



UNIVERSITÉ DE MONTPELLIER
FACULTÉ DES SCIENCES

COMMISSARIAT A L'ENERGIE ATOMIQUE ET
AUX ENERGIES ALTERNATIVES

**Report of the Alternance program for the Masters
second year in Computational Physics**

Author :

Nischal Dhungana

Under the supervision of :

Guillaume Freychet

June 2024

Acknowledgements

I would like to express my sincere gratitude to Dr. Guillaume Freychet, who has been an invaluable mentor and guide throughout this past year. His patience, availability, and unwavering support have been instrumental in my professional development.

I would also like to extend my thanks to the entire team at LPMS for creating such an excellent working environment. Their camaraderie and support made this a truly enriching experience.

Finally, a special thanks to Prof. Brahim Guizal, Director of my Master's program. His guidance won't go unappreciated.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 2 | Context | 7 |
| 2.1 | Host organisation | 7 |
| 2.1.1 | History | 8 |
| 2.1.2 | Sectors of activity | 8 |
| 2.1.3 | Future and orientation strategy | 9 |
| 2.2 | Project description | 10 |
| 2.3 | Objectives | 11 |
| 3 | CD-SAXS | 13 |
| 3.1 | Introduction | 13 |
| 3.2 | Experimental Setup | 15 |
| 3.3 | Scattering Model | 18 |
| 3.4 | Fitting Algorithm | 19 |
| 3.4.1 | Covariance Matrix Adaptation Evolution Strategy (CMAES) | 20 |
| 3.4.2 | Uncertainty estimation by Monte Carlo Markov Chain(MCMC) method | 22 |
| 3.4.3 | Overall view of the CD-SAXS inverse problem resolution | 23 |
| 4 | My contribution | 26 |
| 4.1 | Introduction | 26 |
| 4.2 | Design | 26 |
| 4.2.1 | Components | 26 |
| 4.2.2 | Relationships between components | 28 |
| 4.2.3 | Why this design? | 29 |
| 4.3 | Brief Overview of Photoresist Lithography | 30 |
| 4.4 | Simulation Models | 31 |
| 4.4.1 | Stacked Trapezoid Model | 32 |
| 4.4.2 | Rounded corners model | 33 |
| 4.4.3 | Overlay model | 34 |
| 4.5 | Implementation of MCMC to Measure Uncertainty | 35 |
| 4.6 | Parallelization and GPU Acceleration | 37 |
| 4.6.1 | Overview of the concept of parallelisation | 38 |
| 4.6.2 | Parallelization Strategies | 38 |
| 4.7 | On-the-Fly Uncertainty Estimation | 42 |

5 Future Prospects **45**

6 Conclusion **46**

1 Introduction

The continuous miniaturization of microelectronic components, driven by Moore's Law, has led to a significant reduction in transistor size and increased chip complexity. This rapid advancement has presented new challenges in the field of metrology, the science of measurement. Existing metrology techniques, such as Optical Critical Dimension (OCD) and Critical Dimension Scanning Electron Microscopy (CDSEM), are reaching their limits in terms of resolution and accuracy as feature sizes shrink to the nanometer scale.

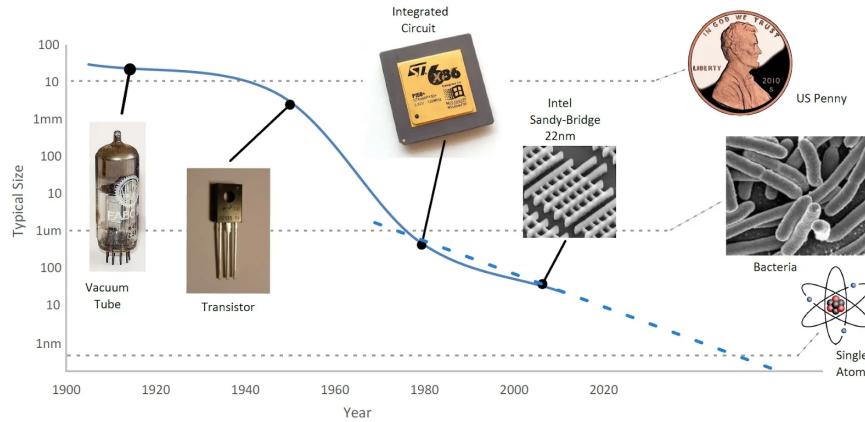


Figure 2: Evolution of microelectronics and illustration of the need for advanced metrology techniques [1].

To address these challenges, a new metrology technique called Critical Dimension Small Angle X-ray Scattering (CD-SAXS) is being developed. In the context of microelectronics, CD refers to characterizing feature length of the nanostructures. CDs are the measurements that directly affect the performance and function of the devices. For example in the figure 3, the CD is the width of the rectangular nanostructures.

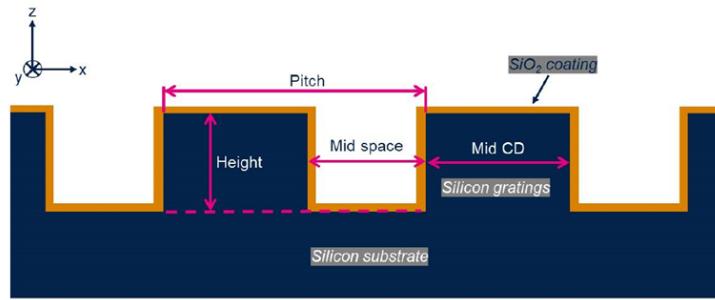


Figure 3: Illustration of a CD measurement

CD-SAXS utilizes short-wavelength X-rays ($\lambda \approx 0.05 - 5\text{nm}$), to probe the internal structure of materials, providing high-resolution measurements of CD with greater accuracy

than conventional methods. CEA-Leti, a leading research institute in microelectronics, is actively involved in the development of CD-SAXS technology.

This work-study project focused on the development of a coherent software for the fit and analysis of CD-SAXS data. The software aims to streamline the data processing workflow and enhance the accuracy of CD measurements. The project involved a comprehensive understanding of CD-SAXS theory, data collection procedures, and fitting algorithms.

The report begins with an overview of the context of the project, highlighting the evolution of microelectronics and the need for advanced metrology techniques. It then delves into the CD-SAXS technique, explaining the principles, data collection, fitting, and analysis. Then the subsequent section describes the software development process, outlining the software's functionalities and design. Finally, the report concludes with a summary of the project's achievements and outlines potential future directions.

2 Context

2.1 Host organisation

The French Alternative Energies and Atomic Energy Commission (CEA) stands as a cornerstone of the nation research landscape. Its multifaceted expertise encompasses a broad spectrum of fields, including nuclear energy, renewable energy, technological research for industry, material sciences, health and life sciences, and defense and security. The CEA network of research centers spans across France, each with its unique specializations and areas of excellence. Among these, CEA Grenoble holds a prominent position, where I have the privilege of pursuing my work-study program. I am part of the laboratory of electrical technology and information (Leti) Institute, a research center dedicated to microelectronics and nanotechnologies. More specifically, I was with “Materials and Structures Properties Laboratory” (MSPL) which is under “Technology Platforms Department” (TPFD), one of six different departments of Leti.



Figure 4: CEA Grenoble Campus

2.1.1 History

The French atomic energy commission, CEA, was born in 1945 after World War II. Its mission was to develop nuclear expertise for France. Pioneering scientists like Frédéric Joliot-Curie and Francis Perrin led the way in building research reactors and nuclear power plants.

The CEA did not stop at just nuclear energy. In the 1960s, they began to diversify into new areas like renewable energy, micro and nanotechnologies, defense, and healthcare. This diversification led to the creation of specialized research centers, including the future innovation hub, CEA Grenoble.

It was founded in 1956 by Nobel laureate physicist Louis Néel. He saw the scientific potential of the Grenoble region and his vision proved to be true. The center grew rapidly in the following decades, attracting talent and investment from around the world.

It came to be known as France "atomic capital" due to its research reactors. However, their influence went far beyond nuclear. They developed their first integrated circuit in 1965, launching their journey into micro and nanotechnologies. They also played a key role in creating Minatec, the first European hub for excellence in this field. In addition, they became a leader in renewable energy research with the Institut national de l'énergie solaire (Ines). European synchrotron radiation facility (ESRF) was also established in Grenoble in 1988, further solidifying the city reputation as a scientific hub.

Today, it is a research powerhouse with over 2,500 researchers and technicians. Their campus houses specialized institutes in various fields, from healthcare to digital technologies. It is also the headquarters for CEA Tech, the technological branch of the CEA with over 4,500 researchers across France.

2.1.2 Sectors of activity

CEA Grenoble plays a pivotal role in the nation economic and technological advancement through its groundbreaking research and innovations across diverse fields.

- **Energy and Sustainability:** Supporting current and future nuclear power, exploring solar, hydrogen for carbon neutrality (2050), researching SMRs (Small modular reactors) and thermonuclear fusion.
- **Digital Technologies:** Contributing through the SPIN program for spintronic (frugal, agile, sustainable computing) aligned with France 2030 plan.
- **Healthcare:** Distinguished research in biology and biotechnology for health, addressing current and future challenges. (e.g., Laboratoire de biologie et biotechnologie pour la santé)



Figure 5: CEA Grenoble campus before and after

- **Defense:** Traditionally significant role, developing cutting-edge technologies for national security and defense (less documented for Grenoble).

CEA Liten for example also serves as an innovation hub for new energy technologies and nanomaterials, emphasizing energy diversification and renewable energy integration. Their research encompasses solar photovoltaics, energy storage, and transportation (hydrogen and fuel cells).

2.1.3 Future and orientation strategy

CEA stands as a powerhouse for innovation in France. Spanning energy, healthcare, defense, and digital technologies, the CEA pushes boundaries by collaborating with universities and industry on ambitious R&D projects.

Their focus is clear: develop transformative solutions for global challenges. This includes renewable energy sources, innovative healthcare systems, advanced defense solutions, and disruptive digital technologies. Sustainability is paramount, with research prioritizing energy efficiency and minimizing environmental impact.

The CEA partners with leading institutions to accelerate technology transfer and create

open innovation ecosystems. These partnerships combine expertise, resources, and networks, leading to breakthrough technologies with significant economic and social value.

CEA Grenoble tries to prosper an innovative spirit. They focus on technologies like AI, advanced materials, nanotechnologies, and quantum technologies. Their research aims to improve industries, from developing next-generation batteries to creating innovative medical devices and advancing digital technologies.

The challenges they tackle are various: climate change, the energy transition, emerging diseases, cybersecurity, and technological sovereignty. Their research is guided by a long-term vision, anticipating future challenges and preparing for them through innovation.

The CEA aims to shape a sustainable future. They innovate in clean energy sources, advanced healthcare, robust national security, and a cutting-edge digital landscape. By working collaboratively and addressing global challenges head-on, the CEA tries to position itself as a major contributor in the global scientific and technological landscape.

2.2 Project description

CEA Grenoble, a frontrunner in material metrology and characterization, recognizes the potential of CD-SAXS for analyzing nanostructures. To this end, CEA has actively invested in CD-SAXS development for several years.

During my work-study program, I contributed to this initiative by developing a Python application for data fitting and analysis obtained through CD-SAXS. This technique relies on an inverse algorithm that translates scattering intensity data into a relevant real-space structure. The algorithm simulates the experiment using a model and iteratively compares the simulated data with the experimental data until a good fit is achieved. A robust codebase is essential for efficient CD-SAXS data processing.

Previously, a collection of functions developed at CEA by former PhD students and at Lawrence Berkeley National Laboratory and Brookhaven National Laboratory served as the foundation for CD-SAXS analysis. However, it lacked the coherence of a well-structured application. The code for various data simulation models was similarly disorganized. My primary task was to develop a user-friendly Python application that integrates these functions, streamlining data fitting and result analysis.

Furthermore, the existing code suffered from slow execution speeds due to a lack of optimization. To address this, I implemented optimizations and parallelization techniques, significantly improving the code efficiency.

2.3 Objectives

The initial aim of this project was commendable - to improve the existing CD-SAXS data analysis codebase. However, as we delved deeper, the project objectives evolved into a series of well-defined, targeted enhancements. This iterative approach ensured that our efforts addressed the specific needs of researchers at CEA Grenoble.

Here is a breakdown of the key objectives that emerged:

- **Unleashing Computational Power:**

- **Vectorization for Server Optimization:** Standard code often struggles to fully leverage the parallel processing capabilities of modern servers. We identified the need to vectorize the code, allowing it to efficiently utilize the computational power available on CEA powerful servers. This significantly boosted the applications overall processing speed.
- **GPU Acceleration - Pushing the Limits:** Recognizing the ever-increasing capabilities of Graphics Processing Units (GPUs), we explored the potential of integrating GPU acceleration into the code. This would potentially unlock even faster performance, enabling us to tackle increasingly complex datasets.

- **User-Centric Design:**

- **Building a Coherent Application:** The existing code, while functional, lacked the user-friendliness and intuitiveness necessary. We prioritized creating a well-structured, modular application. This would not only simplify data fitting and analysis but also make future feature additions and maintenance easier and more efficient.
- **Addition of different simulation models in the same codebase:** The original codebase had several independent simulation models: a model composed of stacked trapezoid, an overlay model and a rounded trapezoid model. We aimed to integrate these three models into a single, cohesive codebase. This consolidation would streamline the process of selecting and applying different models, enhancing the overall user experience.

- **Quantifying Uncertainty - Confidence in Results:**

- **MCMC for Uncertainty Estimation:** A critical component missing from the original code was the ability to estimate the uncertainty associated with the fitted parameters. We addressed this by implementing a Markov Chain Monte Carlo

(MCMC) algorithm. This powerful technique allowed us to generate a statistically representative set of solutions, enabling us to accurately estimate the level of uncertainty in the fitted parameters.

- **Real-Time Uncertainty Estimation - Saving Valuable Time:**

- **On-the-Fly Uncertainty:** Taking the concept of uncertainty estimation further, we aimed to integrate this feature into the data acquisition process itself, specifically during synchrotron measurements. This would allow researchers to estimate the uncertainty in real-time. By setting a desired uncertainty threshold, the system could potentially stop the experiment once this level of certainty is achieved. This innovation has the potential to save valuable synchrotron beamtime, a precious resource for researchers.

This refined set of objectives ensured that the project delivered valuable enhancements tailored to the needs of CEA researchers. The project not only improved the code performance but also transformed it into a user-friendly and powerful tool for analyzing CD-SAXS data with greater confidence.

3 CD-SAXS

3.1 Introduction

Miniaturizing transistors, the building blocks of integrated circuits, is getting tougher for the semiconductor industry. Shrinking their size and spacing (pitch) brings not only manufacturing hurdles but also a metrology problem. Precisely measuring these features during production is crucial for high-quality chips. Existing in-line metrology techniques, like optical critical-dimension (OCD) scatterometry and critical-dimension scanning electron microscopy (CD-SEM), are nearing their limits [2, 3]. OCD struggles with limitations inherent to light and the ever-shrinking features. CD-SEM offers valuable insights because it provides an image of the sample but is restricted by the sampling area and also its resolution. To overcome these obstacles, the industry is exploring X-ray-based metrology. X-rays have much shorter wavelengths than the features being measured, allowing for more precise analysis. Additionally, they are sensitive to variations in composition, providing a richer data set.

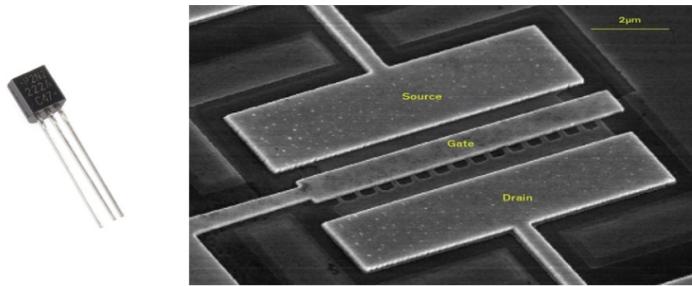


Figure 6: A classic transistor with three terminals. (On the right, an electron microscope image of a single transistor on a microchip.)[4]

CD-SAXS, a promising technique for nanostructure electronics, uses a transmission geometry, sending the beam through the sample and the 750 micrometer-thick silicon wafer. The x-ray spot size varying between 10-1000 μm , it enable the measurement of small patterned areas. Studies have shown CD-SAXS effectiveness in characterizing the shape and spacing of nanometer-sized patterns [6].

CD-SAXS utilizes variable-angle transmission scattering. By rotating the sample, it can probe the vertical profile of the nanostructures. This allows for reconstructing their shape and composition in two or even three dimensions. We can think of it as a diffraction experiment for single crystals, but instead of a crystal, the periodic array of nanostructures acts like one. This technique excels at reconstructing intricate shapes smaller than 15 nm and with spacing around 30 nm, dimensions crucial for the semiconductor industry. [7]

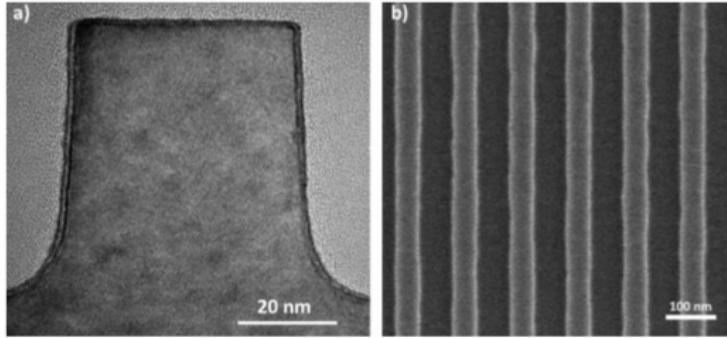


Figure 7: Examples of nanostructures that is typically being studied for characterisation. a is cross section of a line in b. [18]

While CD-SAXS was demonstrated very useful, the main limitation is inherent to its geometry. The beam needs to go through the 750 micrometer-thick silicon wafer. This is critical for x-ray energy below 17 keV because a major part of the beam is absorbed. The first CD-SAXS studies were conducted in synchrotron where the energy of the x-rays can be tuned and where the brilliance is the highest. For laboratory sources, it is currently still an issue since there is no source with enough photons to enable efficient laboratory measurements. This is a major constrain for the spread of CD-SAXS where a tool in the factory is required to have a direct control during the production.

A possibility to go around the absorption limitation is to perform grazing-incidence experiment where the beam is now reflected at the surface of the sample. This technique, called GISAXS become more relevant. It uses X-rays near the critical angle of the probed film resulting in a large sampling area and statistically significant data. This large area allows for faster measurements, enabling *in situ* kinetic studies. It has the same resolution of CD-SAXS but enable to examine nanostructures across large areas, i.e. the beam footprint on the sample surface being the beam size divided by the sinus of the incident angle of the beam on the sample. For GISAXS, the incident angle is usually around 0.3 deg, making a 10 μm beam footprint equal to 2 mm for semiconductors.

Early research on X-ray characterization of patterned nanostructures used reflection methods like X-ray diffraction (XRD) and grazing-incidence small-angle X-ray scattering (GISAXS). These techniques demonstrated X-rays sensitivity to features shape and spacing. Furthermore, X-rays can probe buried features due to their sensitivity to electron density contrast. For instance, a GaInAs/InP multilayer was studied with high-resolution XRD, revealing the sensitivity to both the grating and strain between layers [5].

While probing large areas is interesting for some application such as the study of polymer for example, it is a limit for the semiconductor metrology. Indeed, the patterned areas are

only few micrometers wide except for some specific applications like memory arrays (Hofmann et al., 2009; Scholze et al., 2011). Logic devices require smaller probing areas due to test structure size and the complexity of the multicomponent, 3D nanostructures. But CD-SAXS has no such limitation thus it has remerged as a technique that still needs to be developed but is a promising one.[8]

3.2 Experimental Setup

During this project, I had the opportunity to work and observe how CD-SAXS measurements are performed at the European Synchrotron Radiation Facility (ESRF) in Grenoble, France. The ESRF is a world-renowned research facility that provides intense X-ray beams for a wide range of scientific experiments. Specifically, I worked with the BM02 beamline, which is dedicated to X-ray scattering and diffraction experiments.



Figure 8: The European Synchrotron Radiation Facility (ESRF) in Grenoble, France. (credit: ESRF/Jocelyn Chavy)

The X-ray source obtained from the synchrotron is monochromatized and then focused using various optical elements. The beam is then directed towards the sample. The sample is mounted on a device that allows for rotation and translation in all three dimensions. Once the alignment is complete, the sample is rotated along the vertical axis (along the direction of the line), from -60 to 60 degrees with 1 degree step. The scattered X-rays are collected through a detector in a vacuum chamber. The data then is interpolated and curve for Intensity vs Q_z is extracted for fitting. The schematic of the experiment is shown in Figure 10a, and the actual experimental setup is depicted in Figure 9.

One of the major challenges in the experiment is the alignment of the sample. Since the

area of interest is small, roughly a mm by a mm, the sample needs to be perfectly positioned at the center of the rotation. Then the rotation axis is brought to the beam position. The goal is to keep the same area sample in the sample while rotating. This is achieved by aiming a camera at the sample and focusing on the beam position. A short measurement is then performed to observe the diffraction pattern of the sample. The sample is adjusted until the diffraction pattern is horizontal and centered.

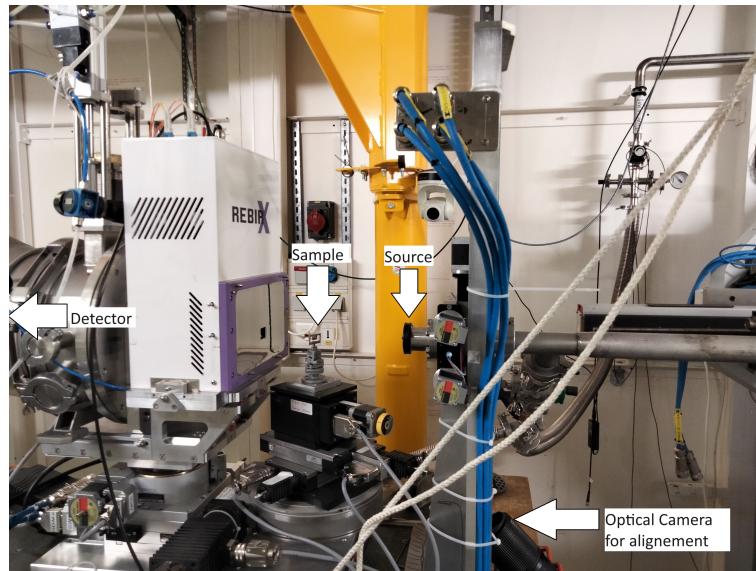
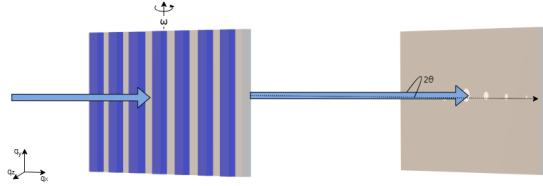
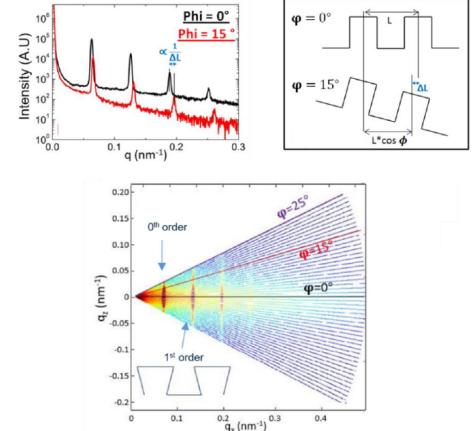


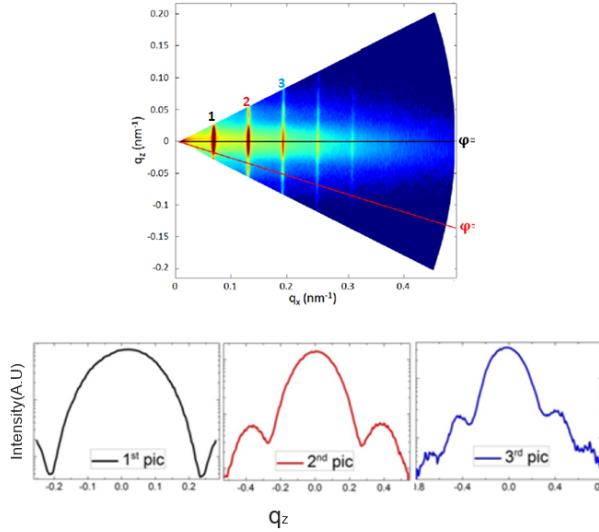
Figure 9: Picture of the experimental setup at the BM02 beamline at ESRF.



(a) Experimental setup



(b) The sample is rotated to get various cuts in the reciprocal space



(c) The obtained data is interpolated and data for fitting is extracted

Figure 10: CD-SAXS instrument layout: X-ray beam passes through the sample, with scattered intensity measured by a 2D detector and converted to $I(Q_x)$. Measurements at various angles (ω) are converted, showing intensities as I vs. Q_Z .

3.3 Scattering Model

We will not delve into the complete theory of SAXS here, as it exceeds the scope of this document. However, Chapters 1 and 2 of this book [9] offer excellent detailed insights into the underlying theory. Building upon this, we will now describe the diffraction of a collimated X-ray beam as follows:

$$I(\mathbf{Q}) = \Omega |\mathcal{F}(\mathbf{Q})|^2, \quad (1)$$

where $I(\mathbf{Q})$ represents the scattered intensity as a function of the scattering vector \mathbf{Q} , Ω is a variable independent of \mathbf{Q} , and $\mathcal{F}(\mathbf{Q})$ is the Fourier transform of a function describing the electronic density distribution within the nanoimprinted pattern.

This relationship is considered valid within the limitations of the CD-SAXS geometry, which includes a transmission geometry and a low probability of multiple scattering. Unfortunately, the measured intensity is the absolute square of the Fourier transform. Therefore there is a loss of the phase information, making it impossible to analytically extract $F(\mathbf{Q})$ from $I(\mathbf{Q})$. Therefore, the primary method for determining feature dimensions involves constructing a real-space model of the pattern cross-section. The Fourier transform of this model is then fitted to the experimental CD-SAXS data.

During my work-study program, we were mainly concerned with a periodic array of lines, referred to line gratings afterwards (see figure 10). The cross-section of these lines can be represented as a stack of trapezoids (see figure 11).

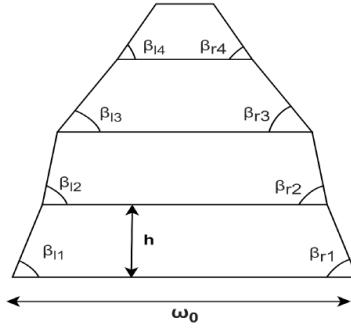


Figure 11: Cross-section of a nanostructure line represented as a stack of trapezoids.

The analytical Fourier transform of a trapezoidal shape can be calculated with the following expression [6]:

$$F(q_x, q_z) = \frac{1}{q_x} \left[-\frac{m_1}{t_1} e^{-iq_x(\frac{\omega_0}{2})} \left(1 - e^{-ih(\frac{q_x}{m_1} + q_z)} \right) + \frac{m_2}{t_2} e^{-iq_x(\frac{\omega_0}{2})} \left(1 - e^{-ih(\frac{q_x}{m_2} + q_z)} \right) \right] \quad (2)$$

where,

$$\begin{aligned} m_1 &= \tan(\beta_1) \\ m_2 &= \tan(\pi - \beta_r) \\ t_1 &= q_x + m_1 q_z \\ t_2 &= q_x + m_2 q_z \end{aligned}$$

The different parameters used in the equations are illustrated in figure 11. The β s represent the side wall angle of the trapezoid, ω_0 is the line width and h is the height of the trapezoids.

so,

$$I_0(\mathbf{q}) = |F(q)|^2 \quad (3)$$

Our analytical model predicts a perfect shape. In order to take into account defects inherent to the lithography process, additional factors are introduced to account for an additional decay of scattered intensity $I(Q_x)$. These defects can be illustrated as variations a distribution in the in-depth line profile. Two main distributions are considered:

- **Random variations in average line position:** the in-depth line profile remains identical along the entire line but the average position of the line fluctuates slightly across the whole sample.
- **Variations in the in-depth line profile:** the in-depth line profile itself varies.

Both factors indicate a degree of long-range order within the pattern. Additionally, they provide insights into specific types of line edge roughness [10]. To account for this distribution, we introduce an effective Debye-Waller factor, similar to the one used for fluctuations in crystal lattices.

Hence,

$$I(\mathbf{q}) = I_0(\mathbf{q}) \exp(-q^2 DW^2) \quad (4)$$

where DW is the Debye-Waller factor.

3.4 Fitting Algorithm

While CD-SAXS excels at detecting deviations from a perfect grating pattern in buried structures, it requires additional processing to convert the raw data into a meaningful real-world structure. This process involves using an inverse algorithm, which essentially translates the scattered intensity information back into the original structures characteristics.

CD-SAXS measurements provides experimental data. The objective of the inverse fitting is to iterate over a high number of different line profile (represented as various combination of

stacked trapezoids) and to converge towards the profile that its Fourier Transform will best match the experimental data. While the objective seems simple to describe, the problem is complex. Traditional optimization methods used for refinement often fall short when dealing with complex internal structures with numerous parameters, either being trapped in local minima or not converging toward the same solutions.

Another challenge arises from the possibility of "degenerate" solutions. These occur when multiple structural models can produce the same scattering data, making it difficult to pinpoint the true structure. This is a common issue in scattering analysis.

Therefore, the ideal scenario for CD-SAXS analysis involves an optimization algorithm that can consistently and rapidly converge on the best possible fit for the data. While some prior knowledge about the underlying structure can accelerate the process, such information is not always readily available. This highlights the need for more efficient algorithms that can handle complex structures even with limited prior knowledge.

Previous research has explored various algorithms to determine the optimal set of parameters for a model that best fits the experimental CD-SAXS data. These parameters essentially describe the nanostructure being analyzed.

One approach utilizes a Markov chain Monte Carlo (MCMC) algorithm. However, this method requires a good initial guess for the structure parameters and limitations on their search range. Additionally, it necessitates multiple independent runs to ensure that the algorithm converges on the correct solution. While this approach can be effective, the need for tight parameter bounds might overlook potential fabrication errors in the sample.

Another strategy involves massive computing resources with parallelization and highly refined grid-based models. This method, known as reverse MCMC, offers greater accuracy but is limited by the availability of such computational power.

Genetic and evolutionary algorithms have emerged as promising alternatives. These methods mimic biological evolution, with the model parameters acting as the "genetic code." Starting with randomly generated parameters, these algorithms iteratively refine them through a "mixing strategy" over multiple generations until the optimal set is found. This approach excels at searching large parameter spaces with wide bounds, making it suitable for complex structures. [11]

3.4.1 Covariance Matrix Adaptation Evolution Strategy (CMAES)

One algorithm is the Covariance Matrix Adaptation Evolution Strategy (CMAES). This method is particularly well-suited for high-dimensional optimization problems, making it ideal for complex nanostructure analysis. CMAES operates by maintaining a population

of candidate solutions, with each iteration generating new candidates based on the previous generation performance. By adapting the covariance matrix of the candidate solutions, CMAES can efficiently explore the parameter space and converge on the optimal solution.

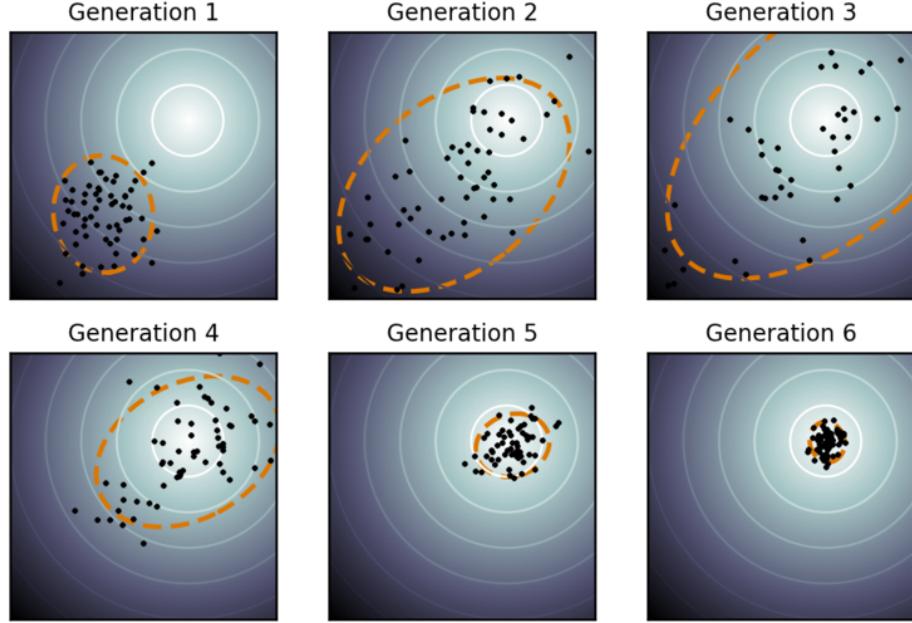


Figure 12: Illustration of CMAES algorithm. The algorithm maintains a population of candidate solutions, with each iteration generating new candidates based on the previous generation performance.(image taken from wikipedia CMAES page)

For CD-SAXS experiment, the algorithm starts with the experimental data collected. Then, a series of in-depth line profile generated, through a set of parameters as describe earlier. Afterwards, the calculated analytical Fourier transform is compared with the experimental data using a mean-absolute error log:

$$\Xi = \frac{1}{N_q - 1} \sum_q |\log_{10} I_{\text{Sim}}(q) - \log_{10} I(q)| \quad (5)$$

where $I_{\text{Sim}}(q)$ is the simulated intensity and $I(q)$ is the experimental intensity. Ξ is called goodness of fit. The algorithm then tries to minimize Ξ by adjusting the parameters of the model.

In an article [11], researchers investigated the efficiency of various algorithms for reconstructing various nanostructures using X-ray scattering data. Their findings specifically highlighted the advantages of the CMAES. Compared to other methods like Markov Chain Monte Carlo (MCMC) and Differential Evolution (DE), CMAES demonstrated significantly faster

convergence times when analyzing an experimental structure. Notably, CMAES reached a solution that matched the quality of previous studies in approximately 1 to 2 orders of magnitude less time than MCMC and less than an order of magnitude time than DE. This speed advantage held true regardless of the specific objective function used to evaluate the goodness of fit. These results suggest that CMAES offers a powerful tool for analyzing complex nanostructures with X-ray scattering data, particularly when dealing with limited computational resources or tight time constraints.

3.4.2 Uncertainty estimation by Monte Carlo Markov Chain(MCMC) method

The CMAES algorithm provides a single best-fit solution for the nanostructure parameters. However, it is essential to understand the uncertainty associated with these parameters. This uncertainty relates to the different possible combinations of parameters that could result in a similar goodness of fit. For instance, decreasing slightly height of one trapezoid and increasing the height of an other one can result in a similar goodness of fit. To address this, we can use MCMC algorithm to explore and find all the sets of population that can result in the same goodness of fit.

This confidence interval then can be calculated from this population of parameters. This interval gives us an idea of the range of possible values for each parameter that could still provide a good fit to the data. The lower and upper bounds of this interval can be used to estimate the uncertainty associated with each parameter.

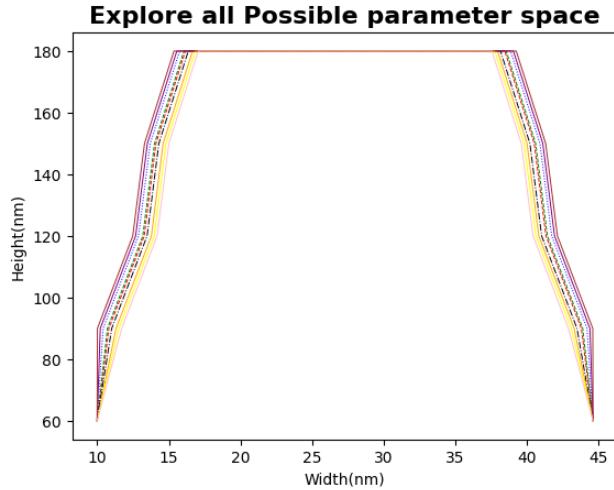


Figure 13: MCMC algorithm exploring all possible sets of parameters that can result in the same goodness of fit. The different colours represent the different set of parameters that can result in the same goodness of fit.

After determining the best-fit model structure, the researchers of this article [12] employed

a MCMC algorithm to calculate the uncertainties associated with the model parameters. This algorithm generates a population of models that can be evaluated to assess the uncertainty in a set of parameters. The probability of a given model being accepted into the population depends on how well the simulated scattering profile it generates matches the experimentally measured one.

Here is the overview of the algorithm:

1. **Seeding:** The algorithm initializes with the model (M) exhibiting the best known goodness-of-fit, denoted GFB.
2. **Proposal generation:** Random perturbations are applied to each parameter within the model, resulting in a new candidate model (M_i) and its corresponding goodness-of-fit GF (GF_i).
3. **Acceptance for better fit:** If $GF_i < GF_{i-1}$, then M_i is automatically accepted into the population (and GFB is updated to $GFB = GF_i$ if the new model has a better fit).
4. **Metropolis-Hastings acceptance for worse fit:** If $GF_i > GF_{i-1}$, the probability (P_i) of accepting M_i is calculated using Eq. (6):

$$P_i = \exp(-0.5 \cdot (GF_i - GFB)) \quad (6)$$

A random number α between 0 and 1 is then generated. If $\alpha < P_i$, M_i is accepted; otherwise, it is rejected, and a new proposal is generated from M_{i-1} .

5. **Equilibrium and resampling:** Steps 2-4 are repeated until the model population reaches an equilibrium. To avoid correlations between accepted models, the population is resampled (e.g., every 50 steps in this case).
6. **Uncertainty calculation:** The uncertainties are calculated from the final accepted model population and represent a 95% confidence intervals. The step size for generating proposals was optimized to achieve an acceptance probability between 0.25 and 0.35, a range known to yield the fastest convergence.

I used a very similar algorithm but with some modifications to increase the efficiency, we will discuss this in the next section.

3.4.3 Overall view of the CD-SAXS inverse problem resolution

The process begins with an initial guess for the parameters of the model, which describe a geometric structure with specific width and height parameters. The model data is then

transformed into the reciprocal space using a Fourier Transform. This transformed data is then compared with the experimental data. Finally, the error between the simulated and experimental data is minimize by exploring the parameters space.

The error between the experimental and simulated data is quantified using an error metric that measures the goodness-of-fit by comparing the logarithms of the simulated and experimental intensity profiles. If the error is within a tolerable range, the fit is considered acceptable, and the best-fit model is extracted. This involves identifying the parameters that provide the best match to the experimental data.

To quantify the uncertainties associated with the fitted parameters, a Monte Carlo approach is employed. This involves exploring the possible parameter space through a series of stochastic simulations. The MCMC method is used to sample the parameter space, ensuring that the distribution of the sampled parameters reflects their likelihood given the data. During the MCMC process, initial samples are generated from the best-fit parameters, new samples are proposed by perturbing the current parameters, and the Metropolis-Hastings criterion is used to decide whether to accept or reject the new samples based on their goodness-of-fit compared to the previous samples. This process is repeated until a sufficient number of samples are collected, ensuring the sampled parameter distribution converges to the true posterior distribution.

The final step involves calculating the error bars (uncertainties) for the fitted parameters based on the Monte Carlo samples, providing 95% confidence intervals and a robust measure of the reliability of the fitted model parameters. By combining CMAES for initial parameter optimization and MCMC for uncertainty quantification, the algorithm offers a robust approach to model fitting, ensuring both optimal parameter estimation and reliable uncertainty analysis.

Here is a figure that shows the overall view of the CD-SAXS algorithm:

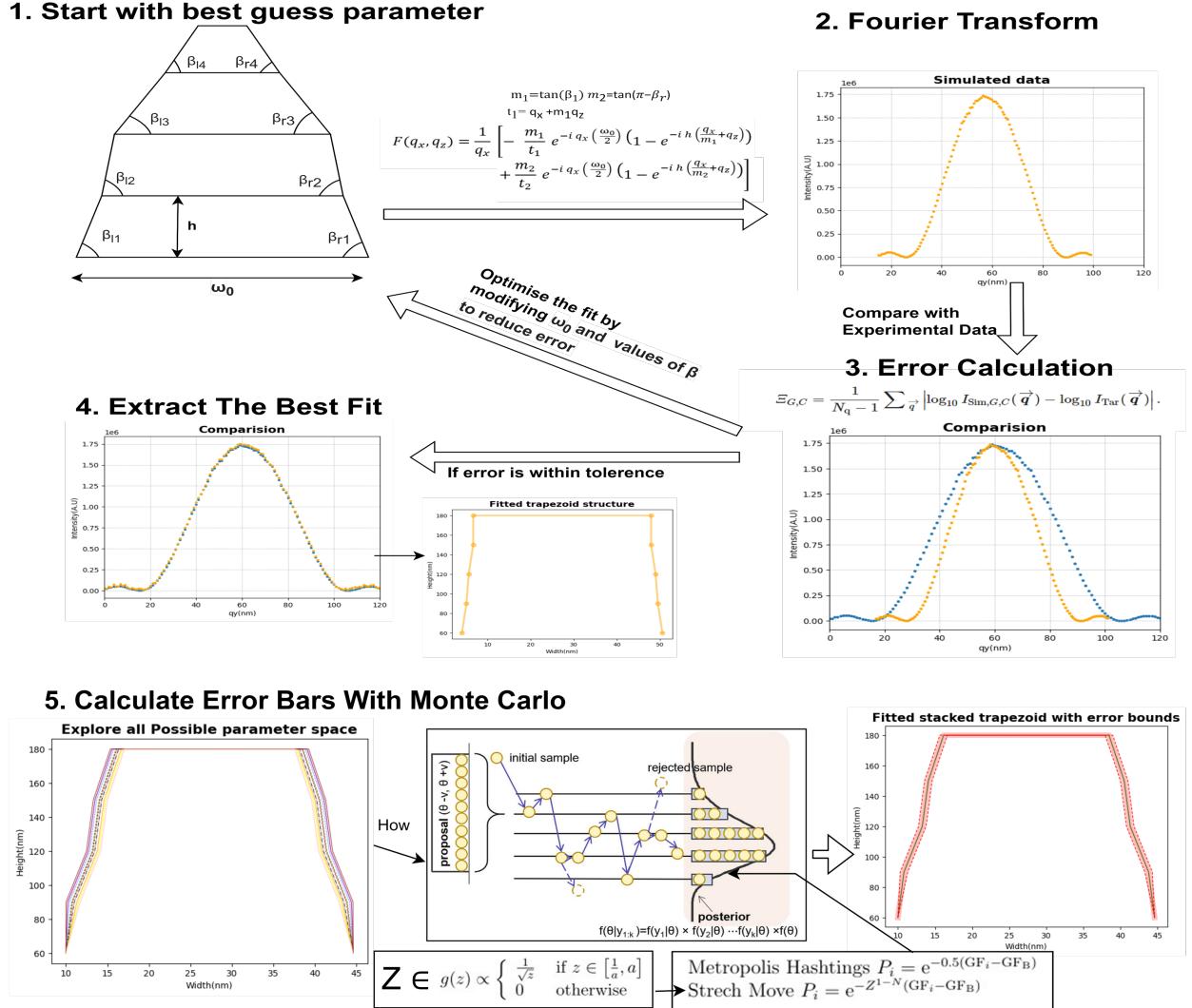


Figure 14: Illustration of the different steps and algorithm used during the CD-SAXS analysis. First, CMAES for initial parameter optimization and then MCMC for uncertainty quantification are used.

4 My contribution

4.1 Introduction

After outlining the context of the work-study program and describing the technique I worked on, it is finally time to discuss my contributions to the project. First, I will explore the design I implemented and explain why this specific design was chosen. Next, we will examine the simulation models I developed and their applications. Following this, I will detail the optimization and parallelization of the code that was done to enhance its performance. Finally, we will discuss how this optimized code could be utilized to estimate uncertainty in real-time during synchrotron experiments.

4.2 Design

To gain a clear understanding of how the developed algorithm work, the design of the algorithm is illustrated with the help of an UML diagram in figure 15. This diagram offers a roadmap, highlighting the various components and their interactions. We will then delve deeper to explore each component role in the algorithm, as explained in the previous section.

4.2.1 Components

Fitter:

This class is a crucial component of the design. It includes the *cmaes* function for estimating the best-fit parameters and the *mcmc* function for assessing the uncertainty in the fit. The class takes a simulation model and experimental data as input. When the *cmaes* function is called, it returns the best-fit parameters. Subsequently, the *mcmc* function can be invoked to provide statistical information about the best fit, including the uncertainties in the parameters.

Residual:

This class calculates the residuals between the experimental data and the model. Currently, we use the log-likelihood as the residual function, but it can be easily extended to other residual functions. The *Fitter* class calls this class and provides the relevant model. The *Residual* class then uses the model *simulate_diffraction* function to calculate the model diffraction pattern and compare it with the experimental data.

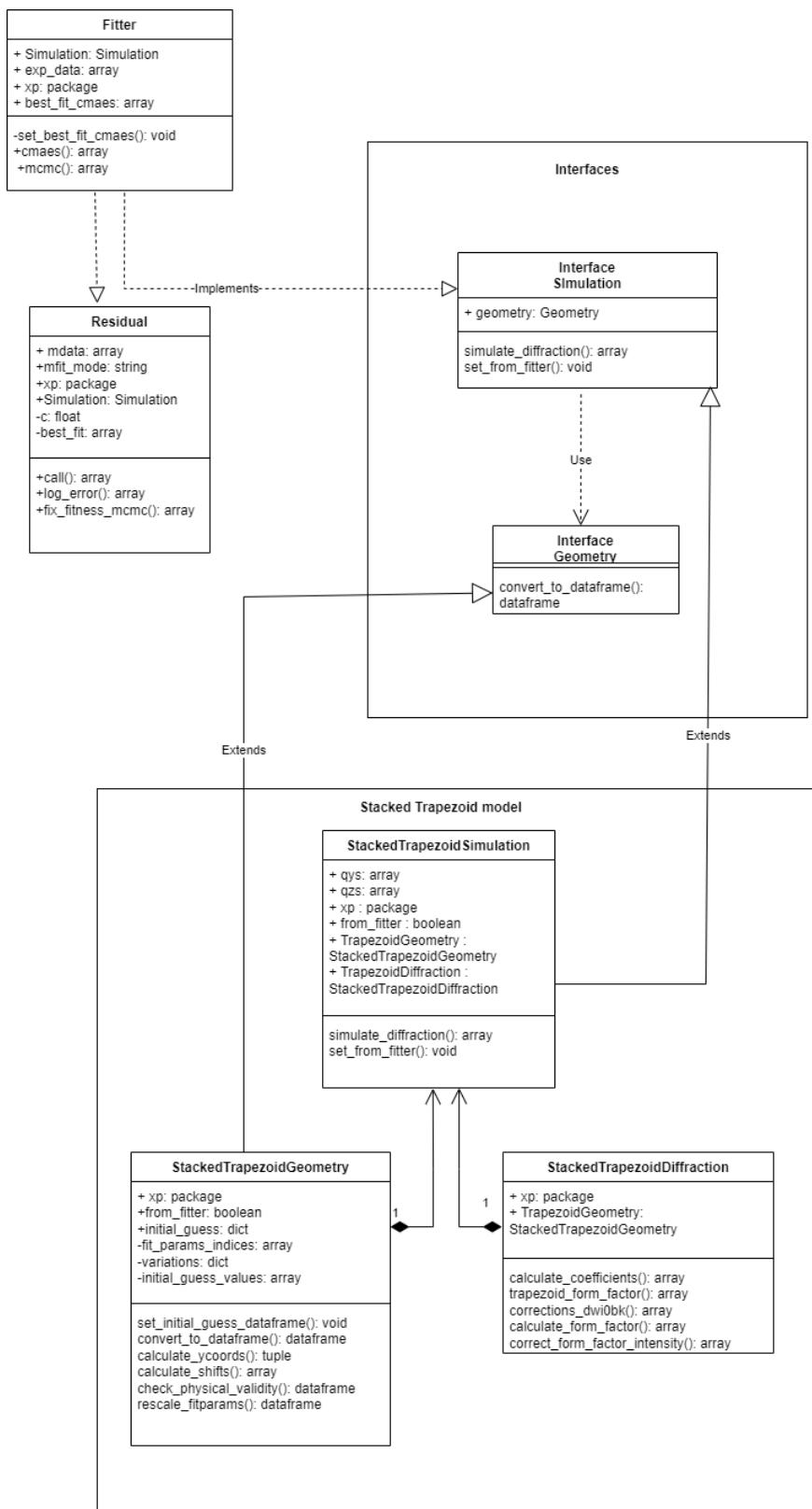


Figure 15: UML diagram of the design for CD-SAXS simulation application.

Interface Simulation:

An interface defines a contract that classes must follow, specifying a set of methods that implementing classes should have. In Python, I have chosen to use Protocol to achieve a similar effect. This is the base class for all simulation models. The functions and classes used in it should be implemented in all simulation models. This interface simplifies future model building by ensuring that simulation functions are not geometry-dependent.

Interface Geometry:

Similar to the Interface Simulation, this is the base class for all geometry models. This interface simplifies future model building by ensuring that geometry functions are not dependent on the specific simulation details.

Model:

In this UML diagram (figure 15), we present the implementation of the stacked trapezoid model. Central to this model is the *StackedTrapezoidSimulation* class, which is a composite class integrating *StackedTrapezoidGeometry* and *StackedTrapezoidDiffraction* classes. The *StackedTrapezoidGeometry* class handles all geometrical calculations and stores the relevant information, while the *StackedTrapezoidDiffraction* class is responsible for all diffraction-related calculations.

These classes work together to simulate the physics and generate data that can be compared with experimental results. Throughout the project, additional models are being developed and will be discussed later. The stacked trapezoid model serves as a prototype, illustrating how other models will be implemented. Each model will adhere to the base interfaces *Simulation* and *Geometry*.

4.2.2 Relationships between components

This structure is designed to simulate and analyze diffraction patterns using a modular approach. At its core, the system comprises several key components: the *Fitter*, *Residual*, and interfaces for *Simulation* and *Geometry*, along with specific model implementations like the *StackedTrapezoidSimulation*. The *Fitter* class orchestrates the fitting process by using the *cmaes* function to estimate the best-fit parameters and the *mcmc* function to assess the uncertainty in the fit. It takes in a model and experimental data, utilizing the *Residual* class to compute the difference between the experimental data and the model simulated data. The *Residual* class leverages the model *simulate_diffraction* function to generate the diffraction pattern, which it then compares with the experimental data.

The model, meaning specific implementations of the interfaces, is designed to operate independently, allowing users to utilize a particular model for simulations without needing to engage with the fitting component. This autonomous functionality ensures that users can easily perform simulations solely with the model of their choice. This design choice enhances flexibility and usability, as it decouples the simulation process from the fitting procedures, making it more accessible for users who may only need to run simulations. The usefulness and implications of this design choice will be discussed shortly.

4.2.3 Why this design?

We will first discuss the shortcomings of the previous design before explaining why the current design was chosen.

Previous design:

The previous design adopted a monolithic approach, where the fitting and simulation components were tightly coupled. This meant that any changes to the fitting algorithm (e.g., adding a new optimization technique) would necessitate modifications to the simulation code as well. This interdependence made it difficult to introduce new functionalities or modify existing ones without potentially impacting other parts of the system.

```

metrocarac > projects > sas > P_23_10_Nischal > code_parallel > fit.py > ...
1 > ....
9 > import os ...
10
11 > creator.create('FitnessMin', dbase.Fitness, weights=(-1.,)) # to minim. fitness
12 > creator.create('Individual', list, fitness=creator.FitnessMin)
13
14
15 > def find_peaks(values, positions, threshold): ...
16
17
18 > def fit_peaks(peaks): ...
19
20
21 > def maxleftright(positions, values):...
126 *
127
128 > def corrections_dwl0bk(intensities, du_factors, dw_factors, ...
168
169 > def trapezoid_form_factor(qxs, qzs, y1, y2, langle, rangle, height):...
203
204
205 > def stacked_trapezoids(qxs, qzs, y1, y2, height, langle, ...
255
256
257 > def log_error(exp_i_array, sim_i_array):...
288
289
290 > def std_error(exp_i_array, sim_i_array):...
303
304
305 > def cmaes(data, qxs, qzs, initial_guess, sigma, ngen, ...
496
497
498 > def fittingp_to_sim(fit_params, initial_guess):...
530
531
532 > def fittingp_to_simp(fit_params, initial_guess):...
559
560
561 > class PickableResidual(): ...
629
630
631 > def spectra_calcul(params, qxs, qzs):...
644
645
646 > def fitting_cmaes(qxs, qzs, intensity, ...
701
702

```

Figure 16: Monolithic design of the previous version. This script combined all the functions for the stacked trapezoid model in one file.

Furthermore, individual models within the system implemented their own fitting and simulation functionalities. This resulted in a significant amount of code duplication, leading to inconsistencies between models. Moreover, adding new models required rewriting these functionalities from scratch, hindering the system scalability. This approach proved inefficient for managing a growing number of models. These two limitations combined made it challenging to optimize or parallelize the code effectively. Since the fitting and simulation components were intertwined, it was difficult to isolate and optimize individual sections.

Similarly the codes for the different models had their own fitting and simulation components, leading to code duplication and inconsistencies across models. This design was not scalable, as adding new models required rewriting the fitting and simulation components for each model.

Current design:

The current design improves upon the previous version by decoupling the fitting and simulation components, resulting in a modular structure. The *Simulation* interface serves as the connector between fitting and simulation, standardizing the process of fitting any model.

Also, by enabling simulation models to be developed independently of the fitting component, new models can be introduced without modifying the existing codebase. Developers can focus on developing new models without being concerned by the fitting process. This design choice significantly enhances the system flexibility and scalability.

Decoupling the components also facilitates the optimization and parallelisation. Indeed one can now focus on specific areas of improvement without considering cross-component interference. For example, the simulation component can be optimized to run faster and more efficiently through advanced numerical methods and parallel computing techniques. This is particularly beneficial for handling large datasets or complex simulations that demand significant computational resources. By isolating the simulation component, developers can experiment with and implement various optimization strategies, ensuring that the component runs as efficiently as possible.

4.3 Brief Overview of Photoresist Lithography

For appropriate context, We will now briefly discuss the Photoresist lithography before moving on to the simulation models. It is a technique used to fabricate micro and nanoscale structures on substrates. It works by first coating the wafer with a light-sensitive material called photoresist. Then, a patterned mask blocks light from specific regions of the photoresist.

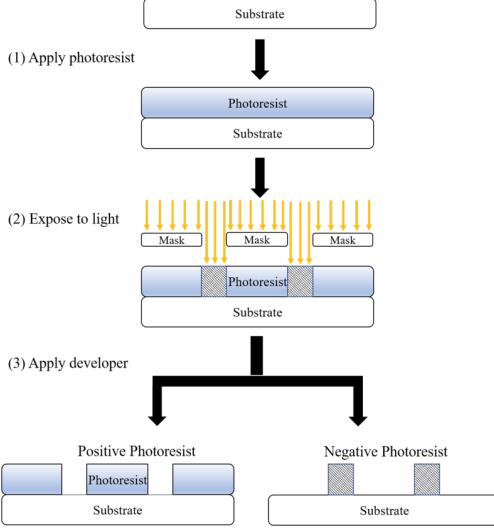


Figure 17: Schematic of the photoresist lithography process (taken from Wikipedia).

Exposing the wafer to light triggers a chemical change in the exposed photoresist, making it either soluble (positive tone) or insoluble (negative tone) in a developer solution. This developer removes the unwanted photoresist, revealing the desired circuit pattern on the underlying wafer. Optionally, etching can be used to create physical features by removing exposed areas of the substrate.

In the realm of nanoscale fabrication, techniques like photoresist lithography stand out as pivotal. To model the structures resulting from photoresist lithography, we can observe that many of these structures can be approximated by stacked trapezoids. This simplification forms the foundation of our modeling approach. By considering the cross-sectional shapes of these lines, which closely resemble trapezoids, we can create computational models that simulate the fabrication process and predict the resulting geometries.

4.4 Simulation Models

While developing our model, we consider lines of nanostructures on a substrate created using the photoresist lithography (see Figure 18) obtained through this process.

Our models aim to balance accuracy and computational efficiency. While real structures are complex and not perfectly trapezoidal, using a series of stacked trapezoids provides a practical approximation that can be refined as needed. This approach serves as a robust starting point for simulations, allowing us to incrementally increase the number of trapezoids to better match the real structures.

Furthermore, we extend this model to account for rounded corners, which are common in actual fabricated structures. By approximating these rounded corners with smaller trape-

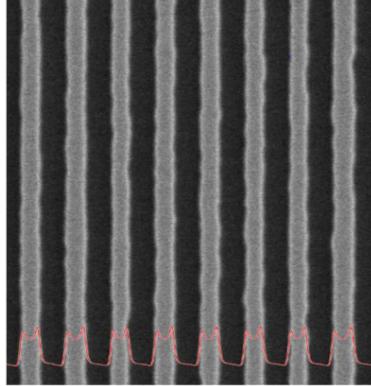


Figure 18: Image obtained with a scanning electron microscope in top view of the structure for modeling. In red, the average signal over the length of the lines is superimposed.[10]

zoids, we can enhance the accuracy of our simulations without significantly increasing computational costs. Future developments may explore more direct modeling techniques, such as using sections of circles, to further improve the fidelity of our models.

Lastly, we introduced the modeling of overlay phenomena, which consists of the misalignment of different layers on a wafer during consecutive lithography stages.

4.4.1 Stacked Trapezoid Model

A close examination of the cross-section of the lines in figure 19 reveals that their shapes can be closely approximated by stacked trapezoids. This crucial observation forms the foundation of our modeling approach.

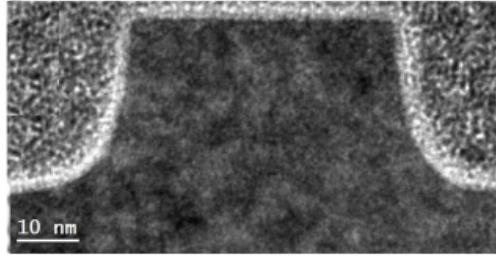


Figure 19: Tunneling electron microscope image of the cross section of the lines.[8]

Figure 11 illustrates the approximation of the model we aim to simulate. The model inputs include a constant height (h) or sets of heights, sidewall angles (β), and the bottom width (ω_0) of the trapezoids. By allowing variable heights for each trapezoid layer, we can achieve better control over the structure, albeit at the cost of increasing the number of parameters to fit. The user can determine the height parameter based on their specific needs. Given the z-coordinates of each corner of the trapezoids, the x-coordinate for each can be calculated using the following formula:

$$x_i = x_{i-1} + \frac{z_i - z_{i-1}}{\tan(\beta)} \quad (7)$$

As discussed in Section 3.3, we can use Equation 2 to calculate the Fourier transform of a single trapezoid and then sum them all up of all the trapezoids to obtain the diffraction pattern.

This model has its limitations, as it simplifies the real structure. The actual structure is more complex and does not consist of perfect trapezoids. Additionally, a more detailed model would entail higher computational costs. Therefore, a balance between accuracy and computational cost is necessary. Despite these limitations, this model provides a solid foundation for simulations. By increasing the number of trapezoids, we can approximate the real structure more closely. Future models will inherit from or build upon the *StackedTrapezoidSimulation* and *StackedTrapezoidGeometry* classes.

4.4.2 Rounded corners model

The rounded corners model is an extension of the stacked trapezoid model. It is designed to simulate the rounded corners often found in real structures (see figure 19). Each corner of the trapezoid is rounded using a circle, with the radius of each circle provided as an input by the user and fitted later. Once the radii are known, the corners are approximated by dividing them into even smaller trapezoids. This allows us to use the same diffraction code written for the stacked trapezoid model to simulate the rounded corners. With this model, we can simulate the diffraction pattern more accurately, reflecting the real structure more closely.

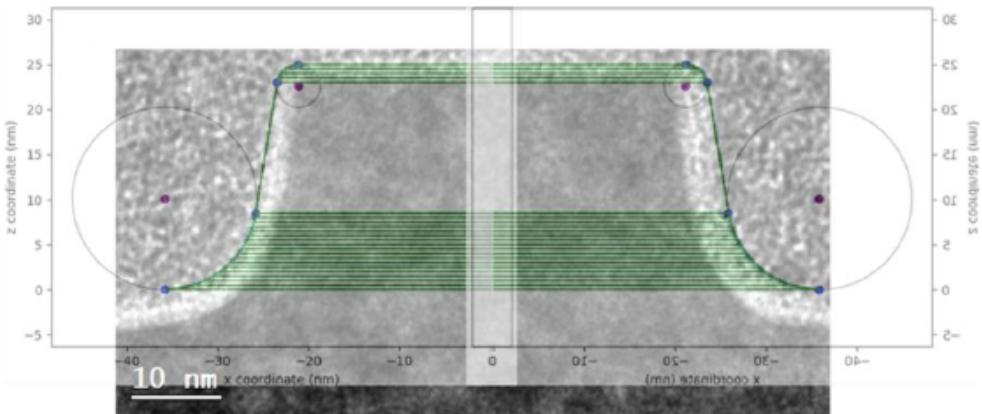


Figure 20: Rounded corners model fitted on top of the scanning electron microscope image. (Image taken from the Work-study project report of Timothee Choisinet.)

While this model has not been implemented yet in the current version of the application, a similar codebase already exists. This existing code provides a strong foundation for future implementation in the new workframe.

Simulating rounded corners with a large number of trapezoids significantly increases the computational cost of the model. To address this, exploring alternative approaches for representing rounded corners is a promising avenue for future development. One potential solution lies in directly incorporating a section of a circle into the model. This could involve calculating the direct Fourier transform of the circular segment and then summing the resulting components to obtain the diffraction pattern. This method has the potential to significantly reduce the computational complexity compared to using a multitude of trapezoids.

4.4.3 Overlay model

The latest and future generation of transistors are transitioning from planar structure towards 3D structure such as fin field-effect (FinFET) and Gate All Around (GAA). (see figure 21) Transistors are built layer-by-layer through a series of lithography steps. Each layer involves depositing a specific material (like a metal for contacts or a special material for transistors) in a defined pattern. Precise alignment between these layers is crucial for the final device to function properly. This means features like contacts, lines, and transistors must be perfectly positioned relative to each other on the wafer.

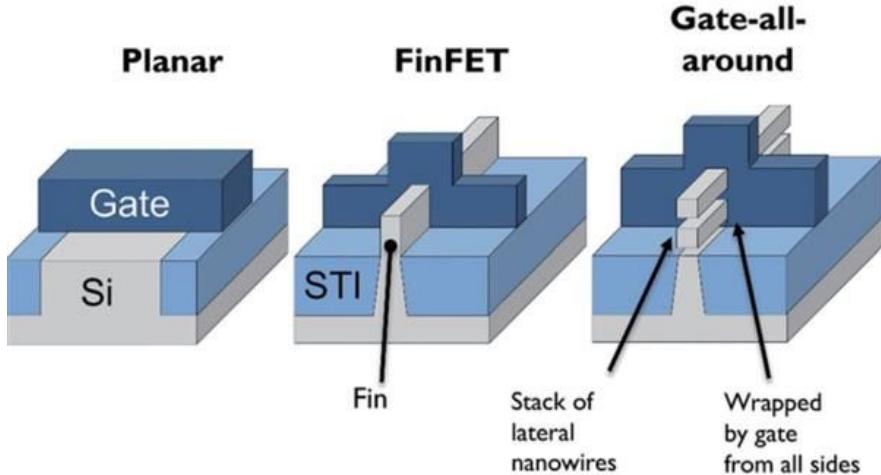


Figure 21: Different kinds of transistors.

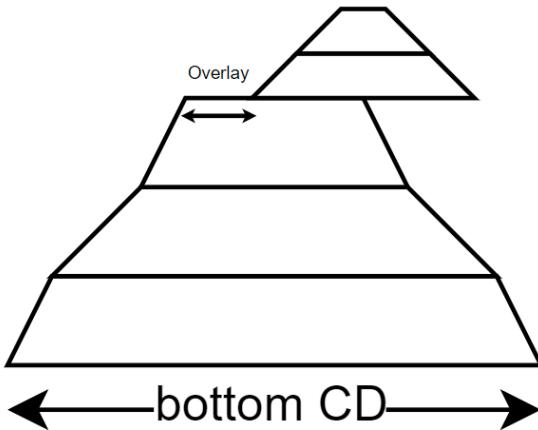


Figure 22: Overlay model: The first stack is three trapezoids and the second stack of two trapezoid is shifted by an amount called the Overlay.

The overlay model is designed to simulate the critical aspect of layer alignment during wafer fabrication. This model builds upon the concept of the stacked trapezoid model, but it also introduces overlay between two lithographical layers. It incorporates two sets of trapezoids: set of three and set of two for each layer in this case. The key difference lies in the positioning of the second layer of trapezoids. These are shifted by a specific amount, called the overlay, in the x-direction. This shift represents potential misalignment between layers.

To calculate the coordinates of the second layer trapezoids, we can leverage the same method used for the stacked trapezoid model. However, for the second layer, an additional x-axis shift equal to the overlay value is applied.

To add the overlay model to the code, we inherit from the *StackedTrapezoidSimulation* and *StackedTrapezoidGeometry* classes. The *StackedTrapezoidGeometry* class will be modified to include the shift in the x direction for the second layer. This shift is entered by the user and can be fitted later on.

4.5 Implementation of MCMC to Measure Uncertainty

As discussed in Section 3.4.2, after finding the best fit for a model using the *cmaes* algorithm, it is crucial to determine the uncertainty for each parameters of the model. This is where the *mcmc* method of the *Fitter* class becomes essential. While we follow a similar approach to the algorithm described in Section 3.4.2, there are key differences that will be discussed in this section, along with the method implementation and its outputs.

We utilize the *emcee* Python package [13] to implement this method. This package is well-

regarded for its ease of use and effective implementation of the MCMC algorithm, having been referenced in several scientific papers [14]. A significant advantage of *emcee* is the flexibility it offers in choosing the type of moves for the MCMC algorithm. Unlike the Metropolis-Hastings criterion used in Section 3.4.2, *emcee* employs the stretch move by default, which has been shown to achieve faster convergence [13, 15]. The stretch move uses the following formula to determine whether to accept or reject a move:

$$P_i = e^{Z^{1-N}(GF_i - GFB)} \quad (8)$$

where P_i is the probability of accepting the move, N is the number of parameters, GF_i is the current goodness of fit, and GFB is the goodness of fit of the initial fit. Z is drawn randomly from the distribution:

$$Z \in g(z) \propto \begin{cases} \frac{1}{z} & \text{for } z \in [\frac{1}{a}, a] \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

In addition, *emcee* includes a built-in function to calculate the autocorrelation time of the chain, which helps determine which part of the chain to use for statistical analysis. We implemented the *mcmc* method using the default stretch move of *emcee*, but also provided options to use other moves like Metropolis-Hastings and Gaussian moves.

For the implementation, the user needs to provide the following inputs:

- The number of parameters
- σ , which is the standard deviation of the initial walker population
- The number of steps to run the MCMC algorithm
- The number of walkers
- A directory to save the statistical analysis information (optional)

Unlike previous approaches [12], we do not vary the model parameters directly. Instead, we use a normalized space where variations range between $-\sigma$ and σ . These normalized values are then rescaled to the actual parameter values using the following formula:

$$\text{Rescaled Params} = \text{Normalized Params} \times \text{Allowed Range} + \text{Best Fit Cmaes} \quad (10)$$

This approach provides better control over the parameter range, preventing unnecessary exploration of regions far from the best fit. It also allows the best fit obtained from the CMAES algorithm to serve as the mean of the initial walker population.

After extracting the chain from the MCMC algorithm, we perform a statistical analysis. This includes calculating the mean, standard deviation, total number of individuals in the chain, minimum and maximum parameter values, lower and upper confidence intervals as specified by the user, and the uncertainty in the parameters, defined as the difference between the upper and lower confidence intervals.

| | mean | std | count | min | max | lower_ci | upper_ci | uncertainty |
|----------|----------|----------|-------|----------|----------|----------|----------|-------------|
| height1 | 20,9913 | 0,388484 | 62370 | 15,38546 | 23,47261 | 20,9873 | 20,99531 | 0,004007 |
| height2 | 24,03406 | 0,437929 | 62370 | 19,33031 | 30,92674 | 24,02954 | 24,03858 | 0,004517 |
| angle1 | 1,541163 | 0,049045 | 62370 | 1,248772 | 2,23387 | 1,540657 | 1,541669 | 0,000506 |
| angle2 | 1,498193 | 0,042529 | 62370 | 0,744646 | 2,297001 | 1,497754 | 1,498632 | 0,000439 |
| range1 | 1,612237 | 0,046393 | 62370 | 1,237489 | 2,198178 | 1,611759 | 1,612716 | 0,000479 |
| range2 | 1,442183 | 0,037094 | 62370 | 1,215714 | 2,245145 | 1,441801 | 1,442566 | 0,000383 |
| y1 | 0,007551 | 0,002295 | 62370 | 3,68E-05 | 0,045677 | 0,007528 | 0,007575 | 2,37E-05 |
| bot_cd | 39,99745 | 0,002691 | 62370 | 39,95721 | 40,03232 | 39,99742 | 39,99748 | 2,78E-05 |
| dwx | 0,087386 | 0,002301 | 62370 | 0,052874 | 0,113798 | 0,087363 | 0,08741 | 2,37E-05 |
| dwz | 0,089797 | 0,002609 | 62370 | 0,053889 | 0,141462 | 0,08977 | 0,089824 | 2,69E-05 |
| i0 | 10,00165 | 0,002688 | 62370 | 9,96178 | 10,02921 | 10,00162 | 10,00168 | 2,77E-05 |
| bkg_cste | 0,109749 | 0,002668 | 62370 | 0,08892 | 0,154258 | 0,109721 | 0,109776 | 2,75E-05 |

Figure 23: Example of output statistics file of the MCMC algorithm. The model used was the stacked trapezoid model.

In conclusion, the implementation of the MCMC method using the *emcee* package has proven to be an effective approach for quantifying the uncertainty in the parameters of our models. By leveraging the flexibility and advanced features of *emcee*, such as the stretch move and autocorrelation time calculation, we have enhanced the robustness and efficiency of our statistical analysis. The normalization and rescaling strategy we employed allows for better control over the parameter space, ensuring a more focused exploration around the best fit obtained from the CMAES algorithm. This method not only provides detailed statistical insights into the model parameters but also sets a solid foundation for future improvements and extensions. The versatility and precision of the MCMC method make it a valuable tool in the continuous effort to refine our simulations and better understand the underlying physical structures.

4.6 Parallelization and GPU Acceleration

The simulation of diffraction patterns can be computationally intensive, especially when dealing with large datasets or complex models. Given that the code is written in Python, which is inherently slower than compiled languages like C++, optimizing for performance is crucial. One effective strategy for improving performance is parallelisation. By distributing

the computational workload across multiple cores or processors, we can significantly reduce simulation time. In this section, we will briefly discuss parallelization and give an overview of the possible strategies for parallelization on both Graphics Processing Unit (GPU) and Central Processing Unit (CPU).

4.6.1 Overview of the concept of parallelisation

Parallelization involves breaking down tasks into independent units. Instead of performing a complex calculation in one go, the task is distributed across multiple processors for simultaneous processing. This approach leverages the power of modern hardware, with multiple cores on a single processor or distributed computing systems. The key is to achieve the right balance: tasks must be independent to avoid communication overhead yet large enough for efficient processing. While challenges exist, such as managing communication and limited parallelizability for certain problems, parallelization is a game-changer in fields like scientific computing, machine learning, and video processing.

If we can use the GPU to perform calculations in parallel, we can significantly accelerate the calculations. However, the GPU is not as flexible as the CPU. It is designed for parallel processing of large amounts of data, making it ideal for specific tasks like matrix multiplication. Nevertheless, the GPU is not well-suited for tasks that require branching or recursion, as these can slow down processing speed. Therefore, to take full advantage of the GPU, we need to structure our code in a way that aligns with the strength of GPU.

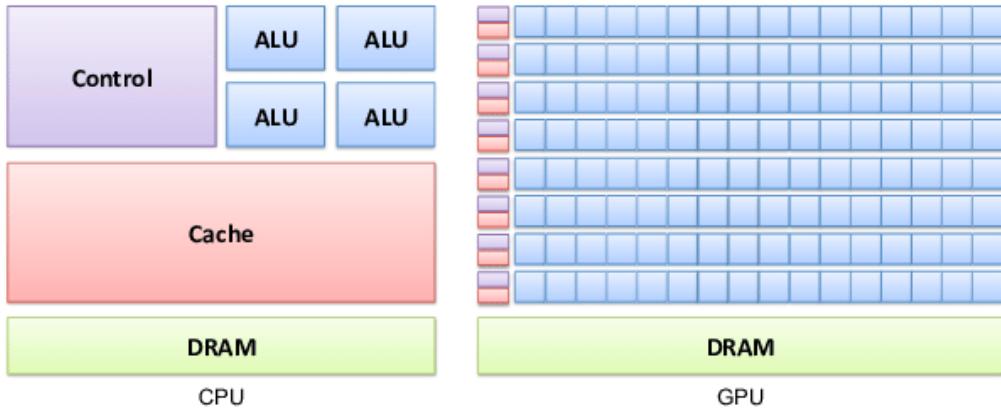


Figure 24: Comparison between GPU and CPU. The GPU has more cores but is less flexible than the CPU, making it suitable for simpler parallel tasks. [16]

4.6.2 Parallelization Strategies

To parallelize the code, we need to identify the independent parts. In both the *cmaes* and *mcmc* algorithms, a population is generated for each generation (or step), and each individual

in the population is evaluated independently through the calculation of the goodness of fit. Since one generation depends on the best fit of the previous generation, parallelizing the generation is not possible. However, parallelizing the evaluation of individuals in the population of one generation is possible.

Initially, to avoid the complexity of GPU parallelization, we used the multiprocessing library in Python to parallelize the code. This library allows us to run multiple processes in parallel, each on a different core. We used the *Pool* class from the multiprocessing library to create a pool of workers, each evaluating the goodness of fit. This approach reduced computation time significantly, as multiple individuals can be evaluated simultaneously.

The initial CDSAXS algorithms contained multiple for-loops, making the parallelization not possible. One of my first task was to replace replaced for-loops by NumPy operations. Indeed, NumPy is a highly optimized library for numerical operations. The major challenge was that the quantities we wanted to perform operations on were not of the same shape, requiring the use of NumPy to broadcast the different arrays we wanted to perform the operations on (see figure 26 for an example).

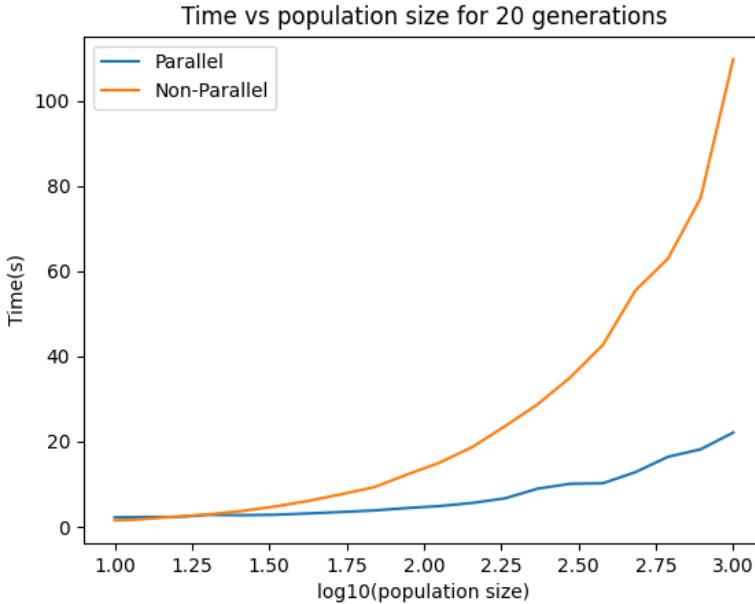


Figure 25: Multiprocessing used to improve the execution time of the code for the *cmaes* algorithm. The generation was fixed to 20.

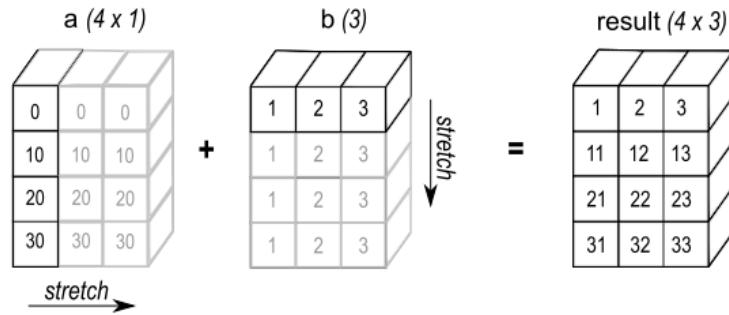


Figure 26: NumPy broadcasting used to perform operations on arrays of different shapes (image taken from NumPy documentation).

After reworking the initial code and fully replacing the for-loops by Numpy operations, it was easier to parallelize the code for GPU. As the code was already vectorized, we used the CuPy library, a GPU-accelerated version of NumPy. CuPy is built on top of CUDA, a parallel computing platform and application programming interface created by Nvidia. CuPy uses CUDA libraries like cuBLAS, cuDNN, cuFFT, cuSPARSE, etc., which are highly optimized for parallel computing (more information can be found in its documentation page [17]).

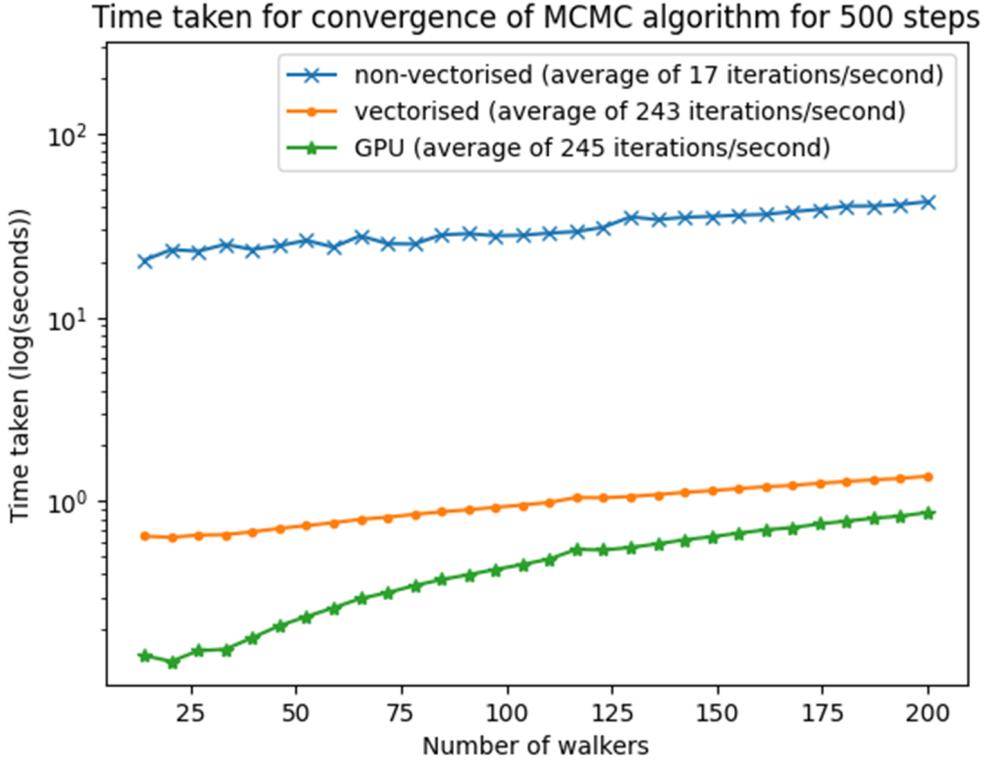


Figure 27: CuPy used to parallelize the code. The number of iterations per second in the parallel version is much higher. (This test was performed on CEA aar164 server Intel(R) Xeon(R) Platinum 8362 CPU @ 2.80GHz, 128 cores, and NVIDIA A100 80GB GPU)

The performances of the code between non-vectorized, vectorized and GPU compatible are presented on the figure 27. There are significant improvements between the vectorized and non vectorized code, more than an order of magnitude gain in time. However, the transition from a vectorized code to the GPU version only provided a marginal gain. This can be explained by the fact that the difference between the GPU and vectorized code is essentially replacing NumPy operations with CuPy operations. And some operations are more efficient in NumPy than in CuPy. For example, cumulative sum in CuPy is slower than in NumPy because the nature of the operation is not parallelizable. However, the calculation of the analytical Fourier transform and the calculation of log error are much faster in CuPy than in NumPy. Thus, we still need to identify the parts of the code that are slowing down the GPU execution time due to communication overhead and replace them with NumPy operations. This is perhaps why we do not see a significant speedup in the execution time in figure 27. However, the code is now ready to be used on a GPU and can be further optimized. This will be the next step in the project.

In conclusion, parallelization and GPU acceleration have significantly enhanced the performance of our diffraction pattern simulation and fitting code. By leveraging the multipro-

cessing library and NumPy optimizations, we achieved considerable speedups in CPU-based parallelisation. Transitioning to GPU parallelization using CuPy further improved execution times, although some operations still perform better in NumPy due to their non-parallelizable nature. Future work will focus on identifying and optimizing these bottlenecks to fully harness the power of GPU acceleration. The groundwork laid here provides a robust foundation for further improvements and scalability in our computational simulations.

4.7 On-the-Fly Uncertainty Estimation

Another objective identified in our research was to utilize the code to estimate uncertainty in real-time during synchrotron experiments. The uncertainty in CD-SAXS experiments depends on various factors such as exposure time, sample thickness, detector distance, etc. Among these, exposure time plays a crucial role in determining uncertainty. The study by Sunday et al. [12] provides an excellent example of this relationship (see figure 28).

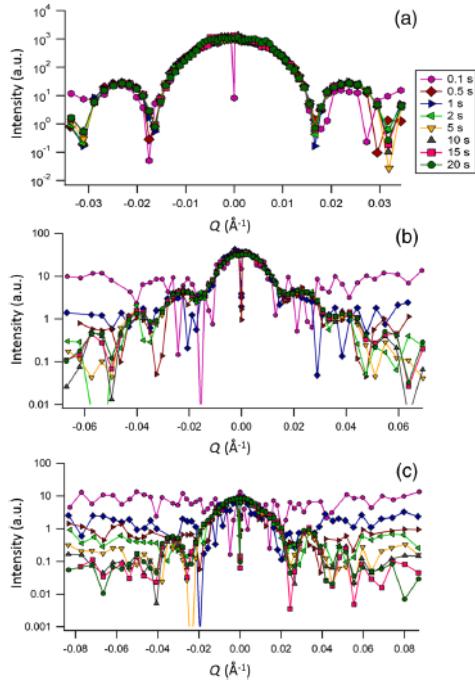


Figure 28: Uncertainty in fit parameters as a function of exposure time. The uncertainty decreases with increasing exposure time. (Sunday et al. [12])

Sunday et al. conducted measurements over a range of angles with specified step sizes for different acquisition times. Their results, shown in the figure 28, present representative data cuts for various order peaks, illustrating the impact of exposure time on data quality. The first-order peak, which exhibits the most intense scattering, remains well-defined even at

shorter acquisition times. However, as the measurement time decreases, the lower intensity features of the higher-order peaks become progressively obscured due to the decreasing signal-to-noise ratio.

If we could estimate uncertainty in real-time during the experiment, it would enable us to adjust various parameters including exposure time affecting uncertainty on the fly. Additionally, we could stop the experiment after measuring a sufficient number of angles. The same paper by Sunday et al. demonstrates that beyond a certain number of angles, the uncertainty does not change significantly. This approach could facilitate performing CD-SAXS experiments in laboratory settings with equipment less powerful than synchrotrons.

To investigate this, a simulation was conducted using the stacked trapezoid model. A two-stack trapezoid was considered with the following parameters: $h_1 = h_2 = 21.5$ nm, bottom CD = 24 nm, first sidewall angle = 85° , and second sidewall angle = 90° . The simulation aimed to mimic data acquisition at different angles with a measurement time of 2 seconds per angle. The uncertainty was calculated after every 15 angles measured. The results are shown in figure 29.

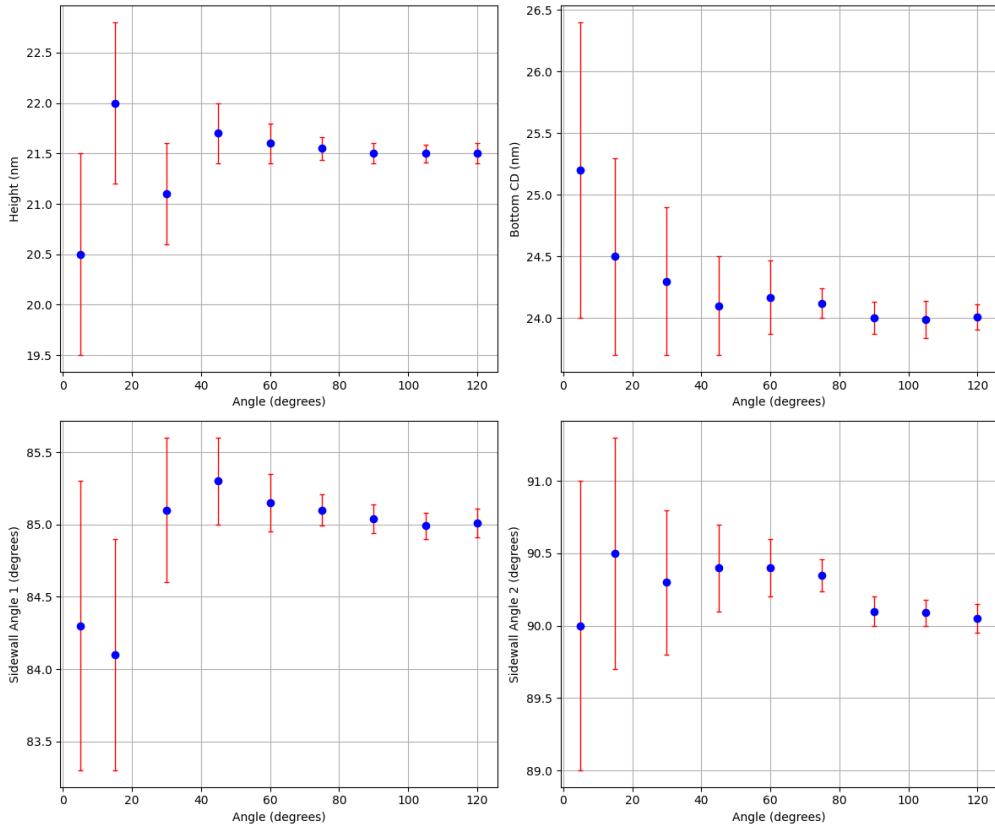


Figure 29: Uncertainty in the fit parameters as a function of the number of angles measured.

As seen in figure 29, the uncertainty levels off after a certain number of angles, similar to the findings by Sunday et al. However, in our simulation, the angle at which uncertainty

stabilizes is around 78 degrees, compared to 30 degrees in their study. This discrepancy could be due to differences in the measured structures and other factors. Nevertheless, this simulation demonstrates the feasibility of real-time uncertainty estimation during experiments, allowing for parameter adjustments as needed. Further testing with extensive real data is necessary to validate this approach, which is an ongoing effort.

5 Future Prospects

Leveraging the successes achieved during this work-study program, this section explores potential avenues for further developments.

- **Add new models:** Three models have been successfully implemented or are nearing completion. The next phase could involve integrating more complex models, such as those used in the semiconductor industry. Notably, a PhD student Timothee Choisinet at the CEA has been developing a new model for a new sets of samples developed at the CEA. This model merits exploration for potential integration with the current codebase.

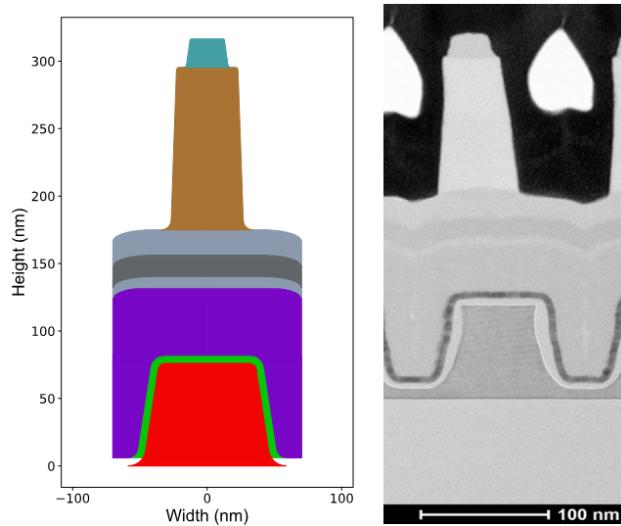


Figure 30: Nanostructure model being developed by Timothee Choisinet as a part of his PhD thesis. On the right is the TEM image of the sample and on the left is the model being developed.

- **Roughness along the line:** The current implementation focuses solely on the cross-sectional analysis of the surface. However, the diffraction patterns acquired through CD-SAXS inherently hold information regarding the roughness along the line of the sample (i.e. variation of profiles along the line). This aspect represents the next frontier for this project, and I will be exploring on it further within the context of my PhD thesis next year.

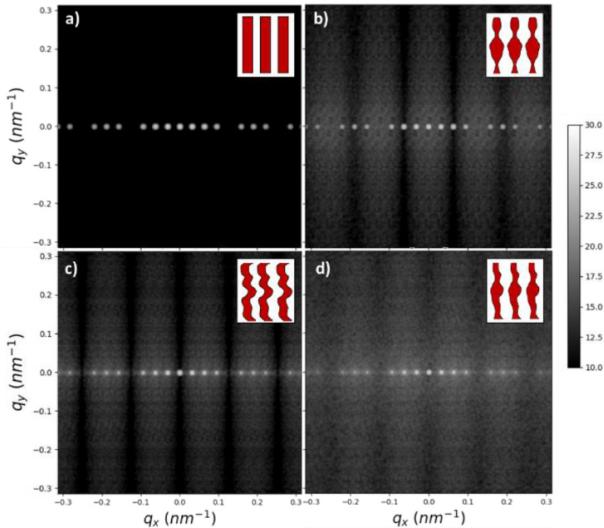


Figure 31: Simulations of diffraction patterns for various samples: a) Reference sample (no roughness), b) samples with correlated edges, c) samples with anti-correlated edges, and d) samples with uncorrelated edges.[18]

- **Make it open source:** The code is currently being developed in a private repository in github. We plan to make it open source in the future. This will allow other researchers to use the code and also contribute to it. We plan to publish the code on the Journal of Open Source Software (JOSS) once the code is cleaned up and documented properly.

6 Conclusion

This project successfully achieved its goals, resulting in a more robust and maintainable code-base for CD-SAXS data analysis. The redesigned structure will facilitate easier integration of new models in the future, allowing the tool to adapt to evolving research needs. Notably, the code now benefits from vectorization and GPU acceleration, significantly improving processing speed. Coupled with on-the-fly fitting and uncertainty estimation features, these enhancements will enable researchers to analyze data much faster, potentially even during experiments.

However, the project faced challenges. The initial lack of documentation and code structure required significant effort to understand both the code itself and the underlying physics behind. Additionally, the slow execution times in the project’s early stages hampered testing and new feature developments.

Despite these hurdles, the overall experience within the CEA team has been highly rewarding. The supportive environment fostered significant learning and growth, equipping me with valuable new skills that will undoubtedly benefit my future career.

References

- [1] Soham Chatterjee. *Beginner's Guide to Moore's Law*. URL: <https://medium.com/@csoham358/beginners-guide-to-moore-s-law-3e00dd8b5057> (visited on 07/01/2021).
- [2] G. F. Lorusso and D. C. Joy. "Experimental resolution measurement in critical dimension scanning electron microscope metrology". In: *Scanning* 25.4 (2003), pp. 175–180. DOI: <https://doi.org/10.1002/sca.4950250403>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/sca.4950250403>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sca.4950250403>.
- [3] Weidong Yang et al. "Line-profile and critical-dimension monitoring using a normal incidence optical CD metrology". In: *IEEE Transactions on Semiconductor Manufacturing* 17.4 (2004), pp. 564–572. DOI: [10.1109/TSM.2004.835728](https://doi.org/10.1109/TSM.2004.835728).
- [4] David Bergel. *How does a transistor work?* URL: <https://davidabergel.wordpress.com/2017/02/22/transistor/> (visited on 07/01/2021).
- [5] T. Baumbach, D. Lübbert, and M. Gailhanou. "Strain and Shape Analysis of Multilayer Surface Gratings by Coplanar and by Grazing-Incidence X-Ray Diffraction". In: (2000). DOI: [10.1063/1.372409](https://doi.org/10.1063/1.372409). URL: <https://publica.fraunhofer.de/handle/publica/197409>.
- [6] Daniel Sunday et al. "Determining the shape and periodicity of nanostructures using small-angle X-ray scattering". In: *Journal of Applied Crystallography* 48 (Oct. 2015). DOI: [10.1107/S1600576715013369](https://doi.org/10.1107/S1600576715013369).
- [7] Ronald Jones et al. "Pattern fidelity in nanoimprinted films using critical dimension small angle x-ray scattering". In: *Journal of Microlithography Microfabrication and Microsystems* 5 (Jan. 2006). DOI: [10.1117/1.2170550](https://doi.org/10.1117/1.2170550).
- [8] Guillaume Freychet. "Analyses morphologiques et dimensionnelles de nanostructures organisées par diffusion centrale des rayons X". Theses. Université Grenoble Alpes, Oct. 2016. URL: <https://theses.hal.science/tel-01612242>.
- [9] Guinier André et al. *Small-angle scattering of X-rays / André Guinier,... Gérard Fournet,... ; translation by Christopher B. Walker,... ; followed by a bibliography by Kenneth L. Yudowitch,... eng.* Structure of matter series. New York London: J. Wiley Chapman and Hall, 1955. URL: [https://www.eng.uc.edu/~beaucag/Classes/Scattering/Guinier%20and%20Fournet%20SAXS\(1955\).pdf](https://www.eng.uc.edu/~beaucag/Classes/Scattering/Guinier%20and%20Fournet%20SAXS(1955).pdf).

- [10] Jérôme Reche. “Nouvelle méthodologie hybride pour la mesure de rugosités sub-nanométriques”. Theses. Université Grenoble Alpes, Oct. 2019. URL: <https://theses.hal.science/tel-02520554>.
- [11] Adam F Hannon et al. “Advancing X-ray scattering metrology using inverse genetic algorithms”. In: *Journal of micro/nanolithography, MEMS, and MOEMS : JM3* 15.3 (2016), p. 034001. DOI: 10.1117/1.JMM.15.3.034001.
- [12] Daniel F Sunday et al. “Evaluation of the effect of data quality on the profile uncertainty of critical dimension small angle x-ray scattering”. In: *Journal of Micro/Nanolithography, MEMS, and MOEMS* 15.1 (2016), p. 014001. DOI: 10.1117/1.JMM.15.1.014001.
- [13] D. Foreman-Mackey et al. “emcee: The MCMC Hammer”. In: *PASP* 125 (2013), pp. 306–312. DOI: 10.1086/670067. eprint: 1202.3665.
- [14] “Energy and Temperature Dependencies for Electron-induced Sputtering from H₂O Ice: Implications for the Icy Galilean Moons”. In: *The Planetary Science Journal* 5 (June 2024), p. 146. DOI: 10.3847/PSJ/ad484d.
- [15] Jonathan Goodman and Jonathan Weare. “Ensemble samplers with affine invariance”. In: *Communications in Applied Mathematics and Computational Science* 5 (Jan. 2010). DOI: 10.2140/camcos.2010.5.65.
- [16] Massinissa Lounis et al. “GPU-based parallel computing of energy consumption in wireless sensor networks”. In: June 2015, pp. 290–295. DOI: 10.1109/EuCNC.2015.7194086.
- [17] CuPy Developers. *CuPy: NumPy-like API accelerated with CUDA*. Accessed: 2024-06-19. 2024. URL: <https://docs.cupy.dev/>.
- [18] Jérôme Reche et al. “Shape and Roughness Extraction of Line Gratings by Small Angle X-Ray Scattering: Statistics and Simulations”. In: *IEEE Transactions on Semiconductor Manufacturing* 35 (Aug. 2022), pp. 1–1. DOI: 10.1109/TSM.2022.3176026.