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

Contents

1 Introduction **4**

2 Context **5**
- 2.1 Host organisation . . . 5
  - 2.1.1 History . . . 5
  - 2.1.2 Sectors of activity . . . 6
  - 2.1.3 Future and oreintation strategy . . . 7
- 2.2 Project description . . . 8
- 2.3 Objectives . . . 8

3 CD-SAXS **10**
- 3.1 Introduction . . . 10
- 3.2 Scattering Model . . . 11
- 3.3 Experimental setup . . . 12
- 3.4 Fitting Algorithm . . . 12
  - 3.4.1 Covariance Matrix Adaptation Evolution Strategy (CMAES) . . . 13
  - 3.4.2 Uncertainity estimation by Monte Carlo Markov Chain(MCMC) method 15
  - 3.4.3 Overall view . . . 16

4 My contribution **18**
- 4.1 Introduction . . . 18
- 4.2 Design . . . 18
  - 4.2.1 Components . . . 18
  - 4.2.2 Relationships between components . . . 20
  - 4.2.3 Why this design? . . . 20
- 4.3 Simulation Models . . . 22
  - 4.3.1 Breif overview of photoresist lithography . . . 22
  - 4.3.2 Stacked Trapezoid Model . . . 23
  - 4.3.3 Rounded corners model . . . 24
  - 4.3.4 Overlay model . . . 25
- 4.4 Implementation of MCMC to Measure Uncertainty . . . 26
- 4.5 Parallelization and GPU Acceleration . . . 28
  - 4.5.1 Overview of Parallelization . . . 28
  - 4.5.2 Parallelization Strategies . . . 29
- 4.6 On the fly uncertainity estimation . . . 32

5 Future Prospects **33**

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 Microscope (CDSEM), are reaching their limits in terms of resolution and accuracy as feature sizes shrink to the nanometer scale.

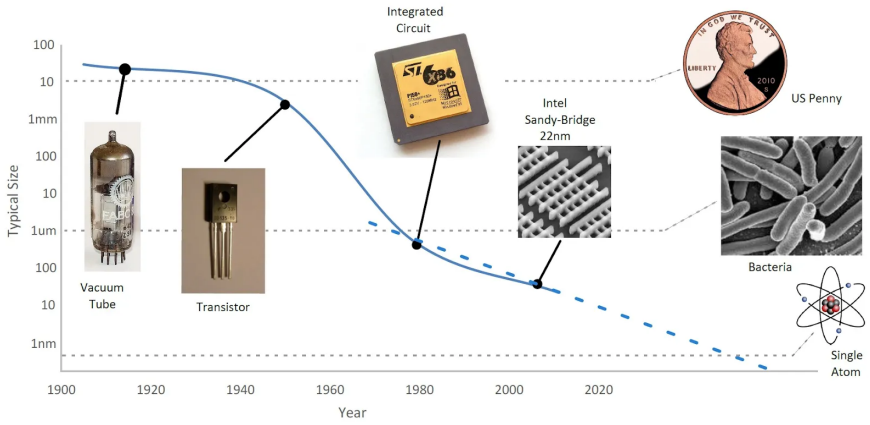


Figure 2: Evolution of microelectronics and the need for advanced metrology techniques [1].

To address these challenges, a new metrology technique called Critical Dimension Small Angle X-ray Scattering (CDSAXS) is being developed. CDSAXS utilizes short-wavelength X-rays ($\lambda \approx 0.05 - 5nm$), to probe the internal structure of materials, providing high-resolution measurements of critical dimensions (CDs) with greater accuracy than conventional methods. CEA-Leti, a leading research institute in microelectronics, is actively involved in the development of CDSAXS technology.

This work-study project focused on the development of a coherent software for the fit and analysis of CDSAXS data. The software aims to streamline the data processing workflow and enhance the accuracy of CD measurements. The project involved a comprehensive understanding of CDSAXS 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 CDSAXS 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's 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's 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 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 3: 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 didn't 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 diver-

sification 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's "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).

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's also the headquarters for CEA Tech, the technological branch of the CEA with over 4,500 researchers across France.

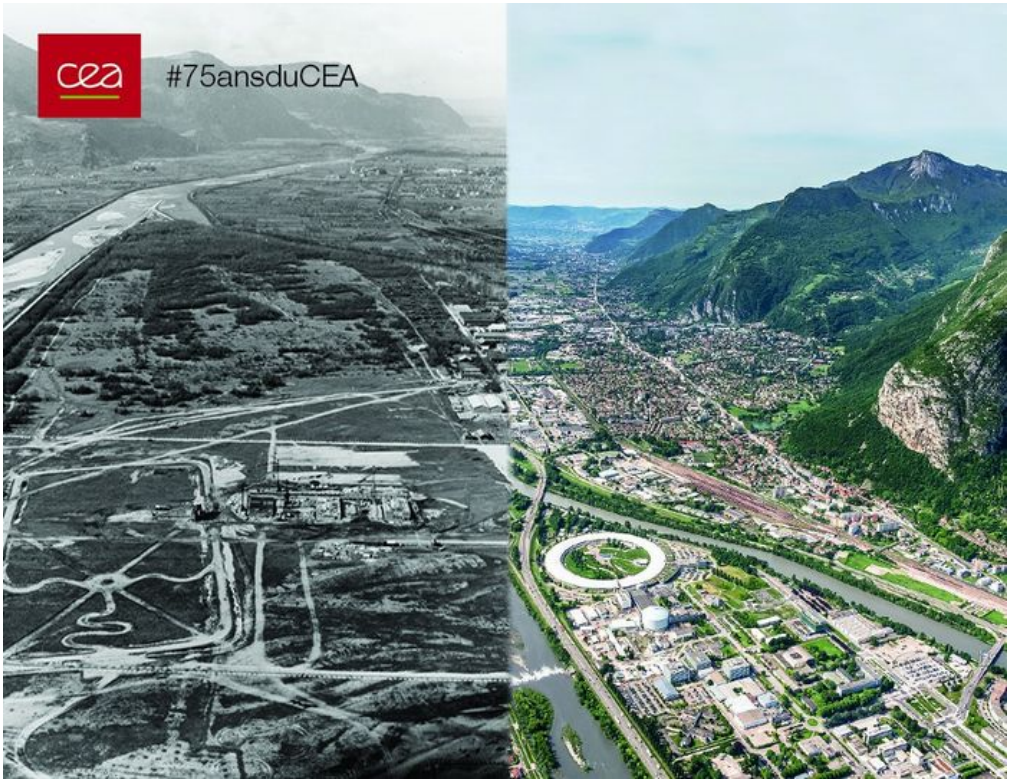


Figure 4: CEA Grenoble campus before and after

2.1.2 Sectors of activity

CEA Grenoble plays a pivotal role in the nation's 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 and thermonuclear fusion.
- **Digital Technologies**: Contributing through the SPIN program for spintronics (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)
- **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 oreintation 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 exemplifies this innovative spirit. They focus on cutting-edge technologies like AI, advanced materials, nanotechnologies, and quantum technologies. Their research aims to revolutionize industries and improve lives, from developing next-generation batteries to creating innovative medical devices and advancing digital technologies.

The challenges they tackle are vast: climate change, the energy transition, emerging diseases, cybersecurity, and technological sovereignty. The CEA goes beyond just solutions; they strive to influence public policy and raise awareness. Their research is guided by a long-term vision, anticipating future challenges and preparing for them through innovation.

The CEA is a vital force in shaping a sustainable future. Their commitment to innovation promises 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 positions itself as a leader 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 rudimentary collection of functions developed at Brookhaven 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 optimisation. To address this, I implemented optimisations and parallelisation techniques, significantly improving the code's 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's 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's 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's powerful servers. This significantly boosted the application's 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 same code base:** The original codebase had a only one simulation model, the stacked trapezoid model. We aimed to integrate overlay and rounded trapezoid model 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's 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 but is restricted by the sampling area. 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.

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-ray's sensitivity to features' shape and spacing. Furthermore, X-rays can probe buried features due to their sensitivity to composition via electron density. For instance, a GaInAs/InP multilayer was studied with high-resolution XRD, revealing sensitivity to both the grating and strain between layers [4].

For sub-100 nm features, small-angle scattering methods like GISAXS become more practical. GISAXS examines nanostructures across large areas, making it a potential metrology tool for semiconductors. 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.

However, GISAXS limitations include a large spot size (50-100 mm wide by 5-10 mm long) and computational challenges for complex nanostructure modeling. This limits its use for semiconductor metrology to simple, large-area patterns 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. [5]

Unlike other X-ray techniques, CDSAXS uses a transmission geometry, enabling a much smaller spot size compared to methods like GISAXS. This allows for more precise measurements on smaller features. Studies have shown CDSAXS's effectiveness in characterizing the shape and spacing of nanometer-sized patterns [6].

CDSAXS utilises 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 spacings around 30 nm, dimensions crucial for the semiconductor industry. [7]

3.2 Scattering Model

We can represent the diffraction of a collimated X-ray beam by:

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

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

This relationship is considered valid within the limitations of the CDSAXS geometry, which includes a transmission geometry and a low probability of multiple scattering. Unfortunately, the conjugate product in the equation leads to a loss of 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's 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 lines of nanostructures (see figure 6). The cross-section of these can be represented as a stack of trapezoids. We

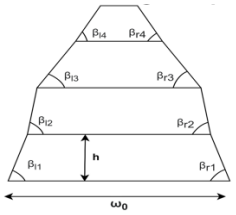


Figure 5: Cross-section of a nanostructure line represented as a stack of trapezoids.

can calculate the analytical fourier transform of a trapezoidal shape to use for fitting can be calculated is given by expression [6]:

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

where,

$\mathrm{m}_1 = \tan(\beta_1), m_2 = \tan(\pi - \beta_r)$,

$t_1 = q_x + m_1 q_z, t_2 = q_x + m_2 q_z$

so,

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

An additional decay of scattered intensity $I(Q_x)$ is expected beyond that predicted by the trapezoidal model. This arises from the distribution of periodicities within the sample. This distribution can be caused by two factors:

- **Random variations in average line position**: In this case, the line width remains constant, but the average position of the lines fluctuates slightly across the sample.
- **Variations in line width**: Here, the line width itself varies, which also affects the periodicity.

Both factors indicate a degree of long-range order within the pattern. Additionally, they provide insights into specific types of line edge roughness [8].

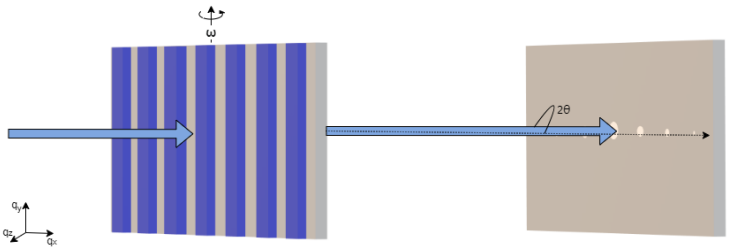
To account for this distribution, we introduce an effective Debye-Waller factor, similar to the one used for fluctuations in crystal lattices.

Hence,

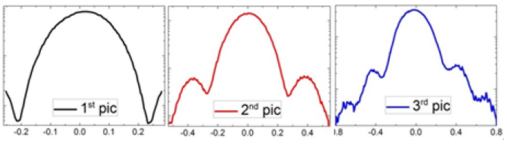
$$I(\mathbf{q}) = I_0(\mathbf{q}) \exp(-q^2 DW^2) \tag{4}$$

where DW is the Debye-Waller factor.

3.3 Experimental setup



(a) experimental setup



(b) Intensity vs Q_z

Figure 6: In a is the schematic of the CD-SAXS instrument layout. The X-ray beam (thick solid lines) is transmitted through the patterned sample. Scattered intensity (I) is measured by a 2D detector as a function of scattering angle (2θ) and converted to $I(Q_x)$, where Q_x is defined in the text. Measurements are performed at various sample incidence angles (θ'). In b After conversion from the (Q_x, ω) plane, the intensities from a trapezoidal cross section would appear as predicted in the model calculation on b, plotted as I as a function of the Fourier component, Q_Z.

3.4 Fitting Algorithm

While CDSAXS 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 structure's characteristics.

However, there's a catch. Traditional optimization methods used for refinement often fall short when dealing with complex internal structures with numerous parameters. These methods rely on iteratively simulating scattering data and comparing it to the measured data. Unfortunately, this approach can be very time-consuming, especially for intricate structures.

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 CDSAXS 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 isn't 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 measured CDSAXS data. These parameters essentially describe the actual structure of 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's parameters and limitations on their search range. Additionally, it necessitates multiple independent runs to ensure 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. [9]

3.4.1 Covariance Matrix Adaptation Evolution Strategy (CMAES)

One such 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's performance. By adapting the covariance matrix of the candidate solutions, CMAES can efficiently explore the parameter space and converge on the optimal solution.

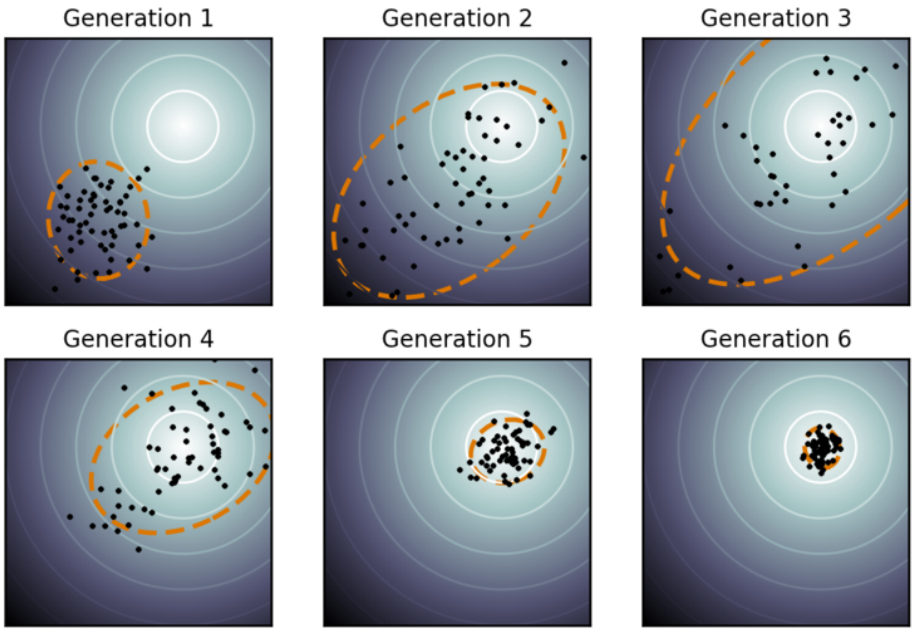


Figure 7: Illustraion of CMAES algorithm. The algorithm maintains a population of candidate solutions, with each iteration generating new candidates based on the previous generation's performance.(image taken from wikipedia CMAES page)

The code for CD-SAXS generates a set of possible solutions for the parametres of the model. Then we calculate the analytical fourier transform of the model and compare it with the experimental data. To compare the two, we use mean-absolute error log:

$$\Xi = \frac{1}{N_{\mathrm{q}} - 1} \sum_{\mathbf{q}} |\log_{10} I_{\mathrm{Sim}}(\mathbf{q}) - \log_{10} I(\mathbf{q})| \tag{5}$$

where $I_{\mathrm{Sim}}(\mathbf{q})$ is the simulated intensity and $I(\mathbf{q})$ is the experimental intensity.

We call it the goodness of fit. The algorithm then tries to minimize this quantity by adjusting the parameters of the model.

In an article [9], 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 achieved 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 Uncertainity estimation by Monte Carlo Markov Chain(MCMC) method

The CMAES algorithm provides a single best-fit solution for the nanostructure parameters. However, it's essential to understand the uncertainty associated with these parameters. This uncertainity 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 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.

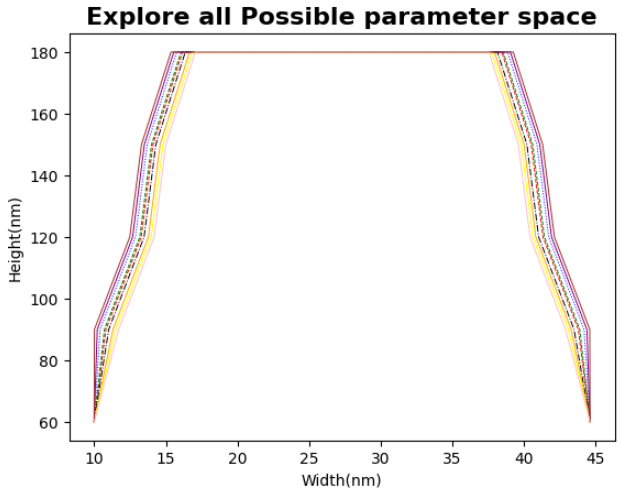


Figure 8: MCMC algorithm exploring all possible sets of parameters that can result in the same goodness of fit.

After determining the best-fit model structure, the researchers of this article [10] 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 there 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 $\mathrm{GF}_i < \mathrm{GF}_{i-1}$, then M_i is automatically accepted into the population (and GFB is updated to $\mathrm{GFB} = \mathrm{GF}_i$ if the new model has a better fit).

4. **Metropolis-Hastings acceptance for worse fit:** If $\mathrm{GF}_i > \mathrm{GF}_{i-1}$, the probability ($\mathrm{P}_i$) of accepting M_i is calculated using Eq. (6):

$$P_i = \exp\left(-0.5 \cdot (\mathrm{GF}_i - \mathrm{GFB})\right) \tag{6}$$

   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 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 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

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 frequency domain using a Fourier Transform, and this transformed data is compared with the experimental data to optimize the fit by adjusting the parameters and reducing the error between the simulated and experimental data.

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:

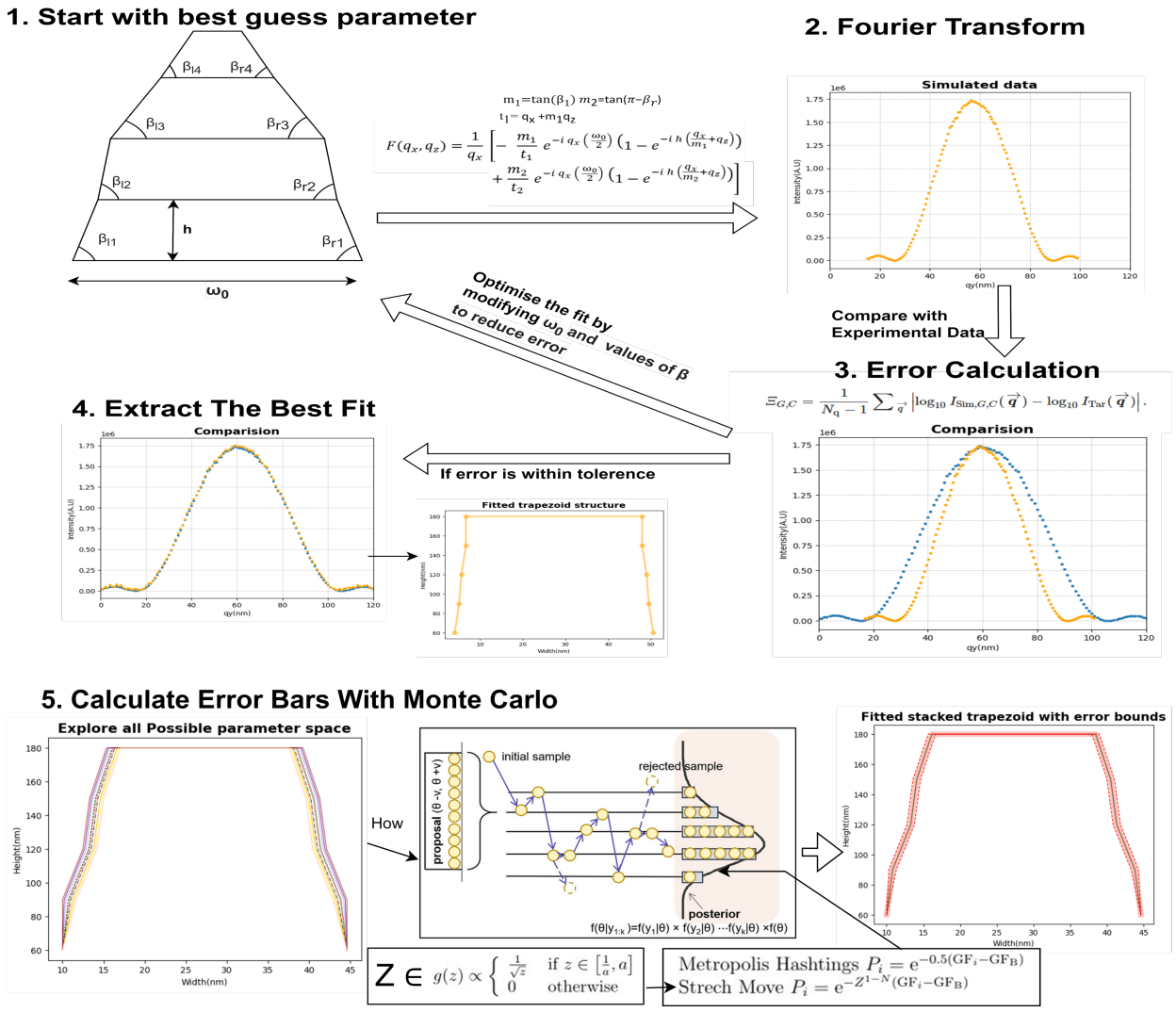


Figure 9: This algorithm integrates CMAES for initial parameter optimization and MCMC for uncertainty quantification. By combining these methods, the algorithm provides a robust approach to model fitting, ensuring both optimal parameter estimation and reliable uncertainty analysis.

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 utilised to estimate uncertainty in real-time during synchrotron experiments.

4.2 Design

To gain a clear understanding of how this system works, let's visualize the design with the help of a UML diagram (see Figure 10). This diagram offers a roadmap, highlighting the various components and their interactions. We'll then delve deeper to explore each component's role in the system.

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's *simulate_diffraction* function to calculate the model diffraction pattern and compare it with the experimental data.

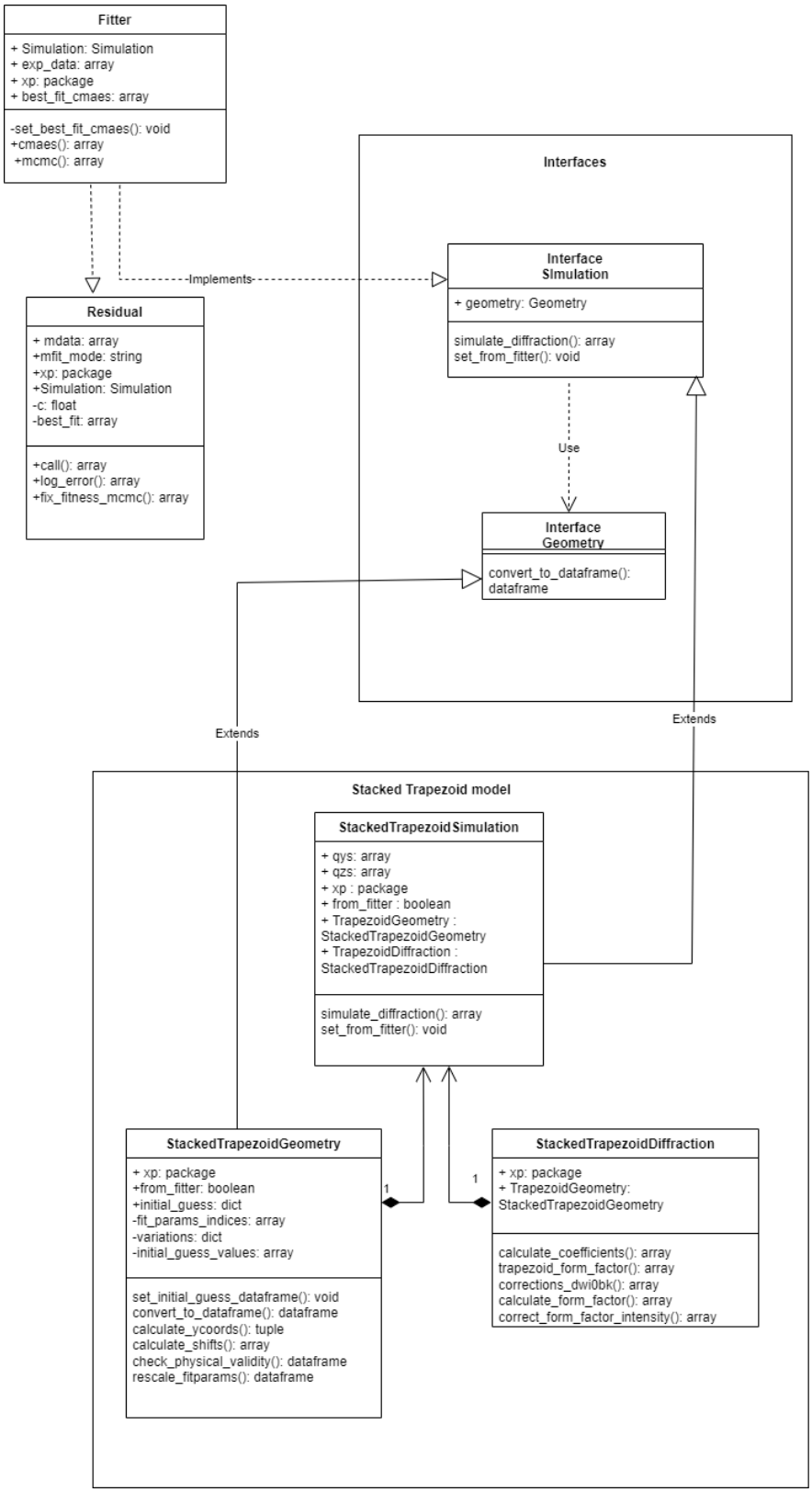


Figure 10: UML diagram of the design for CD-SAXS simulation application.

Simulation interface:

This is the base class for all simulation models. The functions and classes used in it should be implemented in all simulation models.

Geometry interface:

Like Simulation interface. This is the base class for all geometry models.

Model:

In this UML diagram (Figure 10), 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 they 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 system 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, utilising the *Residual* class to compute the difference between the experimental data and the model's simulated data. The *Residual* class leverages the model's *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?

Let's 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.

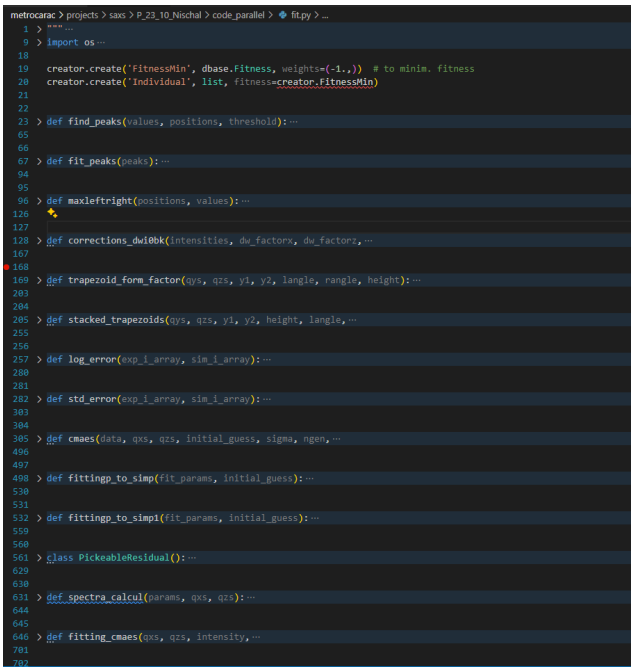


Figure 11: Monolithic design of the previous version. This script combined all the functions for the stacked trapezoid model in one file.

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's 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 for improved performance.

Similarly the codes for the different models each implemented 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

connecter 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 concerning themselves with the fitting process. This design choice significantly enhances the system's flexibility and scalability.

The decoupling of components also facilitates optimization and parallelization by allowing developers to focus on specific areas of improvement without 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 Simulation Models

In the realm of nano scale fabrication, techniques like photoresist lithography stands out as a pivotal technique. 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.

Our models aims 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 trapezoids, 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 introduce the overlay model, which simulates the alignment of different layers on a wafer.

4.3.1 Breif overview of photoresist lithography

These models are build to simulate the simple structure obtained through techniques like photoresist lithography. It is a technique used to fabricate micro and nano scale 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.

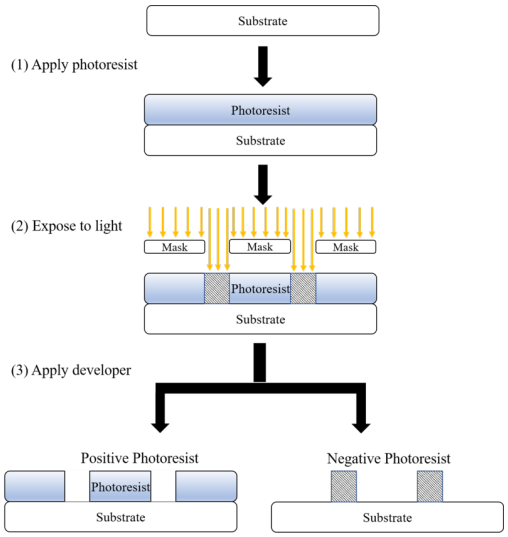


Figure 12: Schematic of the photoresist lithography process (taken from wikipedia).

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.

While developing our model, we consider lines of nanostructures on a substrate (see figure 13) obtained through this process.

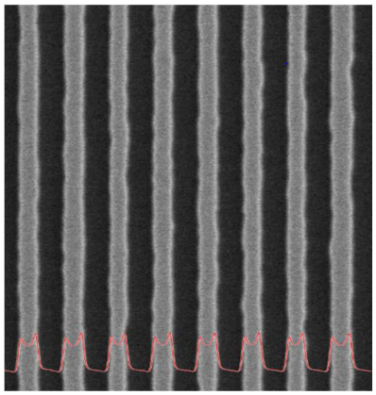


Figure 13: Image obtained with a scanning electron microscope in top view of the structure for modelling. In red, the average signal over the length of the lines is superimposed.[8]

4.3.2 Stacked Trapezoid Model

A close examination of the cross-section of the lines in Figure 13 reveals that their shapes can be closely approximated by stacked trapezoids. This crucial observation forms the foundation of our modeling approach.

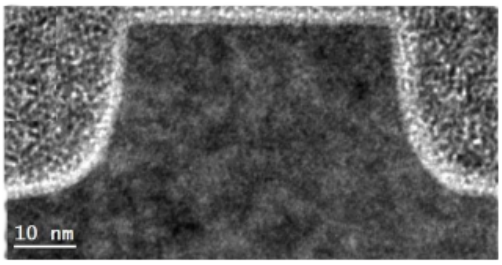


Figure 14: Tunneling electron microscope image of the cross section of the lines.[5]

Figure **??** illustrates the approximation of the model we aim to simulate. The model inputs include a constant height or sets of heights, sidewall angles (β), and the bottom width 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:

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

As discussed in Section 3.2, 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.

4.3.3 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 14). 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.

While this model hasn't been implemented yet, a similar codebase already exists. This existing code provides a strong foundation, potentially making implementation a smoother process.

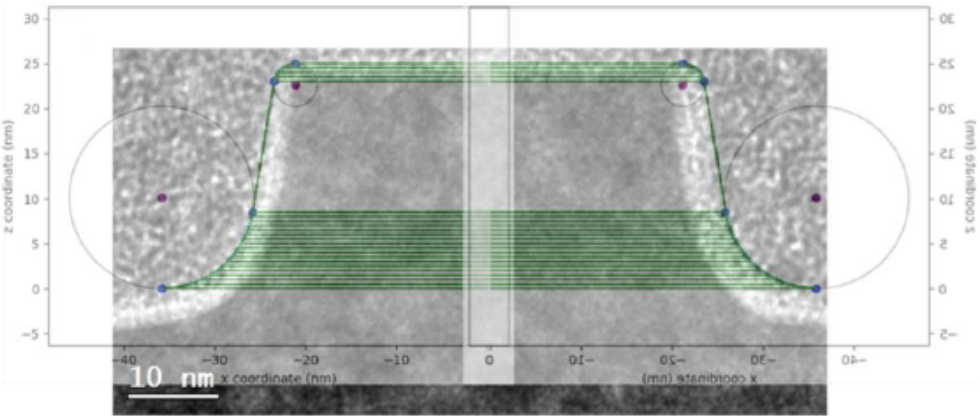


Figure 15: Rounded corners model fitted on top of the scanning electron microscope image.

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.

4.3.4 Overlay model

Silicon wafers are built layer-by-layer through a series of steps called photolithography. 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.

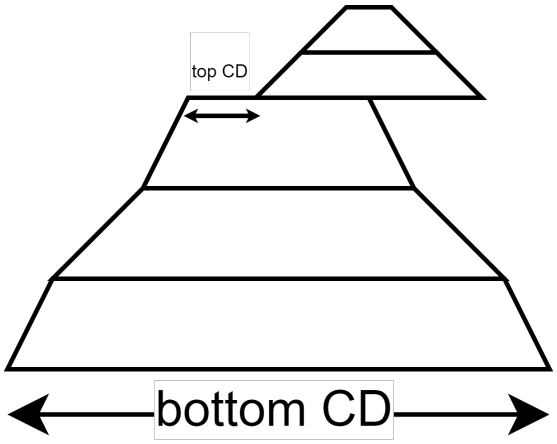


Figure 16: Overlay model.

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 extends it to capture misalignment effects. It incorporates two sets of trapezoids: one set for each layer. The key difference lies in the positioning of the second layer's trapezoids. These are shifted by a specific amount, called the Top Critical Dimension (Top CD), in the x-direction. This shift represents potential misalignment between layers.

To calculate the coordinates of the second layer's 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 Top CD 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 is fitted later on.

4.4 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 in the fitted parameters. 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's implementation and its outputs.

We utilize the *emcee* Python package [11] 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 [12]. 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, 11]. The stretch move uses the following formula to determine whether to accept or reject a move:

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

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:

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

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 [10], 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:

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

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	uncertainity
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
langle1	1,541163	0,049045	62370	1,248772	2,23387	1,540657	1,541669	0,000506
langle2	1,498193	0,042529	62370	0,744646	2,297001	1,497754	1,498632	0,000439
rangle1	1,612237	0,046393	62370	1,237489	2,198178	1,611759	1,612716	0,000479
rangle2	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 17: 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.

4.5 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 parallelization. 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 overview the strategies for parallelization on both GPU and CPU.

4.5.1 Overview of Parallelization

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 achieve a significant speedup. However, there is a catch: the GPU is not as flexible as the CPU. It is designed for parallel processing of large amounts of data, making it ideal for tasks like matrix multiplication. However, 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 GPU's strengths.

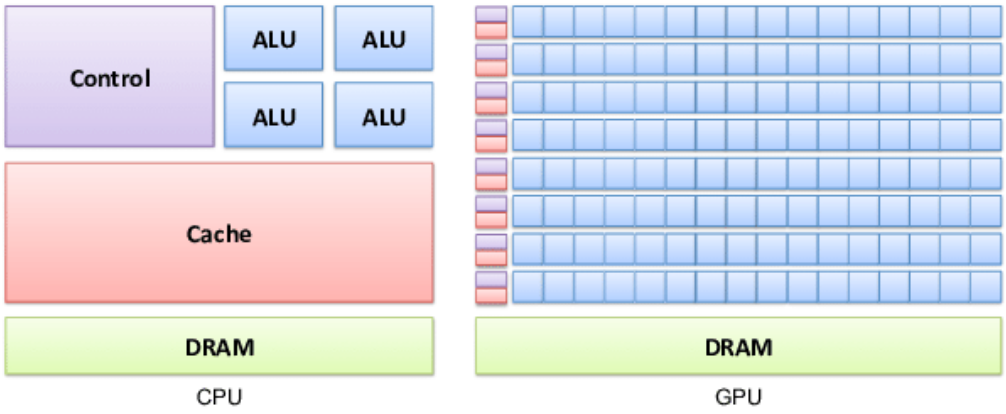


Figure 18: Comparison between GPU and CPU. The GPU has more cores but is less flexible than the CPU, making it suitable for simpler parallel tasks. [14]

4.5.2 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 to calculate 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 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 could be evaluated simultaneously.

However, the code still contained many for-loops that could be parallelized using NumPy operations. NumPy, being a highly optimized library for numerical operations, was used to further optimize the code. Wherever for-loops could be replaced by NumPy operations, we did so, further reducing computation time. The major challenge was that the quantities we wanted to perform operations on were not of the same shape, requiring the use of NumPy broadcasting to perform the operations (see Figure 20 for an example).

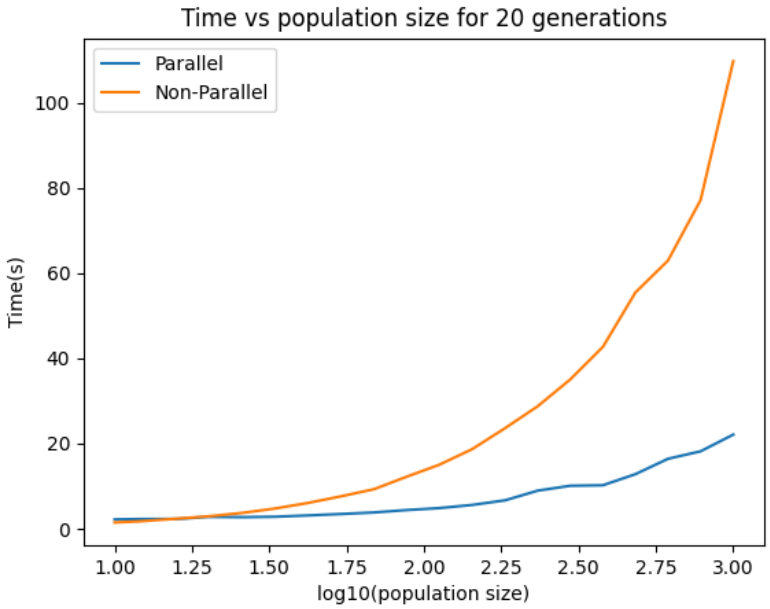


Figure 19: Multiprocessing used to improve the execution time of the code for the *cmaes* algorithm. The generation was fixed to 20.

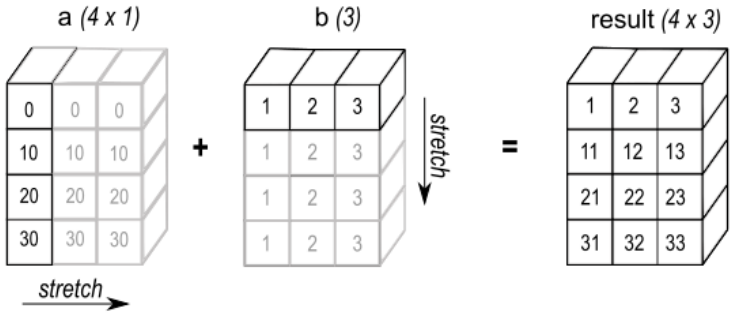


Figure 20: NumPy broadcasting used to perform operations on arrays of different shapes (image taken from NumPy documentation).

After this step, 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 [**cupy**]).

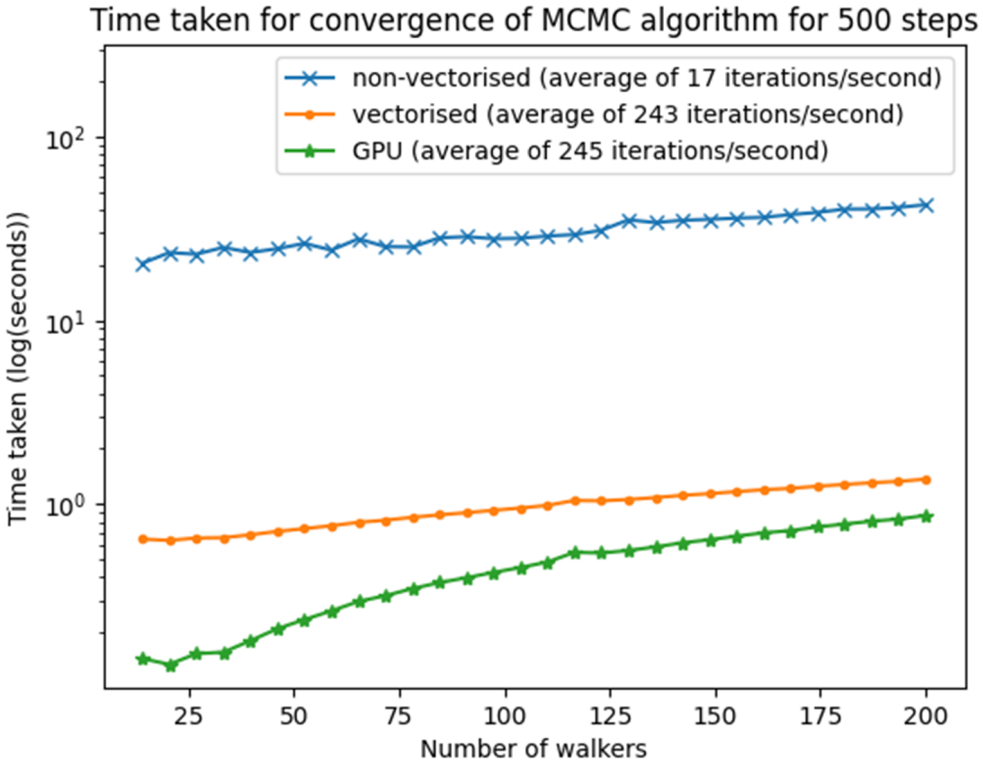


Figure 21: CuPy used to parallelize the code. The number of iterations per second in the parallel version is much higher. (This test was performed on CEA aar164 server Intel(R) Xeon(R) Platinum 8362 CPU @ 2.80GHz, 128 cores, and NVIDIA A100 80GB GPU)

However, we are essentially replacing NumPy operations with CuPy operations, 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 execution time due to communication overhead and replace them with NumPy operations. This is perhaps why we don't see a significant speedup in the execution time in Figure 21. 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 multiprocessing library and NumPy optimizations, we achieved considerable speedups in CPU-based parallelization. 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.

4.6 On the fly uncertainity estimation

5 Future Prospects

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.

[4] 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. URL: https://publica.fraunhofer.de/handle/publica/197409.

[5] 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.

[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.

[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.

[8] 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.

[9] 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.

[10] 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.

[11] D. Foreman-Mackey et al. "emcee: The MCMC Hammer". In: *PASP* 125 (2013), pp. 306–312. DOI: 10.1086/670067. eprint: 1202.3665.

[12] "Energy and Temperature Dependencies for Electron-induced Sputtering from H 2 O Ice: Implications for the Icy Galilean Moons". In: *The Planetary Science Journal* 5 (June 2024), p. 146. DOI: 10.3847/PSJ/ad484d.

[13] 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.

[14] 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.