Simulation of focused ion beam sputtering using static and dynamic models

Anindit Das
2011021

Abstract

Simulating physical experimental techniques enables researchers to comprehensively understand complex processes and phenomena, fostering innovation and optimization in various fields. Ion Beam Sputtering, a versatile and powerful technique used for thin film deposition, exemplifies the need for accurate simulation methodologies. By replicating the intricate interplay of ions with target materials at the atomic level, simulations facilitate the exploration of fundamental mechanisms underlying sputtering dynamics. Simulating IBS is important in its ability to predict and precisely control thin film properties, offering valuable insights into material behavior under diverse conditions. Such predictive capabilities empower researchers to tailor thin films for specific applications, ranging from optical coatings to semiconductor devices, with enhanced efficiency and efficacy. In the term project for the computational physics course, I tried to simulate the process of ion beam sputtering using static and dynamic models to the best of my ability. Although the models are lacking in many aspects and utilize many approximations, they provide valuable insights into the mechanisms of occurrence of the sputtering processes.

1 Objectives

1. Simulate the various processes and measurements of ion beam sputtering processes.

2. Simulate the mechanism of sputtering and particle interactions using static and dynamic models like the Binary Collision model and Molecular dynamics Monte Carlo method.

2 Introduction

Sputtering is a physical process in which atoms or molecules are ejected from a solid surface due to bombardment by energetic particles, such as ions or electrons. This phenomenon typically occurs when energetic particles collide with the surface of a material, transferring energy to surface atoms and causing them to be expelled. Sputtering can result in the removal of atoms or molecules from the surface of a solid material, leading to the formation of a thin film on nearby surfaces or the erosion of the target material itself. Sputtering is commonly utilized in various applications, including thin film deposition, surface analysis, and etching in semiconductor manufacturing and materials science.

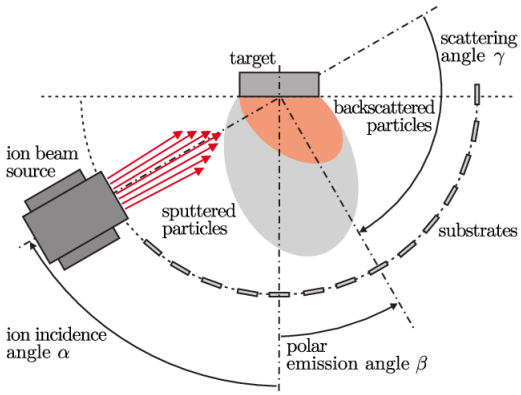

Figure 1: Schematic experimental setup of ion beam sputtering.

2.1 Mechanism

Ion Beam Sputtering (IBS) is a physical vapor deposition (PVD) technique utilized for thin film deposition. The process unfolds as follows:

1. **Ion Generation**: In a plasma chamber, a gas like argon is ionized using RF or DC discharge methods, generating a plasma containing positively charged ions.

2. **Ion Acceleration**: The ions are accelerated to high velocities through electrodes or magnetic

fields, forming a focused beam aimed at the target material.

3. **Target Bombardment**: Accelerated ions collide with the target material's surface, transferring kinetic energy to surface atoms and causing their ejection through a process known as sputtering.

4. **Material Ejection**: Atoms from the target material are ejected into the vacuum chamber, dispersing in various directions and potentially depositing onto nearby surfaces, including substrates for thin film deposition.

5. **Thin Film Deposition**: The ejected atoms accumulate on the substrate surface, adhering through mechanisms like physical adsorption, chemical bonding, or diffusion. Thin film properties such as thickness, composition, and microstructure are controllable by adjusting parameters like ion energy, flux, target material, and substrate temperature.

6. **Process Control and Optimization**: Parameters such as ion energy, flux, target material, temperature, and deposition rate are fine-tuned to achieve desired thin film properties, enabling customization for specific applications.

2.2 Calculation of Sputtering Yield and Backscattering

The sputtering yield (Y) is a crucial parameter in ion beam sputtering, representing the average number of atoms sputtered per incident ion. The Yamamura formula is a widely used empirical expression for calculating the sputtering yield, given by:

$$Y = \frac{a}{1 + \sqrt{bE}} \tag{1}$$

where a and b are material-dependent constants, and E is the kinetic energy of the incident ions.

Backscattering, a phenomenon where incident ions are reflected back from the target surface without causing sputtering, can be mathematically described using the Rutherford scattering formula:

$$\frac{d\sigma}{d\Omega} = \left(\frac{Z_1 Z_2 e^2}{16E}\right)^2 \frac{1}{\sin^4\left(\frac{\theta}{2}\right)} \tag{2}$$

where $\frac{d\sigma}{d\Omega}$ is the differential cross-section for scattering into a solid angle $d\Omega$ around the angle θ, Z_1 and Z_2 are the atomic numbers of the incident ion and target atom respectively, e is the elementary charge, and E is the kinetic energy of the incident ions.

2.3 Numerical Calculation of Sputtering Yield

The sputtering yield (Y) can also be calculated numerically using collision cascade simulations and considering the target material's surface binding energy (E_b). In such simulations, energetic ions collide with the target material, creating a cascade of atomic collisions resulting in atom ejection from the surface. The sputtering yield can be approximated by the ratio of the number of atoms sputtered (N_s) to the number of incident ions (N_i):

$$Y = \frac{N_s}{N_i} \tag{3}$$

To determine N_s, one needs to track the fate of each atom in the collision cascade. Atoms with sufficient kinetic energy to overcome the surface binding energy (E_b) will be ejected from the surface and contribute to N_s. The sputtering yield can then be calculated as the ratio of ejected atoms to incident ions.

3 Models

3.1 Static model

Please visit https://github.com/slashgeaus/Computational-physics/blob/main/Term%20Paper/Main/Static%20model(Working)(Binary%20collison%20approx).ipynb for code for this model.

This is a binary collision model; the binary collision method considers the sputtering process as a series of individual collisions between incident particles and atoms in the target material. Each collision is a binary interaction involving a projectile ion and a target atom. During the collision, energy and momentum are exchanged between the incident particle and the target atom, leading to various outcomes such as elastic scattering, inelastic scattering, or sputtering.

3.1.1 Energy Distributions using Monte Carlo method

The code uses a Monte Carlo method to generate energy distribution for Titanium atoms in ion beam sputtering simulations. It involves the following steps:

- **Number of Points (points):** Specifies the number of data points to be generated for the energy distribution, determining the granularity.

- **Energy Vector (energy):** An array of equally spaced energy values ranging from 0 to 35,
- **Function Evaluation (*func_energy*):** The distribution function *func_energy* is evaluated using a mathematical expression, $\frac{100 \cdot \text{energy}}{(\text{energy}+4)^3}$.
- **Random Variables** Two sets of random variables (*random_1* and *random_2*) are generated using `np.random.uniform()`, used to scatter points

A plot is created to visualize the intensity of energy levels across the specified range.

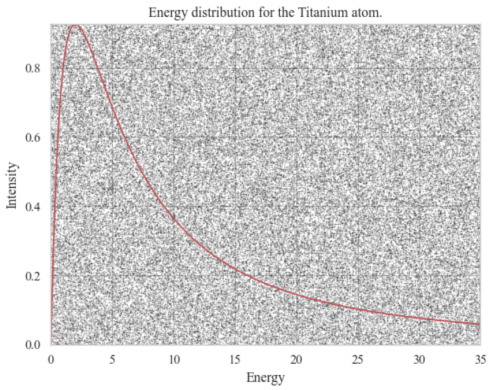

Figure 2: Energy distribution of Titanium ions.

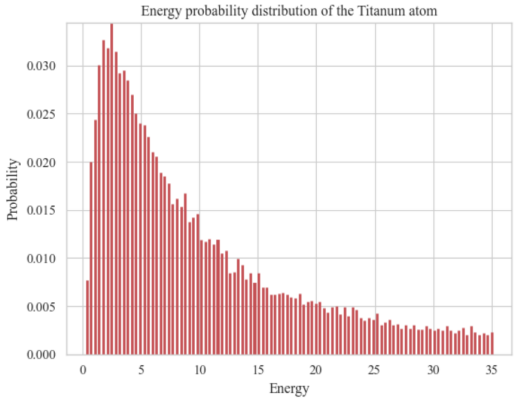

Figure 3: Energy Probability Distribution for Titanium Ions

3.1.2 Sputter Yield and collision cascade

A function is written that computes the sputter yield based on given parameters. It first generates random angles within the specified range. Then, it calculates the sputter yield for each particle using the given formula, considering the mass and energy of the ions, the surface binding energy of the target atoms, and the specified exponent. Finally, it returns the mean sputter yield across all particles.
The parameters required for sputter yield calculation include ion and target masses, ion energy, surface binding energy, particle count, angle range, optional exponent, lattice dimensions, initial ion energy, collision threshold, and energy transfer fraction.

A function is written that simulates a collision cascade within a lattice due to ion bombardment. It initializes a lattice grid and iterates through a specified number of ions. For each ion, it tracks its movement within the lattice, transferring energy to adjacent atoms upon collision. If the transferred energy surpasses a threshold, representing the surface binding energy, the impacted atom is marked as sputtered. This process is repeated for all ions, resulting in a lattice representation of the energy distribution after the collision cascade. The function provides insights into the material erosion process during ion beam interactions.

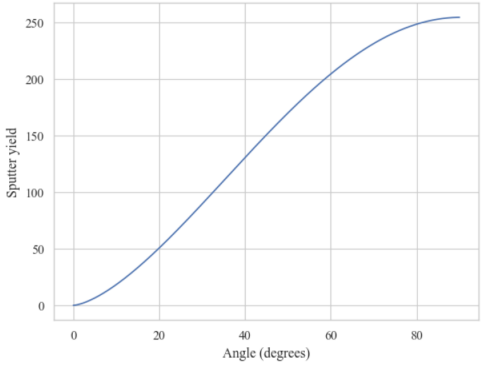

Figure 4: Sputter Yield vs Angle

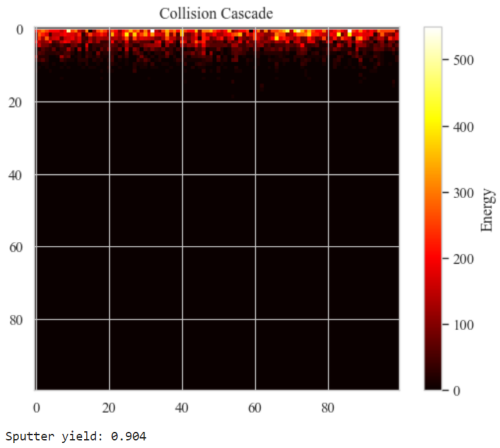

Figure 5: Collision cascade at the target surface.

3.1.3 Backscattering of Ions

A code that simulates the backscattering of ions interacting with a solid surface was written. The steps that occur for the calculations are the following:

- **Initialization**: The function `simulate_backscattering` takes three parameters: `num_ions` (number of ions), `initial_energy` (initial energy of the ions), and `surface_binding_energy` (surface binding energy).
- **Simulation Loop**: The function simulates the interaction of each ion with the surface by iterating `num_ions` times. For each iteration:
  - A random initial direction (angle) of the ion is chosen within the $[0, \pi]$ radians range.
  - The energy loss of the ion due to interaction with the surface is calculated by subtracting the `surface_binding_energy` from the `initial_energy`.
  - If the energy loss is greater than zero, indicating that the ion undergoes backscattering, the final backscattering angle is calculated as $\pi - \theta$, where θ is the initial angle.
  - The calculated backscattering angle is appended to the list `backscattering_angles`.

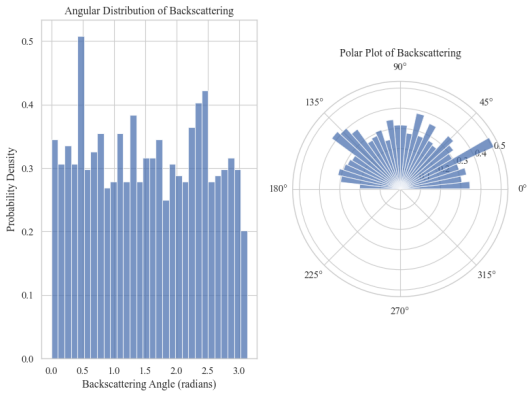

Figure 6: A histogram showing the angular distribution of backscattering, and A polar plot representing the same angular distribution.

3.1.4 Monte Carlo for sputtered particle trajectories

The code simulates the trajectory of titanium ions (Ti) as they travel and collide with each other:

The simulation begins by initializing parameters such as the number of titanium ions (Ti), the dimensions of the simulation environment (target, substrate, and camera), and lists to store results.

Each titanium ion is created and its trajectory is simulated. At each step, the ion's position is updated based on its direction and potential collisions with the surface.

Collision detection is performed probabilistically at each iteration. If a collision occurs, the ion's trajectory is modified to simulate the scattering effect.

After simulating the ion trajectories, the code evaluates whether each ion reaches the substrate. If an ion reaches the substrate, it is considered a successful hit; otherwise, it is a miss.

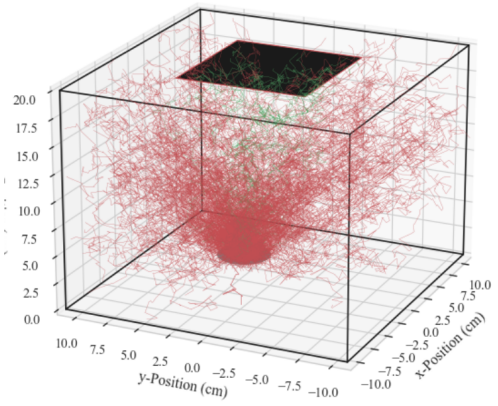

Figure 7: The trajectoreis of Ti ions as they are sputtered from the surface and travel towards the substrate(Black). The succesful hits are coloured **Green**, while the unsuccessful ones are coloured **Red**.

3.2 Dynamic model

Please visit https://github.com/slashgeaus/Computational-physics/blob/main/Term%20Paper/Main/Dynamic%20model(not%20working).ipynb for code for this model.
I was not successful in implementing this model, here I provide the road map and pseudocode for my implementation and possible reasons why the code did not work as intended.

3.2.1 Roadmap for implementation

Initialization

```
def initialize_system:
```

Defines a function to generate initial positions of metal atoms in the lattice based on lattice type and lattice constant, and defines free ions in random places in the remaining free space.

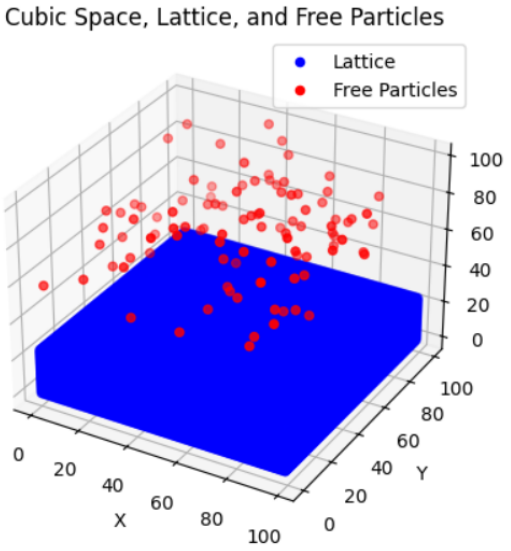

Figure 8: The total space along with all the components needed for simulation.

`def initialize_velocities:`

Initializes velocities of each of the free ions using the a gaussian distribution and the formula $<E> = \frac{3}{2}kT$

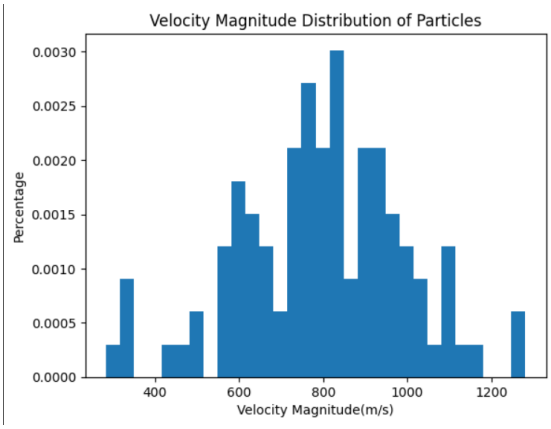

Figure 9: Velocity distribution of all the free particle (ion gas) due to temperatue.(300k)

`def binary collision:`

This will be used to simulate the collison cascade between surface and ion

`def hamiltonian:`
`def metropolis hastings:`

The hamiltonian will be used to simulate the interactions between ions and metropolis would be used to evolve the particle configurations

`def simulation loop:`

This runs for specified number of loops in specified step sizes. Each iteration will run a single step of the metropolis hastings and then calls one step of binary collison to evolve any ions that have reached the surface of the lattice.
The number of collisions which have more energy than the surface binding energy of the lattice will initiate sputtered atoms.

3.3 Implementation using RustBCA external library

Please visit https://github.com/slashgeaus/Computational-physics/blob/main/Term%20Paper/Main(Ext)/Static%20Model-%20sputtering(RustBCA%20model).ipynb for code for this model.

RustBCA is a free, open-source code designed to simulate how ions interact with materials. It stands for "Rust Binary Collision Approximation" and is written in the Rust programming language, with python wrappers so that the function scan be used in python codes.

Using this external Library I modeled the ion sputtering of various ions on Ni substrate and compared the results with other popular molecular simulation programs.

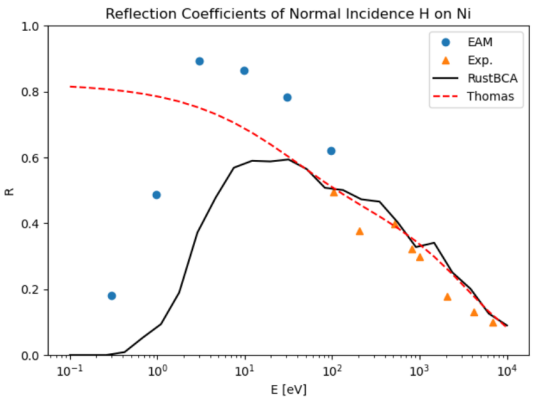

Figure 10: Comparison of RustBCA with experimental data(orange) and other models.

Here we modeled the sputtered yield of Xe on Si substrate: As we can see the sputter yield is quite

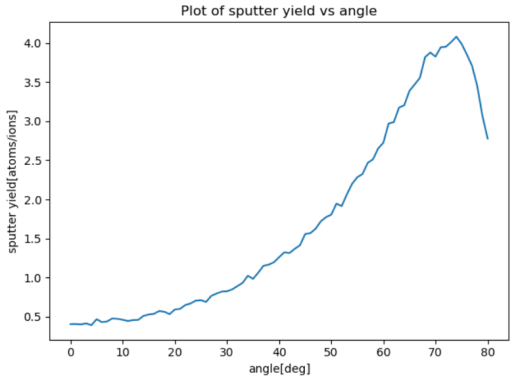

Figure 11: Sputtering angle vs Yield.

similar to the sputter yield curve generated by my code in **Fig. 4**.

The part where RustBCA is better than my **Static model** is the implanted depth calculation and ion trajectory after implantation in the target.

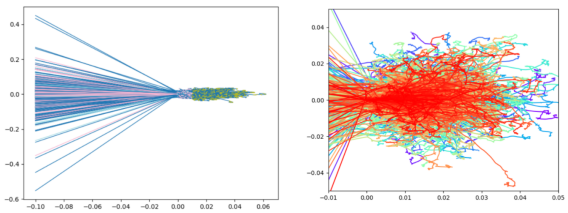

Figure 12: implantation depth and trajectories after implantation.

3.3.1 Advantage of RustBCA:

High-Performance: RustBCA is quite fast, Since it's built with a resource efficient language: RUST.
Accuracy: It offers a variety of algorithms for calculating scattering, providing more accurate results than some other models.
User-Friendly: The code is designed to be easy to use, with clear error messages and user-friendly input/output formats.

4 Acknowledgements

I would like to express my sincere gratitude to Dr. Subhasish Basak, our course instructor, for providing me with the opportunity to work on this project. Their guidance and support throughout the course have been invaluable.

I am also deeply grateful to Dr. Pratap K. Sahoo, my project advisor, for their continuous encouragement, insightful feedback, and expertise in guiding me through this project. Their mentorship has been instrumental in shaping my understanding and approach.

References

1. Eckstein, W. (2013). *Computer simulation of ion-solid interactions* (Vol. 10). Springer Science & Business Media.

2. Drobny, J. G., et al. (2021). RustBCA: A High-Performance Binary-Collision-Approximation Code for Ion-Material Interactions. *Journal of Open Source Software, 6*(64), 3298. doi: `10.21105/joss.03298`

3. Smith Roger, Kenny Steven D. and Ramasawmy Deerajen 2004Molecular-dynamics simulations of sputteringPhil. Trans. R. Soc. A.362157–176. doi: `10.1098/rsta.2003.1308`

4. Hofsäß, H., Zhang, K., Mutzke, A. (2014). Simulation of ion beam sputtering with SDTrimSP, TRIDYN and SRIM. Applied Surface Science, 310, 134–141. https://doi.org/10.1016/j.apsusc.2014.03.152