals are to be fitted and otherwise to nans.'])
    end

% ML case:
elseif length(vXL) == 10
```

```
56
57      if length(vXL{10}) ~= 4
58          error(['DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
59              'and DegreeLimit(2) are mandatory and have to be set ',...
60              'to some degree values other than nan. DegreeLimit(3)',...
61              'and DegreeLimit(4) have the option to be set to nans. ',...
62              'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
63              'to some degree values if two not continius degree ',...
64              'intervals are to be fitted and otherwise to nans.'])
65
66      elseif ~isnan(vXL{10}(1)) && ~isnan(vXL{10}(2)) && ...
67              ~isnan(vXL{10}(3)) && ~isnan(vXL{10}(4))
68
69          [~, LowerDegreeLimit1] = min(abs(Degree-vXL{10}(1)));
70          [~, UpperDegreeLimit2] = min(abs(Degree-vXL{10}(2)));
71          [~, LowerDegreeLimit3] = min(abs(Degree-vXL{10}(3)));
72          [~, UpperDegreeLimit4] = min(abs(Degree-vXL{10}(4)));
73
74          DegreeReduced1 = Degree(LowerDegreeLimit1:UpperDegreeLimit2);
75          RReduced1 = R(LowerDegreeLimit1:UpperDegreeLimit2);
76          EReduced1 = E(LowerDegreeLimit1:UpperDegreeLimit2);
77
78          DegreeReduced2 = Degree(LowerDegreeLimit3:UpperDegreeLimit4);
79          RReduced2 = R(LowerDegreeLimit3:UpperDegreeLimit4);
80          EReduced2 = E(LowerDegreeLimit3:UpperDegreeLimit4);
81
82          DegreeReduced = vertcat(DegreeReduced1,DegreeReduced2);
83          RReduced = vertcat(RReduced1,RReduced2);
84          EReduced = vertcat(EReduced1,EReduced2);
85
86      elseif isnan(vXL{10}(3)) && isnan(vXL{10}(4))
87
88          [~, LowerDegreeLimit] = min(abs(Degree-vXL{10}(1)));
89          [~, UpperDegreeLimit] = min(abs(Degree-vXL{10}(2)));
90
91          DegreeReduced = Degree(LowerDegreeLimit:UpperDegreeLimit);
92          RReduced = R(LowerDegreeLimit:UpperDegreeLimit);
93          EReduced = E(LowerDegreeLimit:UpperDegreeLimit);
94
95      else
96          error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
97              'and DegreeLimit(2) are mandatory and have to be set ',...
98              'to some degree values other than nan. DegreeLimit(3)',...
99              'and DegreeLimit(4) have the option to be set to nans. ',...
100             'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
101             'to some degree values if two not continius degree ',...
102             'intervals are to be fitted and otherwise to nans.'])
103     end
104
105 else
106     error(['test DegreeLimit has to contain 4 limits. DegreeLimit(1) ',...
107         'and DegreeLimit(2) are mandatory and have to be set ',...
108         'to some degree values other than nan. DegreeLimit(3)',...
109         'and DegreeLimit(4) have the option to be set to nans. ',...
110         'DegreeLimit(3) and DegreeLimit(4) should be defined ',...
111         'to some degree values if two not continius degree ',...
112         'intervals are to be fitted and otherwise to nans.'])
113 end
```

B.8 CorrelatedLooping

```
function [Out,CHI] = CorrelatedLooping(m,lb,ub,vXL)
%% CorrelatedLooping is a function that is brute force fitting a
% reflection model to XRR data for varius values of the model parameters. For SL's the
% model parameters are the thickness and the roughness. For ML's the model parameters
% are the D-spacing, Gamma, Sigma1 and Sigma2.
% In the SL case the model parameters are fitted simultaniusly and the optimal
% parameters are found where the chi^2 value is lowest. In the ML case are the D-spacing
% and Gamma fitted simultaniusly. The optimal values of D and Gamma is then passed on
% and the Sigmas are fitted simultaniusly. The optimal parameters are found where the
% chi^2 is minimum.

% INPUT:
% m:       Model vector of size(2,1) for SL or size(4,1) for ML. Contains the initial
%          values of the model parameters. For SL, m(1,1) is the thickness of the film
%          and m(2,1) is the roughness.
% lb:      Lower bound vector with the same size as m. Contains the lowest values that
%          the model parameters can assume.
% ub:      Upper bound vector with the same size as m. Contains the largest values that
%          the model parameters can assume.
% vXL:     Cell of size(1,6) for SL and size(1,10) for ML. Contains the film
%          specifications, such as materials, number of layers and gracing angle fitting
%          interval.
%
% OUTPUT:
% Out: Output model matrix containing the optimal model parameter for each iteration in
%      units of [m]. Same size as m.
% CHI: Output vector that contains the chi^2 value for each iteration.
%
% Author:   Christoffer Raad (christoffer.raad@gmail.com)
% Date:     15/06/2017

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SL case:

fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% SL case:
if length(vXL) == 6
    disp('=========================================')
    disp('Perform correlated fitting of single layer')
    disp('=========================================')

    % Calculating initial Rsynth and CHI^2:
    lbinitial = [nan    nan    nan    nan]';
    ubinitial = [nan    nan    nan    nan]';

    % Calculating initial Rsynth and CHI^2:
    [Out(:,1),CHI(:,1)] = CorrelatedFittingZS(m,lbinitial,ubinitial,vXL);
    fprintf('%s %0.2f \n','Initial chi^2:',CHI(1))

    % Definig initial values befor loop:

    i = 1;
```

```
    RelativeChange = 1;

    while RelativeChange ~= 0
        close all
        fprintf('%s %d \n','Brute force fitting z and Sigma. Iteration =',i)

        % Perform fitting:
        [Out(:,i+1),CHI(:,i+1)] = CorrelatedFittingZS(m(:,i),lb(:,i),ub(:,i),vXL);

        lb(:,i+1) = [lb(1,1)  lb(2,1)  ]';
        m(:,i+1)  = [Out(1,i) Out(2,i) ]';
        ub(:,i+1) = [ub(1,1)  ub(2,1)  ]';

        RelativeChange = ((CHI(:,i+1)-CHI(:,i))/CHI(:,i))*100;
        fprintf('%s %0.2f \n','Percentage change of chi^2:',RelativeChange)

        i = i+1;
    end
    FID = figure(1);
    print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_contourZS'),'-depsc2');
    FID = figure(2);
    print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_plot'),'-depsc2');

    % ML case:
elseif length(vXL) == 10

    disp('========================================')
    disp('Perform correlated fitting of multilayer')
    disp('========================================')

    lbinitial = [nan    nan    nan    nan]';
    ubinitial = [nan    nan    nan    nan]';

    % Calculating initial Rsynth and CHI^2:
    [Out(:,1),CHI(1)] = CorrelatedFittingDG(m,lbinitial,ubinitial,vXL);
    fprintf('%s %0.2f \n','Initial chi^2:',CHI(1))

    % Definig initial values befor loop:
    Out(:,3) = Out(:,1);
    CHI(3) = CHI(1);
    m(:,2) = m(:,1);
    lb(:,2) = lb(:,1);
    ub(:,2) = ub(:,1);

    % Definig initial values befor loop:
    i = 1;
    RelativeChange = 1;

    % Looping:
    while RelativeChange ~= 0
        close all
        fprintf('%s %d \n','Brute force fitting D-spacing and Gamma. Iteration =',i)

        % Perform fitting:
        [Out(:,i*2),CHI(:,i*2)] = CorrelatedFittingDG(m(:,i*2-1),lb(:,i*2-1),...
                                                      ub(:,i*2-1),vXL);
        % Calculating change of chi^2:
        RelativeChange = ((CHI(:,i*2-1)/CHI(:,i*2))/CHI(:,i*2))*100;
        fprintf('%s %0.2f \n','Percentage change of chi^2:',RelativeChange)
```

```
115         fprintf('\n')
116
117         lb(:,i*2) = [lb(1,i+1)  lb(2,i+1)   lb(3,i+1)  lb(4,i+1)  ]';
118         m(:,i*2)  = [Out(1,i*2) Out(2,i*2)  Out(3,i*2) Out(4,i*2) ]';
119         ub(:,i*2) = [ub(1,i+1)  ub(2,i+1)   ub(3,i+1)  ub(4,i+1)  ]';
120
121         fprintf('%s %d \n','Brute force fitting Sigma1 and Sigma2. Iteration =',i)
122
123         [Out(:,i*2+1),CHI(i*2+1)] = CorrelatedFittingSS(m(:,i*2),lb(:,i*2),...
124                                                         ub(:,i*2),vXL);
125
126         lb(:,i*2+1) = [lb(1,i*2)    lb(2,i*2)    lb(3,i*2)    lb(4,i*2)    ]';
127         m(:,i*2+1)  = [Out(1,i*2+1) Out(2,i*2+1) Out(3,i*2+1) Out(4,i*2+1) ]';
128         ub(:,i*2+1) = [ub(1,i*2)    ub(2,i*2)    ub(3,i*2)    ub(4,i*2)    ]';
129
130         RelativeChange = ((CHI(:,i*2)-CHI(:,i*2+1))/CHI(:,i*2+1))*100;
131         fprintf('%s %0.2f \n','Percentage change of chi^2:',RelativeChange)
132         fprintf('\n')
133
134         i = i+1;
135
136     end
137      FID = figure(1);
138      print(FID,'D:\Dropbox\Master\Thesis\Pics\FitEx\SI6478DG1','-depsc2');
139      print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_contourDG'),'-depsc2');
140      FID = figure(3);
141      print(FID,'D:\Dropbox\Master\Thesis\Pics\FitEx\SI6478SS1','-depsc2');
142      print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_contourSS'),'-depsc2');
143      FID = figure(4);
144      print(FID,'D:\Dropbox\Master\Thesis\Pics\FitEx\SI6478plot1','-depsc2');
145      print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_plot'),'-depsc2');
146
147 else
148     error(['The model parameter, m, must be of size 2x1 for single layers and 4x1',...
149            'for multilayers.']);
150 end
151 end
```

B.9 UncorrelatedLooping

```
function [Out,CHI] = UncorrelatedLooping(m,lb,ub,vXL,BraggNo)
%% UncorrelatedLooping is a function that is brute force fitting a reflection model to
% XRR data. The fit is evaluated by the minimum least square chi^2. The optimal fit is
% found by an iterative process which is stopped when the chi^2 value do not change.

% INPUT:
% m:       Model vector of size(2,1) for SL or size(4,1) for ML. Contains
%          the initial values of the model parameters. For SL, m(1,1) is
%          the thickness of the film and m(2,1) is the roughness.
% lb:      Lower bound vector with the same size as m. Contains the lowest
%          values that the model parameters can assume.
% ub:      Upper bound vector with the same size as m. Contains the largest
%          values that the model parameters can assume.
% vXL:     Cell of size(1,6) for SL and size(1,10) for ML. Contains the
%          film specifications, such as materials, number of layers and
%          gracing angle fitting interval. For specific instructions see
%          Fittingscript.
% BraggNo: Matrix of Bragg peaks numbers which are to be fitted.
%
% OUTPUT:
% Out: Output model vector containing the best model parameter in units of [m]. Same
%      size as m.
%
% Author:   Christoffer Raad (christoffer.raad@gmail.com)
% Date:     15/06/2017

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% SL case:
if length(vXL) == 6
    disp('=========================================')
    disp('Perform uncorrelated fitting of single layer')
    disp('=========================================')

    lbinitial = [nan nan]';
    ubinitial = [nan nan]';

    % Calculating initial chi^2:
    [Out(:,1),CHI(1)] = Uncorrelatedfitting(m,lbinitial,ubinitial,vXL,BraggNo);
    fprintf('%s %0.2f \n','Initial chi^2:',CHI(1))

    % Definig initial values befor loop:
    i = 1;
    RelativeChange = 1;

    % Looping:
    while RelativeChange ~= 0
        close all
        fprintf('%s %d \n','Iteration =',i)

        % Perform fitting:
        [Out(:,i+1),CHI(i+1)] = Uncorrelatedfitting(m(:,i),lb(:,i),ub(:,i),vXL,BraggNo);
```

```
56
57          % Calculating initial chi^2:
58          RelativeChange = ((CHI(i+1)-CHI(i))/CHI(i))*100;
59          fprintf('%s %0.2f \n','Relative percentage change of chi^2:',RelativeChange);
60
61          % Defining new bounds:
62          lb(:,i+1) = [lb(1,1)    lb(2,1)    ]';
63          m(:,i+1)  = [Out(1,i+1) Out(2,i+1) ]';
64          ub(:,i+1) = [ub(1,1)    ub(2,1)    ]';
65
66          i = i+1;
67      end
68
69      FID = figure(3);
70      print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_UnCorPlot'),'-depsc2');
71
72      % ML case:
73  elseif length(vXL) == 10
74      disp('========================================')
75      disp('Perform uncorrelated fitting of multilayer')
76      disp('========================================')
77
78      lbinitial = [nan nan nan nan]';
79      ubinitial = [nan nan nan nan]';
80
81      % Calculating initial chi^2:
82      [Out(:,1),CHI(1)] = Uncorrelatedfitting(m,lbinitial,ubinitial,vXL,BraggNo);
83      fprintf('%s %0.2f \n','Initial chi^2:',CHI(1))
84
85      % Definig initial values befor loop:
86      i = 1;
87      RelativeChange = 1;
88
89      % Looping:
90      while RelativeChange ~= 0
91          close all
92          fprintf('%s %d \n','Iteration =',i)
93
94          % Perform fitting:
95          [Out(:,i+1),CHI(i+1)] = Uncorrelatedfitting(m(:,i),lb(:,i),ub(:,i),vXL,BraggNo);
96
97          % Calculating change of chi^2:
98          RelativeChange = ((CHI(i+1)-CHI(i))/CHI(i))*100;
99          fprintf('%s %0.2f \n','Relative percentage change of chi^2:',RelativeChange);
100
101         lb(:,i+1) = [lb(1,1)    lb(2,1)    lb(3,1)    lb(4,1)    ]';
102         m(:,i+1)  = [Out(1,i+1) Out(2,i+1) Out(3,i+1) Out(4,i+1)]';
103         ub(:,i+1) = [ub(1,1)    ub(2,1)    ub(3,1)    ub(4,1)    ]';
104
105         i = i+1;
106     end
107     % Saving eps file:
108     FID = figure(5);
109     print(FID,strcat(WitnessSample{1}{2},WitnessSample{1}{1},'_UnCorPlot'),'-depsc2');
110 end
```

B.10 CorrelatedFittingDG

```
function [Out,CHIbest] = CorrelatedFittingDG(m,lb,ub,vXL)

% Loading witness sample specifications from 'FileToBeFitted.txt':
fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% Loading XRR data:
[Degree, R, E] = LoadXRRData(WitnessSample{1}{1}, WitnessSample{1}{2});

%
[vXL,N] = Bounds(m,lb,ub,vXL);

% Only using data within DegreeLimit specifications:
[DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

% Calculating film definitions:
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

% Pre-allocation physical memory:
CHI = nan(length(vXL{7}),length(vXL{8}));

% Compute reflectance and chi^2:
for i = 1:length(vXL{7})
    for j = 1:length(vXL{8})
        Rsynth = IMD(ThetaI,NumberOfLayers,n,d(:,i,j),sigma(:,N(3),N(4)),lambda);
        CHI(i,j) = nansum((Rsynth-RReduced).^2./EReduced.^2);
    end
end

% Finding values of D and Gamma for which chi^2 is minimum:
[valDspacing,idxDspacing] = min(CHI);
[~,idxGamma] = min(valDspacing);

if length(vXL{8})~=1 && length(vXL{7})~=1
    figure('units','normalized','OuterPosition',[0 0 1 1])
    hold on
    contour(vXL{8}(1,:),vXL{7}(1,:),log(CHI/length(RReduced)),40);
    plot(vXL{8}(1,idxGamma),vXL{7}(1,idxDspacing(idxGamma)),'k.','MarkerSize',9);
    set(gca,'FontWeight','Normal','FontSize',20,'FontName','Times');
    xlabel('Bilayer thickness ratio, $$\Gamma$$','Interpreter','Latex');
    ylabel('D-spacing, D, $$[\mathrm{m}]$$','Interpreter','Latex');
    colormap('jet');
    h = colorbar;
    xlabel(h,'$$\mathrm{log\left(\chi^{2}\right)}$$','FontWeight','Normal','FontSize',...
           20,'FontName','Times','Interpreter','Latex');
    hold off
end

% Calculating best fit and corresponding chi^2:
Rsynthbest = IMD(ThetaI,NumberOfLayers,n,d(:,idxDspacing(idxGamma),idxGamma),...
     sigma(:,N(3),N(4)),lambda);
CHIbest = nansum((Rsynthbest-RReduced).^2./EReduced.^2);

% Plotting reflectance and best fit:
```

```
D = vXL{7}(1,idxDspacing(idxGamma))*10^(10);
Gamma = vXL{8}(1,idxGamma);
Sigma1 = vXL{4}(1,N(3))*10^(10);
Sigma2 = vXL{6}(1,N(4))*10^(10);

legendstr1 = sprintf('%s %0.2f %s','D =',D,'[$$\mathrm{\AA}$$] , ');
legendstr2 = sprintf('%s %0.4f %s','$$\Gamma$$ =',Gamma,', ');
legendstr3 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{B_{4}C}}$$ =',Sigma1,...
    '[$$\mathrm{\AA}$$] , ');
legendstr4 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{Ni}}$$ =',Sigma2,...
    '[$$\mathrm{\AA}$$] , ');
legendstr5 = sprintf('%s %0.2f %s %0.0f %s %0.2f ','$$\chi^{2}$$ = $$\frac{',CHIbest,...
    '}{',length(RReduced),'}$$ =',CHIbest/length(RReduced));

figure('units','normalized','OuterPosition',[0 0 1 1])
hold on
plot(Degree,R,'k.-');
plot(DegreeReduced,Rsynthbest,'b-');
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([0 4 10^(-8) 1])
xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter','Latex');
ylabel('Reflectance, R','Interpreter','Latex');
legend({'Measured data',['Fit; ',legendstr1,legendstr2,legendstr3,legendstr4,...
    legendstr5]},'Location','South','FontSize',15,'Interpreter','Latex');
hold off

Out(1,1) = vXL{7}(1,idxDspacing(idxGamma));
Out(2,1) = vXL{8}(1,idxGamma);
Out(3,1) = vXL{4}(1,N(3));
Out(4,1) = vXL{6}(1,N(4));
CHIbest = CHIbest/length(RReduced);
end
```

B.11 CorrelatedFittingSS

```
function [Out,CHIbest] = CorrelatedFittingSS(m,lb,ub,vXL)

% Loading witness sample specifications from 'FileToBeFitted.txt':
fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% Loading XRR data:
[Degree, R, E] = LoadXRRData(WitnessSample{1}{1}, WitnessSample{1}{2});

%
[vXL,N] = Bounds(m,lb,ub,vXL);

% Only using data within DegreeLimit specifications:
[DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

% Calculating film definitions:
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

% Pre-allocation physical memory:
CHI = nan(length(vXL{4}),length(vXL{6}));

% Compute reflectance and chi^2:
for i = 1:length(vXL{4})
    for j = 1:length(vXL{6})
        Rsynth = IMD(ThetaI,NumberOfLayers,n,d(:,N(1),N(2)),sigma(:,i,j),lambda);
        CHI(i,j) = nansum((Rsynth-RReduced).^2./EReduced.^2);
    end
end

% Manual fitting D-spacing and Gamma:
[valSigma1,idxSigma1] = min(CHI);
[~,idxSigma2] = min(valSigma1);

if length(vXL{4})~=1 && length(vXL{6})~=1
    figure('units','normalized','OuterPosition',[0 0 1 1])
    hold on
    contour(vXL{6}(1,:),vXL{4}(1,:),log(CHI/length(RReduced)),60);
    plot(vXL{6}(1,idxSigma2),vXL{4}(1,idxSigma1(idxSigma2)),'k.','MarkerSize',9);
    set(gca,'FontWeight','Normal','FontSize',20,'FontName','Times');
    xlabel('Roughness, $$\mathrm{\sigma_{Ni}}$$, $$[\mathrm{m}]$$','Interpreter',...
        'Latex');
    ylabel('Roughness, $$\mathrm{\sigma_{B_{4}C}}$$, $$[\mathrm{m}]$$','Interpreter',...
        'Latex');
    colormap('jet');
    h = colorbar;
    xlabel(h,'$$\mathrm{log\left(\chi^{2}\right)}$$','FontWeight','Normal','FontSize',...
        20,'FontName','Times','Interpreter','Latex') ;
    hold off
end

% Calculating best fit and corresponding chi^2:
Rsynthbest = IMD(ThetaI,NumberOfLayers,n,d(:,N(1),N(2)),sigma(:,idxSigma1(idxSigma2),...
    idxSigma2),lambda);
CHIbest = nansum((Rsynthbest-RReduced).^2./EReduced.^2);
```

```
56
57 D = vXL{7}(1,N(1))*10^(10);
58 Gamma = vXL{8}(1,N(2));
59 Sigma1 = vXL{4}(1,idxSigma1(idxSigma2))*10^(10);
60 Sigma2 = vXL{6}(1,idxSigma2)*10^(10);
61
62 legendstr1 = sprintf('%s %0.2f %s','D =',D,'[$$\mathrm{\AA}$$] , ');
63 legendstr2 = sprintf('%s %0.4f %s','$$\Gamma$$ =',Gamma,', ');
64 legendstr3 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{B_{4}C}}$$ =',Sigma1,...
65     '[$$\mathrm{\AA}$$] , ');
66 legendstr4 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{Ni}}$$ =',Sigma2,...
67     '[$$\mathrm{\AA}$$] , ');
68 legendstr5 = sprintf('%s %0.2f %s %0.0f %s %0.2f ','$$\chi^{2}$$ = $$\frac{',...
69     CHIbest,'}{',length(RReduced),'}$$ =',CHIbest/length(RReduced));
70
71 figure('units','normalized','OuterPosition',[0 0 1 1])
72 hold on
73 errorbar(Degree,R,E,'k.-');
74 plot(DegreeReduced,Rsynthbest,'b-');
75 set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
76 axis([0 4 10^(-8) 1])
77 xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter','Latex');
78 ylabel('Reflectance, R','Interpreter','Latex');
79 legend({'Measured data',['Fit; ',legendstr1,legendstr2,legendstr3,legendstr4,...
80     legendstr5]},'Location','South','FontSize',15,'Interpreter','Latex');
81 grid on
82 hold off
83
84 Out(1,1) = vXL{7}(1,N(1));
85 Out(2,1) = vXL{8}(1,N(2));
86 Out(3,1) = vXL{4}(1,idxSigma1(idxSigma2));
87 Out(4,1) = vXL{6}(1,idxSigma2);
88 CHIbest = CHIbest/length(RReduced);
89 end
```

B.12 CorrelatedFittingZS

```
function [Out,CHIbest] = CorrelatedFittingZS(m,lb,ub,vXL)

% Loading witness sample specifications from 'FileToBeFitted.txt':
fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% Loading XRR data:
[Degree, R, E] = LoadXRRData(WitnessSample{1}{1}, WitnessSample{1}{2});

%
[vXL,N] = Bounds(m,lb,ub,vXL);

% Only using data within DegreeLimit specifications:
[DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

% Calculating film definitions:
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

% Pre-allocating physical memory:
CHI = nan(length(vXL{4}),length(vXL{5}));

% Compute reflectance and chi^2:
for i = 1:length(vXL{4})
    for j = 1:length(vXL{5})
        Rsynth = IMD(ThetaI,NumberOfLayers,n,d(:,i),sigma(:,j),lambda);
        CHI(i,j) = nansum((Rsynth-RReduced).^2./EReduced.^2);
    end
end

% Finding values of z and sigma for which chi^2 is minimum:
[vald,idxd] = min(CHI);
[~,idxSigma] = min(vald);

% Plotting countur plot of chi^2:
if length(vXL{4})~=1 && length(vXL{5})~=1
figure('units','normalized','OuterPosition',[0 0 1 1])
hold on
contour(sigma(1,:),d(1,:),log(CHI/length(RReduced)),40);
plot(sigma(1,idxSigma),d(1,idxd(idxSigma)),'k.','MarkerSize',9);
set(gca,'FontWeight','Normal','FontSize',20,'FontName','Times');
xlabel('Roughness, $$\mathrm{\sigma_{Ni}}$$, $$[\mathrm{m}]$$','Interpreter','Latex');
ylabel('Thickness, $$\mathrm{z_{Ni}}$$, $$[\mathrm{m}]$$','Interpreter','Latex');
colormap('jet');
h = colorbar;
xlabel(h,'$$\mathrm{log\left(\chi^{2}\right)}$$','FontWeight','Normal','FontSize',...
    20,'FontName','Times','Interpreter','Latex') ;
hold off
end

% Calculating best fit and corresponding chi^2:
Rsynthbest = IMD(ThetaI,NumberOfLayers,n,d(:,idxd(idxSigma)),...
             sigma(:,idxSigma),lambda);
CHIbest = nansum((Rsynthbest-RReduced).^2./EReduced.^2);

```

```
% Plotting reflectance and best fit:
z = d(1,idxd(idxSigma))*10^(10);
Sigma1 = sigma(1,idxSigma)*10^(10);

legendstr1 = sprintf('%s %0.2f %s','$$\mathrm{z_{Ni}}$$ =',z,'[$$\mathrm{\AA}$$] , ');
legendstr2 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{Ni}}$$ =',Sigma1,...
    '[$$\mathrm{\AA}$$] , ');
legendstr3 = sprintf('%s %0.2f %s %0.0f %s %0.2f ','$$\chi^{2}$$ = $$\frac{',...
    CHIbest,'}{',length(RReduced),'}$$ =',CHIbest/length(RReduced));

figure('units','normalized','OuterPosition',[0 0 1 1])
hold on
errorbar(Degree,R,E,'k.-');
plot(DegreeReduced,Rsynthbest,'b+-');
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([0 4 10^(-8) 1])
xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter','Latex');
ylabel('Reflectance, R','Interpreter','Latex');
legend({'Measured data',['Fit; ',legendstr1,legendstr2,legendstr3]},'Location',...
    'South','FontSize',20,'Interpreter','Latex');
hold off

% Writing out parameters:
Out(1,1) = vXL{4}(idxd(idxSigma));
Out(2,1) = vXL{5}(idxSigma);
CHIbest = CHIbest/length(RReduced);
end
```

B.13 Uncorrelatedfitting

```
function [Out,CHIbest] = Uncorrelatedfitting(m,lb,ub,vXL,BraggNo)
%% Uncorrelatedfitting is a function that is brute force fitting a % reflection model to
%  XRR data. The fit is evaluated by the minimum least square chi^2. The optimal fit is
%  found by an iterative process which is stopped when the chi^2 value do not change.

% INPUT:
% m:       Model vector of size(2,1) for SL or size(4,1) for ML. Contains the initial
%          values of the model parameters. For SL, m(1,1) is the thickness of the film
%          and m(2,1) is the roughness.
% lb:      Lower bound vector with the same size as m. Contains the lowest values that
%          the model parameters can assume.
% ub:      Upper bound vector with the same size as m. Contains the largest values that
%          the model parameters can assume.
% vXL:     Cell of size(1,6) for SL and size(1,10) for ML. Contains the film
%          specifications, such as materials, number of layers and gracing angle fitting
%          interval. For specific instructions see Fittingscript.
% BraggNo: Matrix of Bragg peak numbers which are to be fitted. Matrix of Bragg peaks
%          numbers which are to be fitted. If SL's are to be fitted BraggNo must have
%          size(2,2) and if ML's are to be fitted BraggNo must be of size(3,2). In the
%          SL case the entries in BraggNo(:,1) are used to fit the thickness of the
%          layer and the entries in BraggNo(:,2) are used to fit the roughness, although
%          it do not make sense to fit SL's to Bragg peaks.
%          In the ML case, entries in BraggNo(:,1) defines Bragg peaks for fitting of
%          the D-spcaing and BraggNo(:,2) and BraggNo(:,3) defines Bragg peaks for the
%          fitting of Sigma1 and Sigma2 respectivly. In the case where BraggNo consist
%          purly of NaNs no specific Bragg peak is fitted but the whole dataset is used.
%          This is the setting for SL's. For more infomation see the CritAndBragg
%          function.
%
% OUTPUT:
% Out:     Output model vector containing the best model parameter in units of [m]. Same
%          size as m.
% Chibest: Selfexplanatory!
%
% Author:   Christoffer Raad (christoffer.raad@gmail.com)
% Date:     15/06/2017

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Loading witness sample specifications from 'FileToBeFitted.txt':
fid = fopen('FileToBeFitted.txt');
WitnessSample = textscan(fid,'%s');
fclose(fid);

% Loading XRR data:
[Degree, R, E] = LoadXRRData(WitnessSample{1}{1}, WitnessSample{1}{2});

% Saving initial vXL for later:
vXL2 = vXL;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SL case:
if length(vXL) == 6

    % Z fitting:
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
56      BraggNoZ = BraggNo(1,:);
57      % Finding critical angle and Bragg peaks in XRR data:
58      [Low,High,CritLow,CritHigh] = FitArea(vXL,BraggNoZ);
59      vXL{6} = [CritLow CritHigh Low(1) High(1)];
60
61      % Defining bounds:
62      [vXL,N] = Bounds(m,lb,ub,vXL);
63
64      % Defining limits of x axes for later:
65      xAXLIM(1) = vXL{6}(1)-vXL{6}(1)*0.2;
66      if isnan(vXL{6}(3))
67          xAXLIM(2) = vXL{6}(2)+vXL{6}(2)*0.2;
68      else
69          xAXLIM(2) = vXL{6}(4)+vXL{6}(4)*0.2;
70      end
71
72      % Only using data within DegreeLimit specifications:
73      [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);
74
75      % Defining parameters from bounds and the corresponding film stack:
76      [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);
77
78      % Pre-allocation physical memory:
79      Rsynth = nan(length(ThetaI),1,length(vXL{4}),1,1);
80      CHI = nan(1,length(vXL{4}),1,1);
81
82      % Calculating model refelction and chi^2 for all Z:
83      for i = 1:length(vXL{4})
84          Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,d(:,i),sigma(:,N(2)),lambda);
85          CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
86      end
87
88      % Finding best Z:
89      [valZ,idxZ] = min(CHI(:));
90
91      % Plotting:
92      figure('units','normalized','OuterPosition',[0 0 1 1])
93      subplot(2,2,1)
94      hold on
95      plot(Degree,R,'k.-')
96      plot(DegreeReduced,RReduced,'ro')
97      set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
98      axis([0 1 min(R(1:50)) 1])
99      xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
100             'Latex');
101     ylabel('Reflectance, R','Interpreter','Latex');
102     title('Measurements to be fitted','Interpreter','Latex')
103     grid on
104     hold off
105
106     subplot(2,2,2)
107     hold on
108     plot(Degree,R,'k.-')
109     plot(DegreeReduced,Rsynth(:,idxZ),'b-','LineWidth',1)
110     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
111     axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1])
112     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
113            'Latex');
114     ylabel('Reflectance, R','Interpreter','Latex');
```

```
title('Fit','Interpreter','Latex')
grid on
hold off

if length(vXL{4}) ~= 1
subplot(2,1,2)
hold on
plot(vXL{4},CHI./length(RReduced),'b.-')
H2 = plot(vXL{4}(idxZ),valZ/length(RReduced),'ro','LineWidth',2);
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([vXL{4}(1) vXL{4}(end) min(CHI./length(RReduced)) max(CHI./length(RReduced))]);
xlabel('Thickness, $$\mathrm{z_{Ni}}$$, [m]','Interpreter','Latex');
ylabel('$$\chi^2$$','Interpreter','Latex')
title('$$\chi^2$$ as funciton of $$\mathrm{z_{Ni}}$$','Interpreter','Latex')
legend(H2,{'Minimum $$\chi^2$$'},'FontSize',17,'Interpreter','Latex')
grid on
hold off
end

% Writing out best Z:
Out(1,1) = vXL{4}(idxZ);

% Sigma
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BraggNoSigma = BraggNo(2,:);
% Finding critical angle and Bragg peaks in XRR data:
[Low,High,~,~] = FitArea(vXL,BraggNoSigma);
vXL{6} = [Low(1) High(1) Low(2) High(2)];

% Defining bounds:
[vXL,~] = Bounds(m,lb,ub,vXL);

% Defining limits of x axes for later:
xAXLIM(1) = vXL{6}(1)-vXL{6}(1)*0.2;
if isnan(vXL{6}(3))
    xAXLIM(2) = vXL{6}(2)+vXL{6}(2)*0.2;
else
    xAXLIM(2) = vXL{6}(4)+vXL{6}(4)*0.2;
end

% Only using data within DegreeLimit specifications:
[DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

% Defining parameters from bounds and the corresponding film stack:
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

% Pre-allocation physical memory:
Rsynth = nan(length(ThetaI),1,length(vXL{5}),1,1);
CHI = nan(1,length(vXL{5}),1,1);

% Calculating model refelction and chi^2 for all Sigmas:
for i = 1:length(vXL{5})
    Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,d(:,idxZ),sigma(:,i),lambda);
    CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
end

% Finding best Sigma:
[valSigma,idxSigma] = min(CHI(:));
```

```
    % Plotting:
    figure('units','normalized','OuterPosition',[0 0 1 1])
    subplot(2,2,1)
    hold on
    plot(Degree,R,'k.-')
    plot(DegreeReduced,RReduced,'ro')
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1])
    xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
           'Latex');
    ylabel('Reflectance, R','Interpreter','Latex');
    title('Measurements to be fitted','Interpreter','Latex')
    grid on
    hold off

    subplot(2,2,2)
    hold on
    plot(Degree,R,'k.-')
    plot(DegreeReduced,Rsynth(:,idxSigma),'b-','LineWidth',1)
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1])
    xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
           'Latex');
    ylabel('Reflectance, R','Interpreter','Latex');
    title('Fit','Interpreter','Latex')
    grid on
    hold off

    if length(vXL{5}) ~= 1
    subplot(2,1,2)
    hold on
    plot(vXL{5},CHI./length(RReduced),'b.-')
    H2 = plot(vXL{5}(idxSigma),valSigma/length(RReduced),'ro','LineWidth',2);
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([vXL{5}(1) vXL{5}(end) min(CHI./length(RReduced)) max(CHI./length(RReduced))]);
    xlabel('Roughness, $$\mathrm{\sigma_{Ni}}$$, [m]','Interpreter','Latex');
    ylabel('$$\chi^2$$','Interpreter','Latex');
    title('$$\chi^2$$ as funciton of $$\mathrm{\sigma_{Ni}}$$','Interpreter','Latex');
    legend(H2,{'Minimum $$\chi^2$$'},'FontSize',17,'Interpreter','Latex');
    grid on
    hold off
    end

    % Writing out best Sigma:
    Out(2,1) = vXL{5}(idxSigma);

    % Graphs:
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Computing the Rsynth with optimal parameters for all incident angles:
    [vXL,~] = Bounds(m,lb,ub,vXL2);
    [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

    % Only using data within DegreeLimit specifications:
    [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

    % Calculating model refelction and chi^2 optimal parameters:
    Rsynthbest = IMD(ThetaI,NumberOfLayers,n,d(:,idxZ),...
                     sigma(:,idxSigma),lambda);
    CHIbest = nansum((Rsynthbest-RReduced).^2./EReduced.^2);
```

```

    Z = vXL{4}(idxZ)*10^(10);
    Sigma = vXL{5}(idxSigma)*10^(10);

    legendstr1 = sprintf('%s %0.2f %s','$$\mathrm{z_{Ni}}$$ =',Z,...
                         '[$$\mathrm{\AA}$$] , ');
    legendstr2 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{Ni}}$$ =',...
                         Sigma,'[$$\mathrm{\AA}$$] , ');
    legendstr3 = sprintf('%s %0.2f %s %0.0f %s %0.2f ',...
                         '$$\chi^{2}$$ = $$\frac{',CHIbest,'}{',length(RReduced),...
                         '}$$ =',CHIbest/length(RReduced));

    figure('units','normalized','OuterPosition',[0 0 1 1])
    hold on
    plot(Degree,R,'k.-');
    plot(DegreeReduced,Rsynthbest,'b-');
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([0 4 10^(-8) 1])
    xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
           'Latex');
    ylabel('Reflectance, R','Interpreter','Latex');
    legend({'Measured data',['Fit; ',legendstr1,legendstr2,legendstr3]},...
            'Location','South','FontSize',15,'Interpreter','Latex');
    grid on
    hold off

    CHIbest = CHIbest/length(RReduced);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ML case:
elseif length(vXL) == 10

    % Gamma
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BraggNoGamma = BraggNo(1,:);
    % Finding critical angle and Bragg peaks in XRR data:
    [~,~,CritLow,CritHigh] = FitArea(vXL2,BraggNoGamma);
    vXL{10} = [CritLow CritHigh nan nan];

    % Defining bounds:
    [vXL,N] = Bounds(m,lb,ub,vXL);

    % Only using data within DegreeLimit specifications:
    [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

    % Defining parameters from bounds and the corresponding film stack:
    [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

    % Pre-allocation physical memory:
    Rsynth = nan(length(ThetaI),1,length(vXL{8}),1,1);
    CHI = nan(1,length(vXL{8}),1,1);

    % Calculating model refelction and chi^2 for all Gammas:
    for i = 1:length(vXL{8})
        Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,d(:,N(1),i),sigma(:,N(3),N(4)),...
                          lambda);
        CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
    end

```

```
% Finding best Gamma:
[valGamma,idxGamma] = min(CHI(:));

% Plotting:
figure('units','normalized','OuterPosition',[0 0 1 1])
subplot(2,2,1)
hold on
plot(Degree,R,'k.-')
plot(DegreeReduced,RReduced,'ro')
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([0 1 min(R(1:50)) 1])
xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
       'Latex');
ylabel('Reflectance, R','Interpreter','Latex');
title('Measurements to be fitted','Interpreter','Latex')
grid on
hold off

subplot(2,2,2)
hold on
plot(Degree,R,'k.-')
plot(DegreeReduced,Rsynth(:,idxGamma),'b-','LineWidth',1)
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([0 1 min(R(1:50)) 1])
xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
       'Latex');
ylabel('Reflectance, R','Interpreter','Latex');
title('Fit','Interpreter','Latex')
grid on
hold off

if length(vXL{8}) ~= 1
subplot(2,1,2)
hold on
plot(vXL{8},CHI./length(RReduced),'b.-')
H2 = plot(vXL{8}(idxGamma),valGamma/length(RReduced),'ro','LineWidth',2);
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([vXL{8}(1) vXL{8}(end) min(CHI./length(RReduced)) max(CHI./length(RReduced))]);
xlabel('Bilayer thickness ratio, $$\Gamma$$','Interpreter','Latex');
ylabel('$$\chi^2$$','Interpreter','Latex');
title('$$\chi^2$$ as funciton of $$\Gamma$$','Interpreter','Latex');
legend(H2,{'Minimum $$\chi^2$$'},'FontSize',17,'Interpreter','Latex');
grid on
hold off
end

% Writing out best Gamma:
Out(2,1) = vXL{8}(idxGamma);

% D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BraggNoD = BraggNo(1,:);
% Finding critical angle and Bragg peaks in XRR data:
[Low,High,~,~] = FitArea(vXL2,BraggNoD);
vXL{10} = [Low(1) High(1) Low(2) High(2)];

% Defining bounds:
[vXL,~] = Bounds(m,lb,ub,vXL);

```

```
351     % Defining limits of x axes:
352     xAXLIM(1) = vXL{10}(1)-vXL{10}(1)*0.2;
353     if isnan(vXL{10}(3))
354         xAXLIM(2) = vXL{10}(2)+vXL{10}(2)*0.2;
355     else
356         xAXLIM(2) = vXL{10}(4)+vXL{10}(4)*0.2;
357     end
358
359     % Only using data within DegreeLimit specifications:
360     [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);
361
362     if ~isnan(vXL{10}(3))
363     [~,tmp3] = max(diff(DegreeReduced));
364     tmp4 = tmp3+1;
365     else
366         tmp3 = length(DegreeReduced);
367         tmp4 = 1;
368     end
369
370     % Defining parameters from bounds and the corresponding film stack:
371     [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);
372
373     % Pre-allocation physical memory:
374     Rsynth = nan(length(ThetaI),1,length(vXL{7}),1,1);
375     CHI = nan(1,length(vXL{7}),1,1);
376
377     % Calculating model refelction and chi^2 for all D:
378     for i = 1:length(vXL{7})
379         Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,d(:,i,idxGamma),sigma(:,N(3),N(4)),...
380                           lambda);
381         CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
382     end
383
384     % Finding best D:
385     [valDspacing,idxDspacing] = min(CHI(:));
386
387     % Plotting:
388     figure('units','normalized','OuterPosition',[0 0 1 1])
389     subplot(2,2,1)
390     hold on
391     plot(Degree,R,'k.-')
392     plot(DegreeReduced,RReduced,'ro')
393     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
394     axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1]);
395     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
396            'Latex');
397     ylabel('Reflectance, R','Interpreter','Latex');
398     title('Measurements to be fitted','Interpreter','Latex');
399     grid on
400     hold off
401
402     subplot(2,2,2)
403     hold on
404     plot(Degree,R,'k.-')
405     plot(DegreeReduced(1:tmp3),Rsynth(1:tmp3,idxDspacing),'b-','LineWidth',1)
406     plot(DegreeReduced(tmp4:end),Rsynth(tmp4:end,idxDspacing),'b-','LineWidth',1)
407     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
408     axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1])
409     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
```

```
       'Latex');
ylabel('Reflectance, R','Interpreter','Latex');
title('Fit','Interpreter','Latex')
grid on
hold off

if length(vXL{7}) ~= 1
subplot(2,1,2)
hold on
plot(vXL{7},CHI./length(RReduced),'b.-')
H2 = plot(vXL{7}(idxDspacing),valDspacing/length(RReduced),'ro','LineWidth',2);
set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
axis([vXL{7}(1)...
      vXL{7}(end) ...
      min(CHI./length(RReduced))-min(CHI./length(RReduced))*0.1 ...
      max(CHI./length(RReduced))+max(CHI./length(RReduced))*0.1])
xlabel('D-spacing, D, $$[\mathrm{m}]$$','Interpreter','Latex');
ylabel('$$\chi^2$$','Interpreter','Latex')
title('$$\chi^2$$ as funciton of D','Interpreter','Latex')
legend(H2,{'Minimum $$\chi^2$$'},'FontSize',17,'Interpreter','Latex')
grid on
hold off
end

% Writing out best D:
Out(1,1) = vXL{7}(idxDspacing);

% Sigma1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BraggNoSigma1 = BraggNo(2,:);
% Finding critical angle and Bragg peaks in XRR data:
[Low,High,~,~] = FitArea(vXL2,BraggNoSigma1);
vXL{10} = [Low(1) High(1) Low(2) High(2)];

% Defining bounds:
[vXL,~] = Bounds(m,lb,ub,vXL);

% Defining limits of x axes:
xAXLIM(1) = vXL{10}(1)-vXL{10}(1)*0.2;
if isnan(vXL{10}(3))
    xAXLIM(2) = vXL{10}(2)+vXL{10}(2)*0.2;
else
    xAXLIM(2) = vXL{10}(4)+vXL{10}(4)*0.2;
end

% Only using data within DegreeLimit specifications:
[DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

if ~isnan(vXL{10}(3))
[~,tmp3] = max(diff(DegreeReduced));
tmp4 = tmp3+1;
else
    tmp3 = length(DegreeReduced);
    tmp4 = 1;
end

% Defining parameters from bounds and the corresponding film stack:
[ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

```

```
469     % Pre-allocation physical memory:
470     Rsynth = nan(length(ThetaI),1,length(vXL{4}),1,1);
471     CHI = nan(1,length(vXL{4}),1,1);
472
473     % Calculating model refelction and chi^2 for all Sigma1:
474     for i = 1:length(vXL{4})
475         Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,...
476                           d(:,idxDspacing,idxGamma),sigma(:,i,N(4)),lambda);
477         CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
478     end
479
480     % Finding best Sigma1:
481     [valSigma1,idxSigma1] = min(CHI(:));
482
483     % Plotting:
484     figure('units','normalized','OuterPosition',[0 0 1 1])
485     subplot(2,2,1)
486     hold on
487     plot(Degree,R,'k.-')
488     plot(DegreeReduced,RReduced,'ro')
489     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
490     axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1]);
491     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
492            'Latex');
493     ylabel('Reflectance, R','Interpreter','Latex');
494     title('Measurements to be fitted','Interpreter','Latex');
495     grid on
496     hold off
497
498     subplot(2,2,2)
499     hold on
500     plot(Degree,R,'k.-')
501     plot(DegreeReduced(1:tmp3),Rsynth(1:tmp3,idxSigma1),'b-','LineWidth',1);
502     plot(DegreeReduced(tmp4:end),Rsynth(tmp4:end,idxSigma1),'b-','LineWidth',1);
503     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
504     axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1]);
505     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
506            'Latex');
507     ylabel('Reflectance, R','Interpreter','Latex');
508     title('Fit','Interpreter','Latex');
509     grid on
510     hold off
511
512     if length(vXL{4}) ~= 1
513     subplot(2,1,2)
514     hold on
515     plot(vXL{4},CHI./length(RReduced),'b.-')
516     H2 = plot(vXL{4}(idxSigma1),valSigma1/length(RReduced),'ro','LineWidth',2);
517     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
518     axis([vXL{4}(1) ...
519           vXL{4}(end) ...
520           min(CHI./length(RReduced))-min(CHI./length(RReduced))*0.1 ...
521           max(CHI./length(RReduced))+max(CHI./length(RReduced))*0.1])
522     xlabel('Roughness, $$\mathrm{\sigma_{B_{4}C}}$$, $$[\mathrm{m}]$$','Interpreter',...
523            'Latex');
524     ylabel('$$\chi^2$$','Interpreter','Latex');
525     title(['$$\mathrm{\chi^2}$$ as funciton of ','$$\mathrm{\sigma_{B_{4}C}}$$'],...
526            'Interpreter','Latex');
527     legend(H2,{'Minimum $$\mathrm{\chi^2}$$'},'FontSize',17,'Interpreter','Latex');
```

```
    grid on
    hold off
    end

    % Writing out best Sigma1:
    Out(3,1) = vXL{4}(idxSigma1);

    % Sigma2
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BraggNoSigma2 = BraggNo(3,:);
    % Finding critical angle and Bragg peaks in XRR data:
    [Low,High,~,~] = FitArea(vXL2,BraggNoSigma2);
    vXL{10} = [Low(1) High(1) Low(2) High(2)];

    % Defining bounds:
    [vXL,~] = Bounds(m,lb,ub,vXL);

    % Defining limits of x axes:
    xAXLIM(1) = vXL{10}(1)-vXL{10}(1)*0.2;
    if isnan(vXL{10}(3))
        xAXLIM(2) = vXL{10}(2)+vXL{10}(2)*0.2;
    else
        xAXLIM(2) = vXL{10}(4)+vXL{10}(4)*0.2;
    end

    % Only using data within DegreeLimit specifications:
    [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

    if ~isnan(vXL{10}(3))
    [~,tmp3] = max(diff(DegreeReduced));
    tmp4 = tmp3+1;
    else
        tmp3 = length(DegreeReduced);
        tmp4 = 1;
    end

    % Defining parameters from bounds and the corresponding film stack:
    [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

    % Pre-allocation physical memory:
    Rsynth = nan(length(ThetaI),1,length(vXL{6}),1,1);
    CHI = nan(1,length(vXL{6}),1,1);


    % Calculating model refelction and chi^2 for all Gammas:
    for i = 1:length(vXL{6})
        Rsynth(:,i) = IMD(ThetaI,NumberOfLayers,n,d(:,idxDspacing,idxGamma),...
                          sigma(:,idxSigma1,i),lambda);
        CHI(i) = nansum((Rsynth(:,i)-RReduced).^2./EReduced.^2);
    end

    % Finding best Sigma2:
    [valSigma2,idxSigma2] = min(CHI(:));

    % Plotting:
    figure('units','normalized','OuterPosition',[0 0 1 1])
    subplot(2,2,1)
    hold on
    plot(Degree,R,'k.-')
```

```
    plot(DegreeReduced,RReduced,'ro')
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1]);
    xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
           'Latex');
    ylabel('Reflectance, R','Interpreter','Latex');
    title('Measurements to be fitted','Interpreter','Latex');
    grid on
    hold off

    subplot(2,2,2)
    hold on
    plot(Degree,R,'k.-')
    plot(DegreeReduced(1:tmp3),Rsynth(1:tmp3,idxSigma2),'b-','LineWidth',1);
    plot(DegreeReduced(tmp4:end),Rsynth(tmp4:end,idxSigma2),'b-','LineWidth',1);
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([xAXLIM(1) xAXLIM(2) 10^(-6) 1]);
    xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
           'Latex');
    ylabel('Reflectance, R','Interpreter','Latex');
    title('Fit','Interpreter','Latex');
    grid on
    hold off

    if length(vXL{6}) ~= 1
    subplot(2,1,2)
    hold on
    plot(vXL{6},CHI./length(RReduced),'b.-')
    H2 = plot(vXL{6}(idxSigma2),valSigma2/length(RReduced),'ro','LineWidth',2);
    set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
    axis([vXL{6}(1) ...
          vXL{6}(end) ...
          min(CHI./length(RReduced))-min(CHI./length(RReduced))*0.1 ...
          max(CHI./length(RReduced))+max(CHI./length(RReduced))*0.1]);
    xlabel('Roughness, $$\mathrm{\sigma_{Ni}}$$, $$[\mathrm{m}]$$','Interpreter',...
           'Latex');
    ylabel('$$\chi^2$$','Interpreter','Latex');
    title('$$\mathrm{\chi^2}$$ as funciton of $$\mathrm{\sigma_{Ni}}$$','Interpreter',...
          'Latex');
    legend(H2,{'Minimum $$\mathrm{\chi^2}$$'},'FontSize',17,'Interpreter','Latex');
    grid on
    hold off
    end

    % Writing out best Sigma2:
    Out(4,1) = vXL{6}(idxSigma2);

    % Graphs:
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Computing the Rsynth with optimal parameters for all incident angles:
    [vXL,~] = Bounds(m,lb,ub,vXL2);
    [ThetaI,NumberOfLayers,n,d,sigma,lambda] = FilmDefinitions(vXL);

    % Only using data within DegreeLimit specifications:
    [DegreeReduced,RReduced,EReduced] = DataReducing(Degree,R,E,vXL);

    % Calculating model refelction and chi^2 optimal parameters:
    Rsynthbest = IMD(ThetaI,NumberOfLayers,n,d(:,idxDspacing,idxGamma),...
                     sigma(:,idxSigma1,idxSigma2),lambda);
```

```
646     CHIbest = nansum((Rsynthbest-RReduced).^2./EReduced.^2);
647
648     D = vXL{7}(idxDspacing)*10^(10);
649     Gamma = vXL{8}(idxGamma);
650     Sigma1 = vXL{4}(idxSigma1)*10^(10);
651     Sigma2 = vXL{6}(idxSigma2)*10^(10);
652
653     legendstr1 = sprintf('%s %0.2f %s','D =',D,'[$$\mathrm{\AA}$$] , ');
654     legendstr2 = sprintf('%s %0.4f %s','$$\Gamma$$ =',Gamma,', ');
655     legendstr3 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{B_{4}C}}$$ =',...
656                          Sigma1,'[$$\mathrm{\AA}$$] , ');
657     legendstr4 = sprintf('%s %0.2f %s','$$\mathrm{\sigma_{Ni}}$$ =',...
658                          Sigma2,'[$$\mathrm{\AA}$$] , ');
659     legendstr5 = sprintf('%s %0.2f %s %0.0f %s %0.2f ',...
660                          '$$\chi^{2}$$ = $$\frac{',CHIbest,...
661                          '}{',length(RReduced),...
662                          '}$$ = ',CHIbest/length(RReduced));
663
664     figure('units','normalized','OuterPosition',[0 0 1 1])
665     hold on
666     plot(Degree,R,'k.-');
667     plot(DegreeReduced,Rsynthbest,'b-');
668     set(gca,'YScale','log','FontWeight','Normal','FontSize',20,'FontName','Times');
669     axis([0 4 10^(-8) 1])
670     xlabel('Gracing angle, $$\theta_{i}$$, $$[\mathrm{^{\circ}}]$$','Interpreter',...
671            'Latex');
672     ylabel('Reflectance, R','Interpreter','Latex');
673     legend({'Measured data',...
674            ['Fit; ',legendstr1,legendstr2,legendstr3,legendstr4,legendstr5]},...
675            'Location','South','FontSize',17,'Interpreter','Latex');
676     grid on
677     hold off
678
679     % Writing out best chi:
680     CHIbest = CHIbest/length(RReduced);
681 end
682 end
```

B.14 DDR code

```
function [CoatingRate, DDR] = DynamicDepositionRate(RateStepsPerSec,z,P,Aimedz,Pressure)
%% DinamicDepositionRate is a function that calculates the dinamic deposition rate, DDR,
%  on the basis of coating rate and the corresponding layer thickness, z. If more than
%  three input parameters are provided the produced DDRs are futhermore used to
%  determine new coating rates at aimed thicknesses, i.e. Aimedz.
%
% INPUT:
% RateStepsPerSec: Vector of coating rates at which layers are produced.
% z:               Vector of produced thicknesses corresponding to the coating rates in
%                  RateStepsPerSec.
% P:               Pressure the coationgs with thickness z are produced at.
% Aimedz:          Vector of thicknesses that are to be produced in future coatings.
% Pressure:        The pressure at which the Aimedz coatings should be preduced at.
%
% OUTPUT:
% The output of this function is a text file with a table containing pressure, z,
% coating rate and DDR. The parameters in the first part of the table relates to
% measurements of coatings and the the corresponding DDR. The second part of the table
% contains pressure, aimed thicknesses, coating rates and aimed DDRs of furture
% coatings. Coating rates of aimed thicknesses are calculated on basis of the first part
% of the table.
%
% CoatingRate: Coating rates of the thicknesses specified in Aimedz.
% DDR:         Dinamic deposition rates calculated from RateStepsPerSec and z.
%
% Author:      Christoffer Raad (christoffer.raad@gmail.com)
% Date:        28/04/2017
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EXAMPLE:
% Rate   = [ 2000  2000  2000   ]';
% z50    = [ 47.00 43.34 43.00  ]'*0.1; % [nm]
% P50    = [ 2.5   4.0   5.0    ]';
% Aimedz = [1.4 2.8 5 10 20 30]';
% Pressure = 5;
% [CoatingRate, DDR] = DynamicDepositionRate(Rate,z50,P50,Aimedz,Pressure);
%
% Pressure   z           Coating rate     |  DDR
% [mTorr]        [nm]        [steps/s]    |  [nm*m/min]
% ========================================================
% 2.5            4.7         2000.0       |  2.5199
% 4.0            4.3         2000.0       |  2.3236
% 5.0            4.3         2000.0       |  2.3054
% ========================================================
%
% From the values abowe, the aimed thicknesses have
% coating rates as below
%
% Pressure   Aimed z     Coating rate    |   Aimed DDR
% [mTorr]        [nm]    [steps/s]       |   [nm*m/min]
% ========================================================
% 5.0            1.4         6714.3      |   2.5199
% 5.0            2.8         3357.1      |   2.5199
% 5.0            5.0         1720.0      |   2.3054
% 5.0            10.0        860.0       |   2.3054
```

```
% 5.0            20.0        430.0       |   2.3054
% 5.0            30.0        286.7       |   2.3054
% =======================================================

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculate dinamic deposition rate and corresponding steps per sec rates:

% Defining chamber dimentions:
Osteps = 668000;
r = 0.475;
Ometer = pi*2*r;

% Calculates dinamic deposition rates:
% DDR(:,1) -> high z. DDR(:,end) -> low z.
DDR = (z.*RateStepsPerSec.*60)./((Osteps/Ometer));
CoatingRate = [];


figure
hold on
plot(P,DDR,'ko')
Hline = refline([0 mean(DDR)]);
Hline.Color = 'r';
Hline.XData = [floor(min(P))-1 ceil(max(P))+1];
axis([floor(min(P))-1 ceil(max(P))+1 floor(min(DDR))-1 ceil(max(DDR))+1]);
xlabel('Thickness, z [nm]');
ylabel('Dynamic deposition rate, DDR [nm*m/min]')
title('Dynamic deposition rates')
legend(['Dynamic deposition rates [nm*m/min]',strcat('Avg. DDR:',{' '},...
    num2str(mean(DDR)))])
grid on
hold off

if nargin > 3

    % Finding measured samples that corrospond with whised pressure:
    idx = find(P-Pressure==0);
    if isempty(idx)
        [~,idx] = min(abs(P-Pressure));
        Pressure = P(idx);
    end

    tmpDDR = DDR(idx);
    tmpz = z(idx);
    [tmpz2,sortidx] = sort(tmpz);
    tmpDDR2 = tmpDDR(sortidx);

    [~,ind] = histc(Aimedz,vertcat(0,tmpz2,inf));

    % Pre-allocating physical memory:
    CoatingRate = nan(length(Aimedz),1);
    AimedDDR = nan(length(Aimedz),1);

    for i = 1:length(Aimedz)
        if ind(i) == 1
            CoatingRate(i,1) = tmpDDR2(1)'.*Osteps./(Aimedz(i).*Ometer*60);
            AimedDDR(i,1) = tmpDDR2(1);
        elseif ind(i) == length(tmpz2)+1
            CoatingRate(i,1) = tmpDDR2(end)'.*Osteps./(Aimedz(i).*Ometer*60);
```

```
            AimedDDR(i,1) = tmpDDR2(end);
        else
            CoatingRate(i,1) = interp1(tmpz2(ind(i)-1:ind(i)),...
                tmpDDR2(ind(i)-1:ind(i)),...
                Aimedz(i)).*Osteps./(Aimedz(i).*Ometer*60);
            AimedDDR(i,1) = interp1(tmpz2(ind(i)-1:ind(i)),tmpDDR2(ind(i)-1:ind(i)),...
                Aimedz(i));
        end
    end

    %% Printing results in command window and to text file:
    tmp = [length(z) length(Aimedz)];
    Pnew = ones(max(tmp),1).*Pressure;


    FID = fopen(strcat('CoatingRate','.txt'), 'w');
    fprintf(FID,'%s\t %s\t\t\t %s\t %s\t %s \n','Pressure','z','Coating rate','|',...
        'DDR');
    fprintf(FID,'%s\t\t %s\t\t %s\t\t %s\t %s \n','[mTorr]','[nm]','[steps/s]','|',...
        '[nm*m/min]');
    fprintf(FID,'%s \n','=======================================================');
    for i = 1:length(z)
        fprintf(FID,'%0.1f\t\t\t %.1f\t\t %.1f\t\t\t %s\t %.4f \n',P(i),z(i),...
            RateStepsPerSec(i),'|',DDR(i));
    end
    fprintf(FID,'%s \n \n','=======================================================');
    fprintf(FID,'%s \n%s \n \n','From the values abowe, the aimed thicknesses have',...
        'coating rates as below');
    fprintf(FID,'%s\t %s\t %s\t %s\t %s \n','Pressure','Aimed z','Coating rate','|',...
        'Aimed DDR');
    fprintf(FID,'%s\t\t %s\t\t %s\t\t %s\t %s \n','[mTorr]','[nm]','[steps/s]','|',...
        '[nm*m/min]');
    fprintf(FID,'%s \n','=======================================================');
    for i = 1:length(Aimedz)
        fprintf(FID,'%.1f\t\t\t %.1f\t\t %.1f\t\t\t %s\t %.4f \n',Pnew(i),Aimedz(i),...
            CoatingRate(i),'|',AimedDDR(i));
    end
    fprintf(FID,'%s \n \n','=======================================================');
    fclose(FID);

    type(strcat('CoatingRate','.txt'))
end
end
```

DTU Space
Technical University of Denmark

Elektrovej, building 328
2800 Kgs. Lyngby

www.space.dtu.dk