MATERIAL SPUTTERING AT LOW ION ENERGIES: FROM ATOMISTICS TO ENGINEERING MODELS WITH MOLECULAR DYNAMICS, MONTE CARLO, AND DATA-DRIVEN APPROACHES

BY

HUY DANG TRAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
with a concentration in Computational Science and Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Doctoral Committee:

Professor Huck Beng Chew, Chair and Director of Research
Professor Joshua L. Rovey
Professor Deborah Levin
Professor Azer P. Yalin, Colorado State University
Professor John D. Williams, Colorado State University

ABSTRACT

Sputtering is a fundamental phenomenon describing the ejection of particles from a material surface under the bombardment of energetic ions. The removal of these atoms, under a controlled process, has revolutionized nano- and micro-scale manufacturing through surface patterning, and fabrication of thin film structures with enhanced physical, mechanical, and optical properties. However, sputtering is generally undesirable for plasma-facing materials, such as those in fusion and space electric propulsion devices, since it erodes the surfaces of critical material components and alters the plasma operational conditions. Regardless, sputtering predictions are highly unreliable at low ion incidence energies, typically less than 2 keV, due to the severely sparse and conflicting experimental data available, as well as the lack of understanding of the fundamental sputtering mechanisms at these ion energies. For instance, measurements conducted in different laboratories show that the sputter yield (number of ejected surface atoms per impact ion) of xenon (Xe) ions on carbon surfaces can vary by nearly an order of magnitude at low ion energies. Additionally, the sputter yield, differential yield (ejection angle), and energy distribution of the sputtered carbon atoms are very sparse for the noble gas bombardment of carbon at off-normal ion incidence angles. This dissertation seeks to harness the power of highly-parallelized supercomputing facilities, along with recent advancements in data science, to provide fundamental insights into the sputtering mechanisms of carbon under the lower-energy bombardment of noble gas ions, with the long-term goal of developing physics-based predictive models, which are applicable to general ion/target combinations across scales.

My thesis research is divided into three parts. The first part is motivated by the reported large scatter in the sputtering data of carbon materials under the bombardment of low-energy noble

gas ions. Here, I conducted scale-bridging molecular dynamics (MD) simulations on the xenon bombardment of carbon substrates across ion energies of 75 eV to 2 keV, and at ion incidence angles of 0° to 75°, to resolve uncertainties in the sputtering data. My results showed rapid amorphization of the carbon subsurface with ion bombardment, but the structural characteristics ($sp/sp^2/sp^3$ bond proportion, atomic density) eventually plateau once steady-state sputtering is achieved. In addition, my MD simulations showed that virtually indistinguishable steady-state amorphous sub-surface carbon structures are obtained across the range of ion energies and ion incidence angles, as well as for several different initial carbon structures (graphite, diamond), which suggests that the steady-state sputtering yield data obtained from MD is independent of the initial carbon structure and prior sputtering history.

In the second part of my thesis research, I accounted for the evolving surface morphology and sputtering yield with ion fluence by upscaling my MD simulation results to a Monte Carlo (MC) model, which considers the shadowing of the incident ion flux, redeposition of sputtered carbon material, and secondary sputtering induced by the surface impact of the carbon sputterants. Results show that initially rough surface morphologies are consistently smoothened and flattened by a normal ion flux, resulting in sputtering yields that approach MD predictions. Under a highly oblique ion flux, however, the activation of multiple cooperative roughening and smoothening mechanisms at different scales lead to the formation of characteristic surface steps at the microscale, with steady-state sputtering yields that are up to an order-of-magnitude lower than MD predictions. While the observed surface features and the ensuing sputtering yield at steady-state are generally not sensitive to the initial surface morphology, the initial morphology controls the critical ion fluence required to attain steady-state sputter yield. Based on these observations,

different initial surface topologies are proposed to delay and abate sputtering, which is a key step towards achieving sputtering-by-design.

The parameters of a semi-empirical sputtering model are calibrated based on the MD-MC data at steady-state sputtering yield using a Bayesian approach. While this reduced-order semi-empirical model captures the sputtering predictions for the xenon ion bombardment of carbon substrates, extending this model to other ion/target combinations requires recalibration of the model fitting parameters. The key parameter that has the largest uncertainty is the sputtering threshold energy, which is the minimum ion incidence energy to initiate sputtering. In the third part of my thesis research, I focus on obtaining a data-driven expression for the threshold energy for sputtering applicable to general noble gas ion/target combinations. Using MD simulations, I quantify the sputtering threshold energies of monoatomic surfaces under the bombardment of noble gas ions across a broad range of ion-target combinations. The resulting threshold energies are related to the ion-target properties through an evolutionary algorithm for symbolic regression, and show a strong functional dependence on the nucleus charge governing ion-target repulsion and the target density, in addition to the heat of sublimation and ion-target mass ratio in prior semi-empirical models. This new data-driven formulation has order-of-magnitude improved threshold energy predictions and is applicable to crystalline and amorphous targets.

I conclude my thesis with some proposed strategies to arrive at a new general closed-form expression for the sputtering yield applicable to a wide range of monatomic solids under noble gas ion bombardments. A road map for quantifying the sticking coefficients of carbon sputterants on the surfaces of different monoatomic substrates is also discussed.

ACKNOWLEDGMENTS

I started my Ph.D. in August 2020, the beginning year of the COVID pandemic. In retrospect, I have never expected myself to pursue a Ph.D., as I favor breadth education rather than the depth, localized, and often "niche" topics, typical to a doctorate. Throughout my undergraduate and my industry internship, too frequently I heard from my colleagues and peers that a Ph.D. is only reserved for someone who has previous research experiences, knows exactly the topic they want to pursue, and enjoys discovering new sciences. In a way, I am anything but that. I enjoy learning new things from different disciplines and do not want to settle down for any specific field. While I like solving problems, specifically those that help advance space technology, I often neglect the fundamental nitty-gritty science while favoring more practical solutions. Nevertheless, I value the power of computational engineering, and since most career opportunities require a higher degree, I still decided to pursue a doctorate.

Fast forward to October 2024 writing the thesis and reflecting on my graduate journey, I am, surprisingly, pleased and satisfied with my Ph.D. career, thanks to my advisor, Prof. Huck Beng Chew. While still centered around modeling of material, I am grateful to Prof. Chew for making my Ph.D. as breadth as a Ph.D. could get. Prof. Chew is not afraid to venture into a new area and constantly pushes me out of my comfort zone to search for new approaches to solve problems, regardless of our familiarity of such topics. This resulted in my Ph.D. works utilizing diverse computational techniques, from quantum calculations, atomistic modeling, Monte Carlo method, uncertainty quantification, finite element analysis, cohesive zone law, and phase field approach, to high performance computing, symbolic regression, genetic programming, and scientific machine learning to solve different problems in surface sputtering and fracture

mechanics. In addition to many fruitful discussions with my peers in the research group and our collaborators, I also learned about artificial and convolution neural networks, plasma physics, as well as other experiment techniques such as electron diffraction, atomic force microscopes, and mass spectroscopy. These opportunities are often not easy to have. What I presented here in my dissertation is only a portion of my research and what I learned throughout my graduate studies.

On top of that, Prof. Chew's advising, and our numerous discussions made me realize the importance of science in engineering, especially in computational modeling, and there is fun in discovering new things and the unknown as well. I am also grateful to him for his constant availability, patience and for teaching me that it is worthwhile to approach a problem, no matter how convoluted, with honesty, systematic, and curiosity. This philosophy will continue to be my guidance in the next stage of my career. To this day, I do not know why he selected me as his student, yet it is a privilege to be his student, and I am truly thankful for the opportunity. Once again, I sincerely want to express my utmost gratitude to Prof. Huck Beng Chew, for making research, and consequently my Ph.D. journey, diverse, exciting, memorable, and full of surprises.

I would also like to thank Prof. Josh Rovey, Prof. Deborah Levin of UIUC, Prof. Yanfei Gao of the University of Tennessee and Prof. Yalin Azer, Prof. John Williams of Colorado State University for our collaborations and sound suggestions for my research. I am very grateful for the funding through the Joint Advanced Propulsion Institute, a NASA Space Technology Research Institute, under grant number 80NSSC21K1118, and through National Science Foundation Grants Nos. NSF-DMR-18-09696 and NSF-CMMI-2009684. When I first came to the US for education 11 years ago, I would have never expected to work on a NASA-funded project so the NASA funding is especially meaningful to me. Special thanks to all the High Performing Computing allocations I got to use: Blue Waters, Delta (MAT220005), Anvil (MAT210031), Stampede2 (TG-

PHY220010, TG-MAT210010, TG-PHY210065), Stampede3 (MAT230069, MAT240041, MAT240043), and Frontera (CTS21001, DMR22021, MSS22006). The work presented in this dissertation would have not been possible without these supercomputing resources.

Half of my Ph.D. was during COVID lockdown, along with moving to another state, which makes finding friends and support groups especially challenging. Therefore, I also would like to express my highest gratitude to the few close friends I made here at UIUC: Ning Li, Michael Worthington, William Noh in my research group; Sean Clark, Josh Tompkins (and his partner Moises Ramirez), Dinh Nguyen at other research groups. With special thanks to William, Josh, and Moises, for always being there whenever I am anxious about my Ph.D. and my future. I would also like to thank my college friends at the University of Virginia: Deanna Grant, Diana Nguyen, Josie Li, and Julie Wang, also my middle school friends in Vietnam: Duy Thang, Phuong Vy, and Minh Thuan, who have always been supportive to my decisions and helped me get through the toughest time during my Ph.D.

In the end, I would like to thank my family for their unconditional trust and for being patient with me.

"The challenge we all face is how to maintain the benefits of breadth, diverse experience, interdisciplinary thinking, and delayed concentration in a world that increasingly incentivizes, even demands, hyper-specialization."

— David Epstein, Range: How Generalists Triumph in a Specialized World

To myself 11 years ago – it will be alright!

TABLE OF CONTENTS

CHAPTER 1: MOTIVATION

1.1 Sputtering Revolutionizes Nano-microscale Manufacturing

The advancements in everyday electronics, semiconductors, engineering materials, and structures go hand-in-hand with our ability to fabricate these devices and their components at the micro- ($\sim 10^{-6}$ m) and nano- ($\sim 10^{-9}$ m) scales with high precision. For instance, fiber-like nanostructures, manufactured via laser ablation, have been used in many current state-of-the-art biosensors [1,2] that can (electronic, optical, and magnetic) bind to biological molecules (e.g., peptides, proteins, and nucleic acids) to detect and amplify various signals [3,4]. These nanoscale biosensors are extremely sensitive, and can detect chemical and biological species for human and robotic exploration with high precision, while having low mass and volume with low power demand [5]. As a result, NASA has started to incorporate these sensors for e-textiles in next-generation astronaut suits and rovers [6,7].

However, nano-micro scale fabrications are difficult and expensive as they cannot utilize conventional mesoscale manufacturing techniques. On one hand, shrinking machining and power tools for subtractive approaches significantly compromise the structural integrity and precision of such tools [8]. On the other hand, the material's liquid state at the micro-nano scales becomes highly viscous, which puts significant stress on the micro-nozzle, rendering additive 3D printing techniques ineffective [9]. As a result, producing these micro-nanoscale devices and structures often requires the removal or addition of materials no more than a few Angstrom (Å) wide, i.e. on the scale of individual atoms.

This process of removal of atoms from the material surfaces is termed sputtering, where the ejected (sputtered) surface's atoms can then be collected (deposited) to build nano-micro films

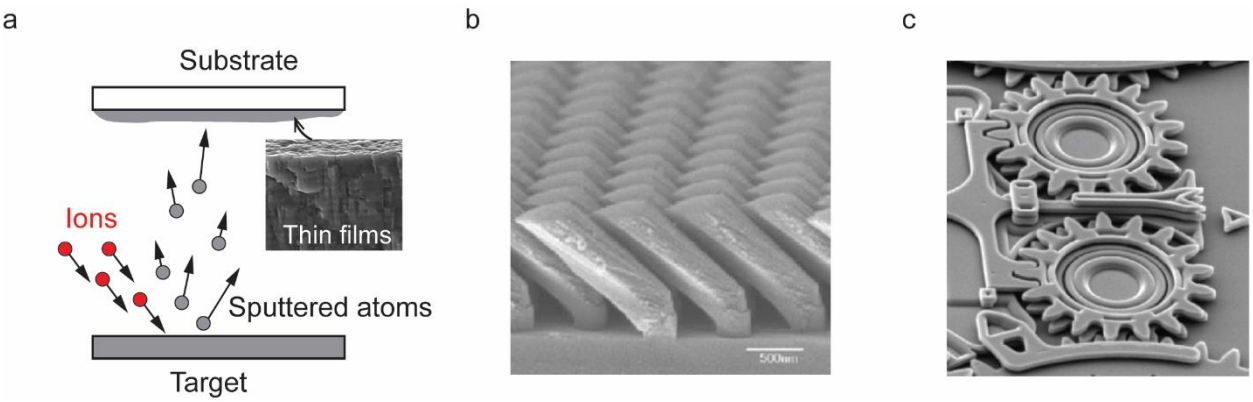

Figure 1.1: (a) Schematic demonstrates the physical vapor deposition (PVD) process **(b)** SEM image for columnar structures, created by PVD with angled substrate [13] **(c)** SEM image of micro-electromechanical systems, created by both PVD for film growth and FIB for etching [14]

and structures (Fig. 1.1a). First reported in the early 1800s, sputter deposition, made possible due to the development of vacuum pumps and electrical power in the 1700s, already dominated the optical-coating market by 1880 [10] for thin film coating. During these years, film atoms are slowly sputtered in vapor form at small velocity via high-temperature glow-discharge [11], which significantly limits the shapes and properties of the film. Fast forward to the 1900s, the revolution in particle physics led researchers to achieve "hypersonic ions", accelerated by either a magnetic or an electric field, which can impact the surface and cause sputtering. This concept was first realized in the 1930s when it marked the first physical vapor deposition (PVD) of thin films via magnetron sputtering [11]. In 1959, Feynman suggested the use of dense ion beams for etching and milling at high beam currents and imaging at low beam currents [10,11], leading to the first focused ion beams (FIB) developed in 1975 [12]. Together in the next 50 years, PVD and FIB have enabled extensive film growth techniques to achieve films of tailorable structures and properties (Fig. 1.1b), which have ultimately led to the development of micro-electromechanical systems (Fig. 1.1c) [13,14] for piezoelectric, accelerometers, inertial measurement units, barometers, radio frequency switches, [15–17] and semiconductors for defect analysis, circuit modification, photomask repair and transmission or scanning electron microscope (TEM and

SEM) sample preparation of site-specific locations on integrated circuits [18–20] to name a few. The implications and importance of these devices and technologies in the advancement of the today's modern world are non-disputable.

1.2 Role of Sputtering in Plasma Facing Materials

While sputtering is highly desirable in manufacturing, it is detrimental to many plasma engineering devices. These devices utilize the physics of plasma, a state of matter consisting of partially ionized (missing electrons) gas atoms, not only to improve processes and performance over many fields as described in the previous section but also in energy production via fusion and in spacecraft propulsion [21–23], with the latter being the main application of focus in my thesis.

Space Electric Propulsion (EP) devices, most notable are the ion thruster and Hall effect thruster, utilize a combination of the magnetic and electric fields to ionize the gas, typically xenon (Xe) due to its large atomic mass, and to eject it outwards. Based on Newton's Third Law, this process creates a reaction force acting on the spacecraft. The ejected ions can be up to twenty times faster than a classical combustion thruster; therefore, EP requires very little mass to accelerate a spacecraft. While chemical propulsion is limited in energy stored in the bonds, EP is only limited by the available electrical power on board the spacecraft [23]. As a trade-off, the initial thrust in EP thrusters is low so EP is mainly suitable for low thrust trajectories, mid-course correction, and deep-space exploration [23]. Consequently, EP has been critical for countless NASA and commercial missions over the past thirty years, such as Dawn, Artemis, Dart, and Starlink [24].

In EP, the accelerated ion can impact plasma facing surfaces to cause sputter erosion, and alter the plasma operational conditions [25–27]. In gridded ion thrusters, the impingement of charge-exchange Xe ions on the inside of the accelerator grid holes widens these holes during operation (Fig. 1.2a) [27]. In the Hall effect thrusters, the impingement occurs at the outer wall

chamber and the pole cover, leading to gradual removal of the protective components, and eventually exposing the sealed critical magnet to the plasma environment (Fig. 1.2b) [25]. In addition, long-duration ion engine testing on the ground inevitably results in the deposition of back-sputtered species (typically steel, aluminum, and graphite) from the facility onto the EP thrusters, causing significant uncertainties in performance and lifetime predictions (Fig. 1.2c) [28].

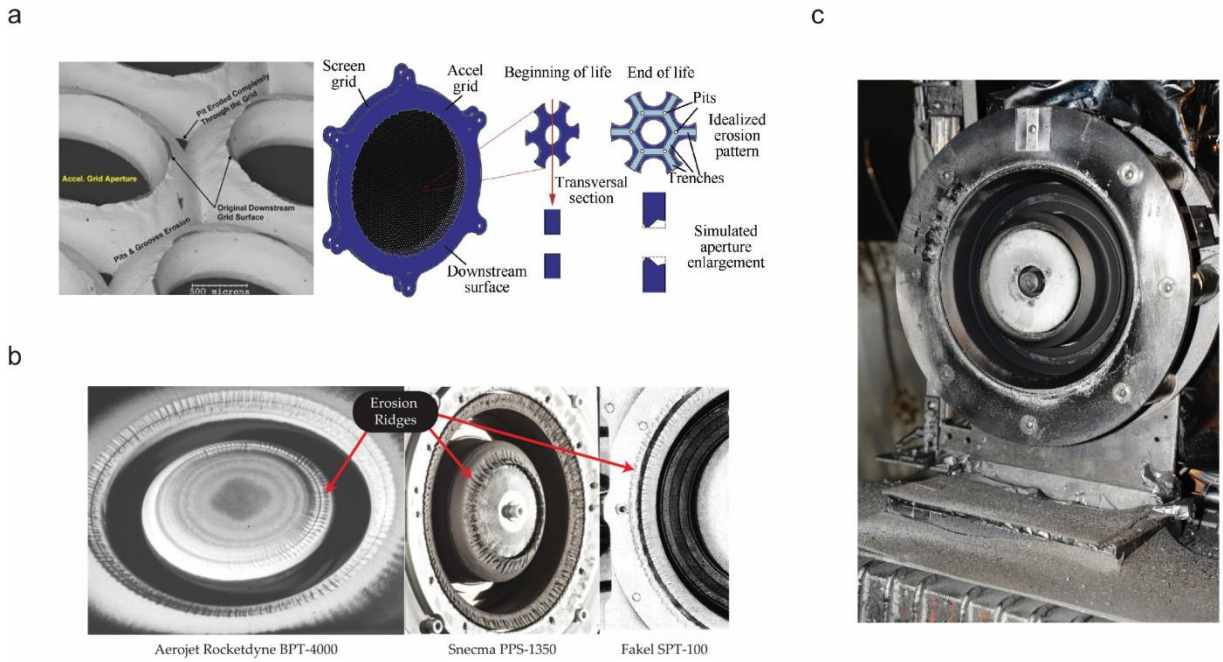

Figure 1.2: Problems related to sputtering in space electric propulsion systems showing **(a)** widening of grid in ion-thruster [27] **(b)** erosion of the pole cover in Hall-effect thruster [25] **(c)** significant contaminated carbon on long duration test of a Hall-effect thruster [26]

1.3 Challenges: Sputtering at Low Ion Incidence Energy

Enabling further advancement in EP technology, such as high-power EP for human space-flight missions on Mars, requires an in-depth physical understanding and prediction of material sputtering under ion bombardment since erosion and facility contamination effects become more pronounced under long duration testing. This becomes the premise of a 5-year multi-university NASA research center through the Joint Advanced Propulsion Institute (JANUS), a NASA Space Technology Research Institute, and the main funding source for the work presented in my thesis.

Aside from the physics underpinning sputtering, the relevant sputtering quantities of interest are sputter yield, Y, defined by the average number of ejected surface's atoms per impact ion [29], the distribution of sputtered atoms angle, θ_t and energy, E_t as a function of the incidence ion impact energy, E_i and angle θ_i (Fig. 1.3a) [30,31]. Sputter yield can be measured via the target's mass loss [32] or through a quartz-crystal microbalance (QCM) for higher sensitivity [33] and normalized by the number of incoming ion particles, measured as current through a Faraday probe [34]. During the sputter experiment (Fig. 1.3b), single or multiple QCM(s) on a rotational stage [33,35,36] can also be used to construct the sputterants' angle distribution profile, commonly termed the differential yield, with the energy of sputtered charge particles measured using an electrostatic spectrometer [37–39].

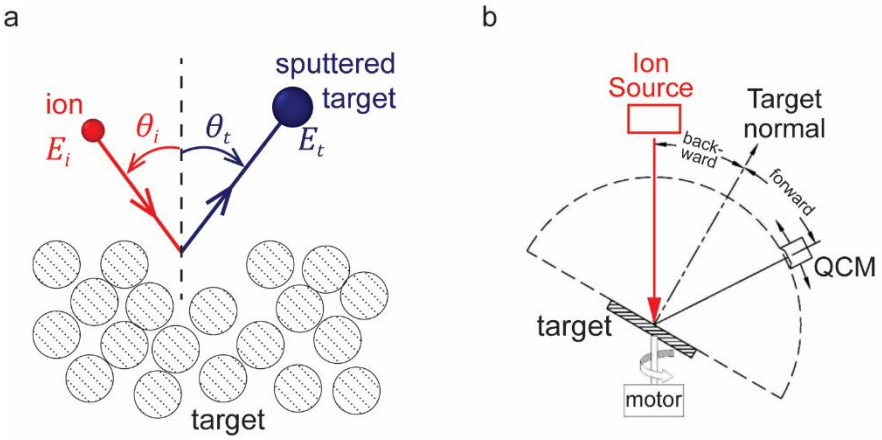

Figure 1.3: Schematics demonstrate **(a)** convention for a sputtering ion-target systems **(b)** sputtering experiment set up to measure sputter yield and sputterants ejection angle distribution [33,35,36]

While numerous experimental studies have been conducted to obtain these quantities associated with plasma-surface interactions [40], they are focused on surface patterning, etching, and milling applications where the ion incidence energy is quite high (> 2 keV). In contrast, the ion impact energy in high-power EP applications ranges between 30 eV – 2 keV [23], which is in a unique regime where the impact energy is sufficient to cause sputtering but yet low enough to cause profound challenges in experimental measurements of these sputtering properties. The most obvious challenge comes from the low sputter yield at these low ion energies, which necessitates

a testing time of up to tens of hours, or even days to ensure the measured mass loss (or gain for QCM [35]) is sufficient for reliable (adequate) sputter yield measurements and to construct a statistically significant distribution. During these long testing times, any minor changes in the ion beam energy distribution, angle, and the target material's compositional or topological properties can easily skew and bias the measured results [32,41,42]. These problems are exacerbated when the ion sources are noble gas ions – the most common EP propellant due to its chemical inertness, since noble gas is expensive and the ions are easily prone to secondary electron emissions, charge exchange attenuation, implantation and outgassing [43].

One important example is to characterize the sputter properties for graphitic or carbon structures under Xe ion bombardment. Graphite is often selected as the coating of choice for the accelerator grids and pole covers to extend the thruster life because of its low sputter yield and low thermal expansion coefficient. However, even this sputter-resistant material will eventually erode under the extended burn times of the ion engine typical for low thrust trajectories and deep-space missions [28,44]. In spite of ongoing efforts over the past several decades, measuring the sputtering yield of carbon at low ion energies of 75 to 1000 eV relevant to electric space propulsion devices is non-trivial, because of the very low sputter yield of carbon [34,45–48]. Studies have reported large scatter in the measured sputter yield data, particularly between measurements conducted in different laboratories, where the sputter yield can vary by nearly an order of magnitude at low ion energies [44]. The available sputter yield data for carbon also tends to be sparse at off-normal ion incidence angles, and extrapolation to the range of interest can introduce large errors. At the Xe energy range of interest, there is only one study by Williams et. al [33] that provides the sputtered carbon angle distribution profile while the sputtered energy profile is, to the best of my knowledge, non-existent. On top of that, carbon has many different allotropes and exists in many polymorphs.

The influence of different carbon structures on the sputtering yield characteristic is unknown. While many studies have shown that similar sputtering yield and differential yield profiles are obtained regardless of the type of carbon structure (e.g., diamond, amorphous carbon, pyrolytic and isotropic graphite, as well as carbon-carbon composites) [32,33,49], the underlying mechanism is unclear, given the significantly different physical and mechanical properties of these carbon structures. Studies also suggest that the measured sputtering yield may be sensitive to surface roughness, as evidenced by surface texturing on carbon films with height variations of ~0.2 to 0.3 μm [32,50]. For example, Deltschew et al. [51] reported a two-fold higher sputtering yield of exposed carbon fibers compared to graphite films, which they attribute to the unique and evolving surface morphology of these fibers. Material surface roughness can also significantly change the local incidence angle, and influence the trajectory of the sputterants; the evolving surface morphology can also change the measured sputtering yield [51,52]. Both these effects are challenging to quantify experimentally.

1.4 Organization of this Thesis

My thesis research investigates the sputtering of carbon substrates by Xe ion bombardment at low ion incidence energies relevant to space electric propulsion. The goal is to elucidate the fundamental mechanisms behind low energy ion sputtering and to develop physics-based constitutive models for sputtering which are applicable to plasma-material experiments and simulations [53–55]. Over the years, there have been various efforts to arrive at predictive and useful models to predict material sputtering, ranging from semi-empirical fitting models, mathematical theories, to reduced-order simulations. Chapter 2 provides a comprehensive critique of these various approaches, and how limitations of these models for low ion energy sputtering can be overcome with multi-scale models as well as data-driven approaches.

Since Xe → C is the most common plasma-surface interaction system in high-power EP, Chapter 3 first investigates the elemental sputtering processes of low energy Xe ion bombardment on different carbon structures using Molecular Dynamics (MD) simulations. Efforts discussed in Chapter 3 provide new insights into the complicated sputtering process and advance further understanding of the influence of surface morphology and carbon structure effects on sputtering. In Chapter 3, a new closed-form analytical equation is also proposed for sputter yield and sputter angle distribution that bridges between the length-scales of MD simulations and experiments.

Chapter 4 upscales the MD simulation results in Chapter 3 to a Monte Carlo (MC) model to allow for surface topology evolution with ion fluence. Through this MD-MC approach, I uncover the fundamental mechanisms underpinning several experimentally observed surface evolution phenomena observed under the bombardment of normal and oblique noble gas ions. Quantitatively, the results presented in Chapter 4 showed that the transient-to-steady-state surface morphology evolution during the noble gas bombardment of carbon surfaces is responsible for significant fluctuations in the sputtering yield – the bounded uncertainties well encompass a large portion of the experimental scatter across the range of ion energies and incidence angle. The predictions of the MD-MC approach were further used to design surface patterns to enhance the sputter resistance capabilities of EP thruster and wall chamber surfaces. Finally, the parameters of the reduced-order, semi-empirical sputtering yield models are then updated accordingly.

While this MD-MC-reduced-order framework can be extended for any noble gas ion-target material systems, the accuracy of the final reduced-order, semi-empirical model relies heavily on the provided elemental sputtering yield properties from MD simulations. However, different ion-target material combinations would require recalibration of the fitting parameters through this MD-MC-reduced-order framework – a computationally expensive process. However, I hypothesize that

the empirical fitting parameters of the final reduced-order model can be expressed as a function of the ion/target properties, and I seek to obtain such an expression for one of the key fitting parameters – the sputtering threshold energy. In Chapter 5, a new data-driven expression for the sputtering threshold energy, applicable to the noble gas ion bombardment of crystalline and amorphous targets, has been formulated based on a massive ion-target dataset generated by MD simulations. The expression has strong underlying physics and achieves significantly improved predictions over prior semi-empirical models.

Finally, Chapter 6 lays out the possible future directions of the work presented in my thesis, including the framework to arrive at a closed-form expression for the remaining fitting parameters. In addition, a possible framework to obtain the sticking coefficients from the deposition of carbon sputterants is also proposed, which is an important parameter for carbon transport modeling to quantify facility back-sputtered effects in EP ground testing.

1.5 References

[1] S. Malik, J. Singh, R. Goyat, Y. Saharan, V. Chaudhry, A. Umar, A.A. Ibrahim, S. Akbar, S. Ameen, S. Baskoutas, Nanomaterials-based biosensor and their applications: A review, Heliyon 9 (2023) e19929. https://doi.org/10.1016/j.heliyon.2023.e19929.

[2] G. Karabulut, N.B. Üllen, S. Karakuş, G. Karabulut, N.B. Üllen, S. Karakuş, Nanostructures in Biosensors: Development and Applications, in: Biosignal Processing, IntechOpen, 2022. https://doi.org/10.5772/intechopen.108508.

[3] V. Chaudhary, S. Rustagi, A. Kaushik, Bio-derived smart nanostructures for efficient biosensors, Current Opinion in Green and Sustainable Chemistry 42 (2023) 100817. https://doi.org/10.1016/j.cogsc.2023.100817.

[4] M. Ramesh, R. Janani, C. Deepa, L. Rajeshkumar, Nanotechnology-Enabled Biosensors: A Review of Fundamentals, Design Principles, Materials, and Applications, Biosensors (Basel) 13 (2022) 40. https://doi.org/10.3390/bios13010040.

[5] S.R.S. Maria, Lunar BioSensor: An Autonomous Instrument to Study the Effects of the Lunar Environment on Biological Organisms, (n.d.). https://ntrs.nasa.gov/citations/20210000690 (accessed September 19, 2024).

[6] R. Bogue, Nanosensors: a review of recent research, Sensor Review 29 (2009) 310–315. https://doi.org/10.1108/02602280910986539.

[7] J.E. Koehne, Carbon Nanofiber Nanoelectrodes for Biosensing Applications, (2014). https://ntrs.nasa.gov/citations/20140006730 (accessed September 19, 2024).

[8] A.R. Razali, Y. Qin, A Review on Micro-manufacturing, Micro-forming and their Key Issues, Procedia Engineering 53 (2013) 665–672. https://doi.org/10.1016/j.proeng.2013.02.086.
[9] L. Xu, L. Qi, K. Li, H. Zou, Polymer nano nozzle fabricated by nanoscale electrohydrodynamic jet printing for high-resolution printing of low-viscosity inks, Materials & Design 233 (2023) 112192. https://doi.org/10.1016/j.matdes.2023.112192.
[10] V.S. Smentkowski, Trends in sputtering, Progress in Surface Science 64 (2000) 1–58. https://doi.org/10.1016/S0079-6816(99)00021-0.
[11] J.E. Greene, Review Article: Tracing the recorded history of thin-film sputter deposition: From the 1800s to 2017, Journal of Vacuum Science & Technology A 35 (2017) 05C204. https://doi.org/10.1116/1.4998940.
[12] C.A. Volkert, A.M. Minor, Focused Ion Beam Microscopy and Micromachining, MRS Bulletin 32 (2007) 389–399. https://doi.org/10.1557/mrs2007.62.
[13] D.J. Poxson, M.-L. Kuo, F.W. Mont, Y.-S. Kim, X. Yan, R.E. Welser, A.K. Sood, J. Cho, S.-Y. Lin, E.F. Schubert, High-performance antireflection coatings utilizing nanoporous layers, MRS Bulletin 36 (2011) 434–438. https://doi.org/10.1557/mrs.2011.110.
[14] D.-X. Ye, T. Karabacak, B.K. Lim, G.-C. Wang, T.-M. Lu, Growth of uniformly aligned nanorod arrays by oblique angle deposition with two-phase substrate rotation, Nanotechnology 15 (2004) 817. https://doi.org/10.1088/0957-4484/15/7/018.
[15] D.A. Antartis, R.N. Mott, I. Chasiotis, Si nanospring films for compliant interfaces, J Mater Sci 53 (2018) 5826–5844. https://doi.org/10.1007/s10853-017-1750-x.
[16] K.-K. Hung, I. Chasiotis, Control of Surface Wrinkling through Compliant Nanostructured Interfaces, Advanced Materials Interfaces 9 (2022) 2101583. https://doi.org/10.1002/admi.202101583.
[17] C. Acar, A. Shkel, MEMS Vibratory Gyroscopes: Structural Approaches to Improve Robustness, Springer Science & Business Media, 2008.
[18] M.K. Miller, K.F. Russell, Atom probe specimen preparation with a dual beam SEM/FIB miller, Ultramicroscopy 107 (2007) 761–766. https://doi.org/10.1016/j.ultramic.2007.02.023.
[19] E.L. Principe, P. Gnauck, P. Hoffrogge, A Three Beam Approach to TEM Preparation Using In-situ Low Voltage Argon Ion Final Milling in a FIB-SEM Instrument, Microscopy and Microanalysis 11 (2005) 830–831. https://doi.org/10.1017/S1431927605502460.
[20] J. Orloff, L.W. Swanson, M. Utlaut, Fundamental limits to imaging resolution for focused ion beams, Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena 14 (1996) 3759–3763. https://doi.org/10.1116/1.588663.
[21] R.A. Pitts, J.P. Coad, D.P. Coster, G. Federici, W. Fundamenski, J. Horacek, K. Krieger, A. Kukushkin, J. Likonen, G.F. Matthews, M. Rubel, J.D. Strachan, J.-E. contributors, Material erosion and migration in tokamaks, Plasma Phys. Control. Fusion 47 (2005) B303. https://doi.org/10.1088/0741-3335/47/12B/S22.
[22] A. Hakola, J. Likonen, A. Lahtinen, T. Vuoriheimo, M. Groth, H. Kumpulainen, M. Balden, K. Krieger, M. Mayer, T. Schwarz-Selinger, S. Brezinsek, M. Kelemen, S. Markelj, M. Barac, S. Gouasmia, I.B. Radovic, A. Uccello, E. Vassallo, D. Dellasega, M. Passoni, M. Sala, E. Bernard, M. Diez, C. Guillemaut, E. Tsitrone, the A.U. Team, the Euro.M. Team, the Euro.W.P. Contributors, Gross and net erosion balance of plasma-

facing materials in full-W tokamaks, Nucl. Fusion 61 (2021) 116006. https://doi.org/10.1088/1741-4326/ac22d2.
[23] R.G. Jahn, Physics of Electric Propulsion, Courier Corporation, 2012.
[24] B. Kantsiper, The Double Asteroid Redirection Test (DART) mission electric propulsion trade, in: 2017 IEEE Aerospace Conference, 2017: pp. 1–7. https://doi.org/10.1109/AERO.2017.7943736.
[25] N.P. Brown, M.L.R. Walker, Review of Plasma-Induced Hall Thruster Erosion, Applied Sciences 10 (2020) 3775. https://doi.org/10.3390/app10113775.
[26] J.W. Dankanich, M. Walker, M.W. Swiatek, J.T. Yim, Recommended Practice for Pressure Measurement and Calculation of Effective Pumping Speed in Electric Propulsion Testing, Journal of Propulsion and Power 33 (2017) 668–680. https://doi.org/10.2514/1.B35478.
[27] A. Yalin, V. Surla, C. Farnell, M. Butweiller, C. Shadburn, J. Williams, Sputtering Studies of Multi-Component Materials by Weight Loss and Cavity Ring-Down Spectroscopy, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, n.d. https://doi.org/10.2514/6.2006-4338.
[28] R.B. Lobbia, J.E. Polk, R.R. Hofer, V.H. Chaplin, B. Jorns, Accelerating 23,000 hours of Ground Test Backsputtered Carbon on a Magnetically Shielded Hall Thruster, in: AIAA Propulsion and Energy 2019 Forum, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-3898.
[29] P. Sigmund, Sputtering by ion bombardment theoretical concepts, in: R. Behrisch (Ed.), Sputtering by Particle Bombardment I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981: pp. 9–71. https://doi.org/10.1007/3540105212_7.
[30] W. Eckstein, Energy distributions of sputtered particles, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 18 (1986) 344–348. https://doi.org/10.1016/S0168-583X(86)80056-8.
[31] W. Eckstein, C. Garciá-Rosales, J. Roth, J. László, Threshold energy for sputtering and its dependence on angle of incidence, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 83 (1993) 95–109. https://doi.org/10.1016/0168-583X(93)95913-P.
[32] M. Tartz, Pyrolytic Graphite and Carbon-carbon Sputter Behaviour Under Xenon Ion Incidence, in: 29th International Electric Propulsion Conference, 2005: p. 10.
[33] J. Williams, M. Johnson, D. Williams, Differential Sputtering Behavior of Pyrolytic Graphite and Carbon-Carbon Composite Under Xenon Bombardment, in: 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, American Institute of Aeronautics and Astronautics, Fort Lauderdale, Florida, 2004. https://doi.org/10.2514/6.2004-3788.
[34] R.P. Doerner, D.G. Whyte, D.M. Goebel, Sputtering yield measurements during low energy xenon plasma bombardment, Journal of Applied Physics 93 (2003) 5816–5823. https://doi.org/10.1063/1.1566474.
[35] A.P. Yalin, J.D. Williams, V. Surla, K.A. Zoerb, Differential sputter yield profiles of molybdenum due to bombardment by low energy xenon ions at normal and oblique incidence, J. Phys. D: Appl. Phys. 40 (2007) 3194. https://doi.org/10.1088/0022-3727/40/10/025.
[36] A. Yalin, B. Rubin, S. Domingue, Z. Glueckert, J. Williams, Differential Sputter Yields Of Boron Nitride, Quartz, and Kapton Due to Low Energy Xe+ Bombardment, in: 43rd

AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, 2007. https://doi.org/10.2514/6.2007-5314.
[37] R.V. Stuart, G.K. Wehner, G.S. Anderson, Energy Distribution of Atoms Sputtered from Polycrystalline Metals, Journal of Applied Physics 40 (1969) 803–812. https://doi.org/10.1063/1.1657467.
[38] A.E. Ieshkin, A.V. Nazarov, A.A. Tatarintsev, D.S. Kireev, A.D. Zavilgelsky, A.A. Shemukhin, V.S. Chernysh, Energy distributions of the particles sputtered by gas cluster ions. Experiment and computer simulation, Surface and Coatings Technology 404 (2020) 126505. https://doi.org/10.1016/j.surfcoat.2020.126505.
[39] S. Ertmer, O. Marchuk, S. Dickheuer, S. Brezinsek, P. Boerner, J. Schmitz, A. Kreter, Measurements of the energy distribution of W atoms sputtered by low energy Ar ions using high-resolution Doppler spectroscopy, Plasma Phys. Control. Fusion 63 (2020) 015008. https://doi.org/10.1088/1361-6587/abc519.
[40] N. Matsunami, Y. Yamamura, Y. Itikawa, N. Itoh, Y. Kazumata, S. Miyagawa, K. Morita, R. Shimizu, H. Tawara, Energy Dependence of Ion-Induced Sputtering Yields of Monatomic Solids, Atomic Data and Nuclear Data Tables 31 (1984) 1. https://doi.org/10.1016/0092-640X(84)90016-0.
[41] R.D. Kolasinski, J.E. Polk, D. Goebel, L.K. Johnson, Sputtering yield measurements at glancing incidence using a quartz crystal microbalance, Journal of Vacuum Science & Technology A 25 (2007) 236–245. https://doi.org/10.1116/1.2435375.
[42] R. Kolasinski, J. Polk, D. Goebel, L. Johnson, Carbon Sputtering Yield Measurements at Grazing Incidence, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, Sacramento, California, 2006. https://doi.org/10.2514/6.2006-4337.
[43] J.E. Polk, A critical review and meta-analysis of xenon-on-carbon sputter yield data, Journal of Applied Physics 135 (2024) 040701. https://doi.org/10.1063/5.0180720.
[44] J.T. Yim, Yim, J.T. (2017). A Survey of Xenon Ion Sputter Yield Data and Fits Relevant to Electric Propulsion Spacecraft Integration., in: 35th International Electric Propulsion Conference, 2017.
[45] D. Rosenberg, G.K. Wehner, Sputtering Yields for Low Energy He^+-, Kr^+-, and Xe^+-Ion Bombardment, Journal of Applied Physics 33 (1962) 1842–1845. https://doi.org/10.1063/1.1728843.
[46] E. Hechtl, J. Bohdansky, Sputtering behavior of graphite and molybdenum at low bombarding energies, Journal of Nuclear Materials 123 (1984) 1431–1436. https://doi.org/10.1016/0022-3115(84)90280-0.
[47] J. Gruber, Low-Energy Sputter Erosion of Various Materials in a T 5, (2001).
[48] E. Oyarzabal, R.P. Doerner, M. Shimada, G.R. Tynan, Carbon atom and cluster sputtering under low-energy noble gas plasma bombardment, Journal of Applied Physics 104 (2008) 043305. https://doi.org/10.1063/1.2968549.
[49] J.J. Blandino, D.G. Goodwin, C.E. Garner, Low energy sputter yields for diamond, carbon–carbon composite, and molybdenum subject to xenon ion bombardment, Diamond and Related Materials 9 (2000) 1992–2001. https://doi.org/10.1016/S0925-9635(00)00350-2.
[50] S. Habenicht, Morphology of graphite surfaces after ion-beam erosion, Phys. Rev. B 63 (2001) 125419. https://doi.org/10.1103/PhysRevB.63.125419.

[51] R. Deltschew, M. Tartz, V. Plicht, E. Hartmann, H. Neumann, Sputter characteristics of carbon-carbon compound material, in: 27th International Electric Propulsion Conference, 2001.
[52] Q. Wei, K.-D. Li, J. Lian, L. Wang, Angular dependence of sputtering yield of amorphous and polycrystalline materials, J. Phys. D: Appl. Phys. 41 (2008) 172002. https://doi.org/10.1088/0022-3727/41/17/172002.
[53] K. Nishii, D.A. Levin, Three-Dimensional Kinetic Simulations of Carbon Backsputtering in Vacuum Chambers from Ion Thruster Plumes, Journal of Propulsion and Power 0 (n.d.) 1–15. https://doi.org/10.2514/1.B39194.
[54] G.Z. Li, T.S. Matlock, D.M. Goebel, C.A. Dodson, C.S.R. Matthes, N.M. Ghoniem, R.E. Wirz, In situ plasma sputtering and angular distribution measurements for structured molybdenum surfaces, Plasma Sources Sci. Technol. 26 (2017) 065002. https://doi.org/10.1088/1361-6595/aa6a7d.
[55] G.Z. Li, R.E. Wirz, Persistent Sputtering Yield Reduction in Plasma-Infused Foams, Phys. Rev. Lett. 126 (2021) 035001. https://doi.org/10.1103/PhysRevLett.126.035001.

CHAPTER 2: LIMITATIONS AND OPPORTUNITIES OF SPUTTERING MODELS ACROSS SCALES – A HISTORICAL PERSPECTIVE

2.1 From Theoretical to Binary Collisions Approximation

As aforementioned, sputtering is an atomistic process where target atoms are ejected by the impact of energetic ions, presumably from atomic collision cascades. In 1969, Sigmund [1] developed the first theory of sputtering of amorphous and polycrystalline targets from the general Boltzmann transport equation, assuming a random slowing down of the primary ion and all recoiling atoms in an infinite medium, and arrived at an integrodifferential equation [1] which can be integrated to obtain an expression for the sputter yield, Y, under the bombardment of energetic ions at normal ion incidence angle

$$Y(E_i) = \frac{0.042\alpha}{U} S_n(E_i) \tag{2.1}$$

where U is the surface binding energy (often taken as the sublimation energy of the target), $S_n(E_i)$ is the nuclear stopping power of the ion in the target at the ion incidence energy E_i (Fig. 1.3a), and α is the efficiency of the conversion from stopping power into energy deposited to the target atoms in the surface region of the target [2]. The challenge with the Sigmund formulation in (2.1) has always been the determination of α. There have been many attempts to compile experimental sputtering data from the literature [3,4] and to fit α in (2.1) to these available experimental data, with U assumed to be the sublimation energy of the target material. This has led to some very complex expressions for α. In addition, several assumptions in the derivation of (2.1), notably a linear collision cascade as the dominant sputtering mechanism, suggest that the equation should not hold for very heavy or light ions, nor for low energies or grazing ion incidence angles [5]. More importantly, Sigmund also assumed $Y = 0$ only when $E_i = 0$ but the sputtering yield should

decrease sharply towards low ion energies since target atoms do not have sufficient energy to overcome the surface barrier energy.

To correct for these limitations, Bohdansky [6] and others [2,3,7] have introduced the concept of threshold energy, E_{th}, as the minimum ion energy required to cause sputtering. Mathematically, Bohdansky [6] enforced $Y = 0$ when $E_i = E_{th}$ and reformulated Sigmund model

$$Y(E_i) = \boldsymbol{Q} S_n(E_i)\left(1 - \left(\frac{\boldsymbol{E_{th}}}{E_i}\right)^{\frac{2}{3}}\right)\left(1 - \frac{\boldsymbol{E_{th}}}{E_i}\right)^2 \tag{2.2}$$

where Q, E_{th} are fitting parameters with Q now encompassing the contributions of both α and U in (2.1). Later, Yamamura [7] generalized (2.2) to better fit the experiment data with an additional parameter μ proposed to account for the interatomic potential

$$Y(E_i) = \boldsymbol{Q} S_n(E_i)\left(1 - \sqrt{\frac{\boldsymbol{E_{th}}}{E_i}}\right)^{\boldsymbol{\mu}} \tag{2.3}$$

Later studies by Matsunami [3] fitted Q, E_{th}, μ to a vast experimental sputter yield data available at the time (1984) and observed an averaged dependence of $\frac{E_{th}}{U_s}$ on the mass ratio of the ion and target with $\mu = 2.8$ while Q was notably a function of the target's mass [3]. However, this new fitting by Matsunami [3] still significantly overestimated the sputter yield at low ion energies, prompting Eckstein & Preuss [8] to introduce an additional fitting parameter, λ

$$Y(E_i) = \boldsymbol{Q} S_n(E_i)\frac{\left(1 - \frac{\boldsymbol{E_{th}}}{E_i}\right)^{\boldsymbol{\mu}}}{\boldsymbol{\lambda} + \left(1 - \frac{\boldsymbol{E_{th}}}{E_i}\right)^{\boldsymbol{\mu}}} \tag{2.4}$$

Unfortunately, the phenomenological nature of these parameters imply they were no longer dependent on general material properties but were specific to a given ion-target system. For instance, Eckstein suggested $\lambda = 0.561, Q = 0.272, \mu = 1.7795,$ and $E_{th} = 10.94$ for (He, Be) ion-target system, and demonstrated an average error of 26.2% from experiment data for ion incidence energies of $10 - 10^4$ eV [9]. With this gradual introduction of semi-empirical (and

arguably non-physical) fitting parameters, the theoretical sputtering theory of Sigmund in (2.1) has now lost its generality.

At the oblique ion incidence angle θ_i, sputtering models generally introduce a correction factor, $K(\theta_i)$, multiplicative to the sputter yield at normal impact angle, $Y(E_i)$ from (2.1) – (2.4)

$$Y(E_i, \theta_i) = Y(E_i)\, K(\theta_i) \tag{2.5}$$

However, $K(\theta_i)$ suffers from similar challenges as $Y(E_i)$; While the original expression for $K(\theta_i)$ was physics-based, assuming linear collision cascades as an extension to the expressions by Sigmund and Yamamura [7], a myriad of phenomenological parameters was later introduced that necessitated refitting to any new ion-target systems. The expression for $K(\theta_i)$ in (2.4) [10] is now

$$K(\theta_i) = (cos\theta_i^{\boldsymbol{c}})^{-\boldsymbol{f}} \exp(\boldsymbol{b}(1 - (cos\theta_i^{\boldsymbol{c}})^{-1})) \tag{2.6}$$

and consists of the three additional phenomenological parameters, f, b, c, to a total of seven fitting parameters (combining (2.4) and (2.6)).

Since these parameters are obtained specifically for an ion-target system, the quality and the usefulness for the sputtering model strongly depends on the availability of the sputtering properties of the ion-target system of interest. When the sputtering yield data is abundant and consistent, meaning the sputtering yield measurements across different laboratories have small scatter, the fitting routine serves as a higher-order interpolation between the data points. In contrast, when the data is either conflicted or sparse, as depicted for Xe → C [11] with $\theta_i = 0°$ and $\theta_i = 30°$, respectively, in Fig. 2.1, the fitted parameters, and subsequently the sputtering yield model, have a huge uncertainty band spanning 2 or 3 orders of magnitude (grey band in Fig. 2.1). The sputter yield data under low energy noble gas ion bombardment of relevance to EP, as mentioned in Section 1.3, are both inconsistent and lacking, to the point of having more fitting parameters than available sputtering yield data points ($\theta_i = 75°$ in Fig. 2.1).

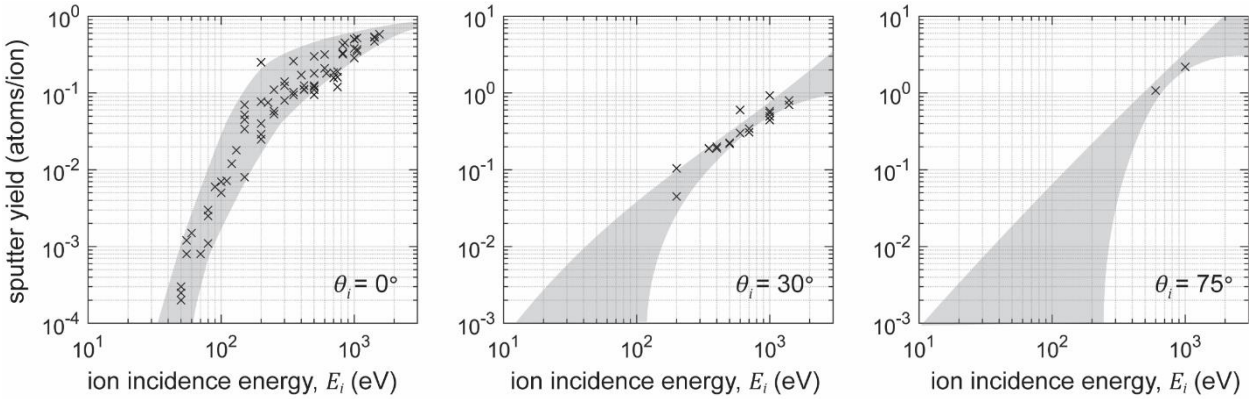

Figure 2.1: Available experiment sputter yield measurements (cross) conducted at various laboratories for Xe → C [11] at different ion incidence angle, θ_i. Grey band shows the uncertainty from fitting Eckstein & Preuss [8, 10] formulation.

These trends and problems are exacerbated for other essential sputter properties, specifically, the ejection angle and distributions of carbon sputterants. Based on the linear cascade assumption, Sigmund [1] suggested that the ejecta angle distribution is independent of ion incidence angle, with profiles that are azimuthally symmetric and approximately diffuse in shape, corresponding to cosine-like profiles. Several experimental efforts [12–15] have shown that under a normal ion incidence angle, the differential yield profile is more under-cosine at lower ion incidence energy but becomes increasingly over-cosine for higher ion energies (Fig. 2.2a). Under an oblique ion incidence angle, the profile becomes more asymmetric with increased distribution of sputterants in the forward direction. Based on these observations, studies have proposed several phenomenological expressions to account for the differential yield profile of the sputterants [13,16], including a linear combination of a polynomial series with a cosine basis [13] or a complex trigonometric expression [16]. For these models, the additional two or three empirical parameters are specific to the ion-target systems, as well as the ion incidence energy and angle.

In the case of the energy distribution of the sputterants, the Sigmund-Thompson [17,18] expression is widely adopted and is based on linear cascade theory,

$$f(E_t) = \frac{E_t}{(E_t + U_s)^{3+2m}} \tag{2.7}$$

where E_t is the sputterant energy, and m is the potential parameter, with $m = \frac{1}{6}$ resulting in the best agreement with computational simulations [18,19]. While equation (2.7) gives the most probable E_t with reasonable agreement to experiment data [18] (Fig. 2.2b), it extends the possible energy distribution to infinity. Eckstein [18] proposed a cut-off energy, E_c, representing the maximum transferrable energy from the ion to the target under perfect linear collisions, resulting in the augmented energy distribution expression

$$f(E_t) = \frac{4\,M_t}{M_i\left(1+\frac{M_t}{M_i}\right)^2} E_i \tag{2.8}$$

Nevertheless, Eckstein [18] recognized that the cut-off energy could be considerably lower than (2.8). In addition, (2.7) - (2.8) further suggests that the emission energy distribution is independent of the ion incidence angle, which is counter-intuitive. Resolving these angular and energy distribution issues requires additional experiment data, which remains notoriously difficult and expensive to obtain under low energy noble gas ion bombardment, even today [11,20].

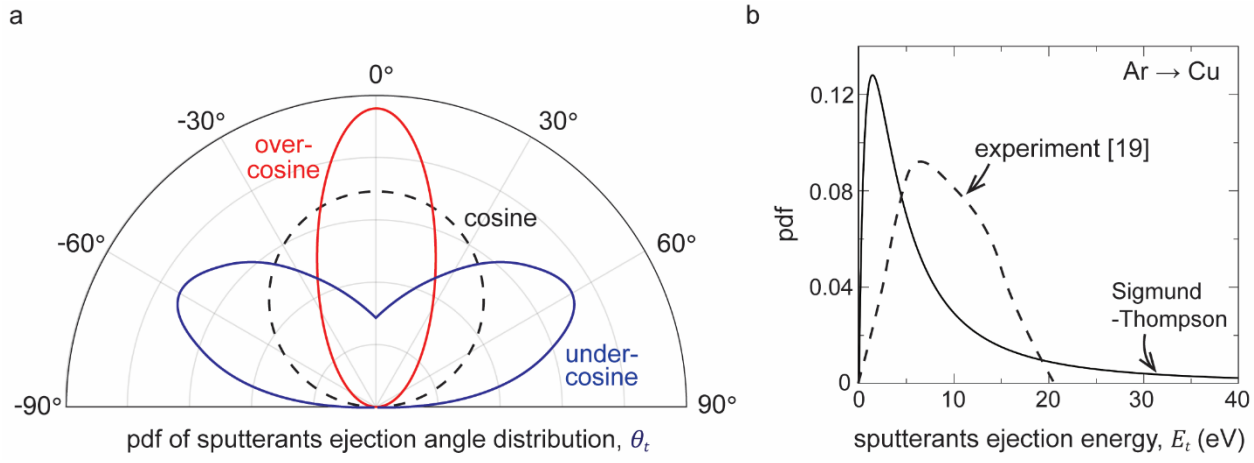

Figure 2.2: **(a)** Schematics demonstrate possible probability distribution function for sputterants ejection angle distribution, the convention for θ_t follows Fig. 1.3 **(b)** Probability distribution function for sputterants ejection energy of Cu under 10 keV Ar ion bombardment, comparing Sigmund-Thompson function (equation 2.7) and experiment measurements [19]

Many studies have resorted to the use of computational approaches to overcome the lack of experimental data and to obtain better empirical fittings for the semi-empirical modes in (2.4) – (2.8). The most popular sputtering codes are mainly based on Monte Carlo simulations of binary collision approximation (BCA) [21], a process describing a sequence of independent binary collisions with sample atoms, with a straight path between the collisions. The most widely used simulation software is The Stopping and Range of Ions in Matter (SRIM) [22], which is based on The Range of Ions in Matter (TRIM) code and uses the Ziegler-Biersack-Littmark (ZBL) [23] interaction potential to solve the scattering integral in BCA. In addition, TRIM – Dynamic (TRIDYN) [24] and Static-Dynamic TRIM with Sequential and Parallel processing (SDTrimSP) [25] are similar to TRIM but use the Krypton-Carbon (KC) interaction potential [26], taking into account of stoichiometry changes due to incorporation of projectile atoms, atomic mixing, and preferential sputtering [25]. These BCA codes not only predict the sputter yield but also provide differential yield and energy distributions of the sputterants, and have demonstrated good agreement with several experiment measurements at ion energies beyond 1 keV [27].

While ZBL (and to some extent, KC potential) is suitable for simulating the interactions between a noble gas ion and any general target material, BCA methods neglect any interactions (e.g. chemical bonding) between the target atoms, which is especially important for $E_i < 1$ keV, resulting in significant differences in the predicted sputter properties when compared to experiments [27–29]. SDTrimSP [25] attempted to account for these interactions by including additional energy loss terms during the collision process, particularly the surface binding energy, vacancy formation energy, and atomic displacement energy. Even though these quantities are well defined for an atomistic system, quantifying them in experiment are impossible and is often chosen to best fit the experiment data. For instance, vacancy formation energy defines the energy to form

a vacancy inside the substrate, which cannot be measured experimentally, and thus is set at 3 eV by default for monoatomic amorphous solid for best sputter yield fit [22,27,29]. While one can certainly quantify these parameters with Molecular Dynamics, the procedure is arbitrary and not yet standardized [30]. Therefore, the results of these BCA simulation codes tend to lose their first-principle characteristics.

2.2 Investigating Sputtering Physics via Molecular Dynamics

From a computational perspective, Molecular Dynamics (MD) simulations provide a promising path forward. Unlike BCA, MD simulations account for the complicated many-body interactions among the target atoms. Specifically, MD tracks the trajectories of the atoms based on Newton's Second Law with the force field computed from a pre-described interatomic potential (IAP) [31]. Here, IAP represents the potential energy of a system of atoms as a function of its spatial positions, and is calibrated based on experiment measurements and/or density functional theory of intrinsic bulk properties (cohesive energy, elastic constant, lattice constant, phonon dispersion, etc.) [32]. Often, most well-utilized IAP are transferable, i.e. able to describe materials properties that are clearly different from those it was fitted to by ensuring the correct representation of chemical bonding, validation of atomistic structures (e.g., body-centered-cubic, dislocation, tetrahedral bond configuration, etc.), as well as interpretability of all parameters [32]. With well-calibrated IAP, MD simulations are thus a powerful tool applicable to a wide range of atomistic problems. One of the most common MD codes is the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS), developed by Sandia National Laboratory [33]. All the MD simulations in this thesis are performed using LAMMPS, with the appropriate target material potential selected from the National Institute of Standards and Technology Interatomic Potentials Repository.

Since its first application towards sputtering problems in the ion energy range of keV in 1986 [34], MD simulations have been widely adopted to study material sputtering. For instance, Kim et. al. used MD simulations to uncover the underpinning nano-plasticity mechanisms in face-centered-cubic metals under low-energy noble gas ion bombardments that consistently clarified previously unexplainable experimental observations. However, running sputtering simulations with MD at low ion energies residing between the sputtering threshold energy and ~2 keV is extremely challenging and computationally expensive [34]. This is because typical atomistic systems comprise of large number of particles. Updating each atomic position per time step requires calculations of the interaction force with every atom within a cut-off radius. To make matters worse, sputtering simulations require very small timesteps in order to capture the dynamics of the fast moving (supersonic) ions and impacted target atoms. Furthermore, many impact events need to be simulated in order to obtain statistically significant sputter properties [27,35,36]. As a result, there are no comprehensive study, to my knowledge, characterizing the sputter properties from MD simulations for a wide range of ion-target combinations, ion incidence energies and angles. Surprisingly, no systematic MD studies for Xe-C systems relevant to EP currently exists, which would be critical in reducing the uncertainties in the sputter yield experimental measurements. Leveraging on the rapid development of high-performance computing, my thesis research seeks to *conduct large-scale massively parallel MD simulations to simulate the ion bombardment of xenon-carbon substrates and various other ion-target systems, to resolve the conflicting experimental measurements relevant to EP.*

2.3 Bridging Atomistic and Experiment Scales

Both semi-empirical and computational approaches previously described rely on the sputter mechanisms at the atomistic scales where the sputter properties are obtained from the collision

cascades between the ion and the target materials. These events occur within the span of a few nanometers [28,36], effectively neglecting the higher order length and time scales effects that are often associated with sputtering experiments [37–39]. Notably, surface morphology effects are well-known to influence the sputtering properties. Specifically, the undulating surface at wavelengths of several micrometers can (a) change the local incidence angle at the impact site, which also alters the trajectory of the sputterants, (b) cause deposition or the emission of secondary sputterants, and (c) partially shield a portion of the surface from incoming ions leading to a shadowing or masking effect. At longer time-scales, these surface morphologies evolve with long exposure to the ion beam, which is typical for EP systems [40,41] where the surface erosion rate depends on the instantaneous topography. Because MD simulations cannot account for these surface morphology effects at higher length- and time-scales, the challenge is in extending the scales of MD to elucidate the sputter properties relevant and comparable to experiments.

The most common measured quantity from sputtering experiments is the surface erosion rate. Early theoretical studies of sputtering erosion generally assume that the sputtering yield is independent of surface curvature effects [1]. Later analytical developments by Bradley and Harper (BH) consider the effects of surface instability caused by curvature dependence, where the sputter yield is greater in a trough than at a crest through a damage transfer region of size a [42,43]. With an unperturbed planar (flat) surface with erosion rate $v(E_i, \theta_i)$, BH expression for a 1D surface of height h located at x and time t under flux f, neglecting thermal activated surface diffusion, is

$$\frac{\partial h}{\partial t} \cong v + \frac{\partial v}{\partial t}\frac{\partial h}{\partial x} + \frac{fa}{n} Y(E_i, \theta_i)\Gamma_1(E_i, \theta_i)\frac{\partial^2 h}{\partial x^2} \tag{2.8}$$

where n is the number of atoms per unit volume in the amorphous solid and $\Gamma_1(E_i, \theta_i)$ is the size of the energy transfer region, which is dependent on the local sputter yield $Y(E_i, \theta_i)$ and curvature of the surface [42]. Since the erosion rate of a flat surface $v(E_i, \theta_i)$ has a one-to-one relationship

with the sputter yield $Y(E_i, \theta_i)$, equation (2.8) essential bridges the discrete atomistic quantities of sputtering to a continuum quantity representing erosion of surface undulations over time, $\frac{\partial h}{\partial t}, \frac{\partial h}{\partial x}$. While the current closed-form solution for equation (2.8) does not exist and is often numerically solved using Monte Carlo [44], the numerical solutions have successfully predicted a wide range of nanoscale surface structures induced by ion bombardment – from nanoripples to quantum dots [45,46]. However, curvature effects Γ_1 are only apparent at high ion energies ($> \sim 5$ keV) and at very small wavelength ($< \sim 100$ nm) where the impact-induced damage can span across multiple nanoripples [46–48]. The extension of BH model to the microscale surface features has found some success when applied to silicon target materials [49–51]. Regardless, BH expression in (2.8) neglects the shadowing effects [52], re-deposition and secondary sputtering [25,39]. It also does not provide two other important quantities for plasma engineering applications: differential yield (ejection angle) and energy distribution of the sputterants.

From a computational perspective, the elemental sputter properties obtained at the atomistic scales can be homogenized and represented as the sputtering properties of a surface element, where multiple elements can be used to represent a microscale surface feature. For instance, Hobler [53], Boxleitner [45], and numerous others [45,50,53–57] have extended BCA to quantify surface evolution effects on the sputter yield as a function of ion fluence. In these studies, a Monte Carlo approach is undertaken by conducting BCA simulations element-wise and the element sizes are updated (relaxed and receded) with appropriate assumptions depending on the material systems after a certain number of bombardments [24,25]. This process successfully simulates the surface evolution at length- and time-scales of relevance to experiments, thus allowing for sputter properties (sputter yield, sputterants energy, differential yield) to be obtained for various surface topographies and material systems ranging from tungsten to silicon [51,58].

However, BCA, as described in Section 2.1, is well-known to break down at low ion energies < $\sim 1-2$ keV of interest to EP applications.

In view of the limitations of BCA at low ion energies, other approaches, including the view factor model or cell model [59–61], have been proposed. However, these models also have their own limitations. For example, the total mass in the view factor model is not conserved [61], while the cell model assumes a mirror-diffuse reflection of the carbon sputterants, which contradicts the results of both MD simulations [28] and experiments conducted by Williams et. al. [13]. Regardless, the homogenization framework proposed in [25] can certainly be based on the sputter properties obtained from MD rather than MC simulations. In this regard, *my thesis research seeks to elucidate the evolving surface topology effects on the sputtering properties of Xe on carbon by incorporating high-fidelity atomistic MD data in MC models.*

2.4 Harnessing High-fidelity Simulation Data in the Age of Artificial Intelligence and Machine Learning

In the era of rapid engineering and prototyping, there is an increasing need for an engineering model that can produce fast and reliable results. To ensure reliability, these models need to be grounded on the underlying physics that is well validated with experimental observations. MD simulations have a strong physical basis and are very effective in elucidating the sputtering mechanisms and quantifying the sputtering properties. However, such simulations are computationally expensive. A way to overcome this issue is to create a large database of MD sputtering data of known ion-target systems and extract from this massive data a general physics-based model that is applicable *all* ion/target systems. The goal here is identifying an Eckstein & Preuss [5,8,18] type of sputter yield model (Section 2.1), but with more general parameters that are applicable to different ion-targets.

Data-driven approaches based on artificial intelligence and machine learning offer a promising path forward. In the field of data science, artificial intelligence (AI) relates to a set of algorithms that mimic certain human characteristics and learn from a series of tasks to take actions that maximize the chances of achieving define goals. For instance, Genetic Algorithm (GA), inspired by natural selection process, "learns" from observations through evolutionary operators such as mutation, crossover, and selection and provides an optimized solutions to a search-based problem [62]. Another subset of AI is Machine Learning (ML), where a statistical algorithm is utilized to learn from data and generalize to unseen data to perform tasks without explicit instructions. As of today, the most popular ML model is Large Language Model (LLM), which naturally coincides with the rise of ChatGPT 3.5. LLMs are deep learning models that are pre-trained on vast amounts of data, where the underlying structures are sets of neural networks with encoding, decoding, and short-long term memory capabilities. The two most common neural networks are artificial neural networks (ANNs) for scalar data, and convolution neural networks (CNNs) for image data [63,64]. The use of GA and ANNs are explored in Chapter 5 and 6 of this thesis respectively.

The challenge to using AI-ML to enable sputter predictions for generalized ion-target combinations lie in the ability to construct a comprehensive dataset from MD that is tractable, but large enough to ensure adequate learning generalizability. A secondary challenge is to establish which sputtering parameters (cohesion energy, nuclear stopping power, damage zone parameters, etc.) are relevant in the prediction of the sputter properties. *My thesis research seeks to construct an extensive MD dataset covering a wide range of ion-target systems, and to study the feasibility of using MD data to arrive at a generalized sputtering model using AI-ML.*

2.5 Thesis Objectives

A reliable model to predict the sputtering properties of different ion-target combinations has been an on-going pursuit for over half a century, yet there is still significant shortcomings of current approaches in elucidating the sputtering properties of materials in the low ion energy regime spanning the sputtering threshold energy and ~2 keV. For instance, sputtering measurements are notoriously challenging for the highly sputter resistant carbon-based targets. The overarching goal of this thesis is to harness multi-scale simulations and data-driven approaches to establish a framework for the construction of constitutive laws governing low energy sputtering of target substrates across scales. For simplicity, the primary focus will be on the Xe ion bombardment of carbon substrates (Chapters 3 and 4), which is relevant to EP systems, but attempts will be made to generalize the sputtering response to other ion-target combinations (Chapters 5 and 6).

The first part of my thesis addresses the uncertainties in the sputter yield measurements for Xe → C reported in the literature with MD simulations, and to elucidate the fundamental sputtering mechanisms and associated structural evolution with ion bombardment (Chapter 3). The second part of my thesis upscales the MD simulation results to a Monte Carlo (MC) model to investigate the microscale processes responsible for surface morphology evolution, as well as the effects on the sputtering properties. Based on the results of the MD-MC model for Xe-C, new parameters for engineering sputter yield equation in (2.4) - (2.6) are proposed. Furthermore, the predicted differential yield (ejection angle) of the sputterants is fitted to a two-mode Gaussian distribution, while the sputterant energy data is fitted to a log-normal distribution.

While the results in Chapters 3 and 4 are specific to Xe bombardment of C targets, the framework can be extended to other ion-target combinations. However, this will be computationally expensive. As a first step towards developing a generalized sputtering model for

low ion energies, applicable to different ion-target combinations, the third part of my thesis (Chapter 5) focuses on extracting a data-driven formulation for the sputtering threshold energy, E_{th}, discussed in Section 2.1. Massive amounts of E_{th} data for different noble gas ion – monoatomic target combinations are generated from MD simulations, and symbolic regression through evolutionary algorithms is used to uncover a new data-driven formulation for E_{th}. Finally, Chapter 6 concludes this thesis by discussing possible future works, aimed at further developing a universal low ion energy sputtering yield model, along with proposed efforts to quantify the sticking coefficient of carbon contaminants for space EP applications.

Note that the time dependence of the sputter properties presented in this thesis is presented in terms of the ion fluence (ions per unit area). Given an ion flux of a typical experiments set up [11-13], *one second of testing time usually corresponds to an ion fluence of* 10^{14} *ions/cm*^{2}.

2.6 References

[1] P. Sigmund, Sputtering by ion bombardment theoretical concepts, in: R. Behrisch (Ed.), Sputtering by Particle Bombardment I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981: pp. 9–71. https://doi.org/10.1007/3540105212_7.

[2] Y. Yamamura, H. Tawara, Energy Dependence of Ion-Induced Sputtering Yields from Monoatomic Solids at normal incidence, Atomic Data and Nuclear Data Tables 62 (1996) 149–253. https://doi.org/10.1006/adnd.1996.0005.

[3] N. Matsunami, Y. Yamamura, Y. Itikawa, N. Itoh, Y. Kazumata, S. Miyagawa, K. Morita, R. Shimizu, H. Tawara, Energy Dependence of Ion-Induced Sputtering Yields of Monatomic Solids, Atomic Data and Nuclear Data Tables 31 (1984) 1. https://doi.org/10.1016/0092-640X(84)90016-0.

[4] R.P. Webb, I. Wilson, Problems using the Sigmund formula for the calculation of sputtering yields, Vacuum 39 (1989) 1163–1165. https://doi.org/10.1016/0042-207X(89)91113-5.

[5] W. Eckstein, C. Garciá-Rosales, J. Roth, J. László, Threshold energy for sputtering and its dependence on angle of incidence, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 83 (1993) 95–109. https://doi.org/10.1016/0168-583X(93)95913-P.

[6] J. Bohdansky, A universal relation for the sputtering yield of monatomic solids at normal ion incidence, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 2 (1984) 587–591. https://doi.org/10.1016/0168-583X(84)90271-4.

[7] Y. Yamamura, S. Shindo, An empirical formula for angular dependence of sputtering yields, Radiation Effects 80 (1984) 57–72. https://doi.org/10.1080/00337578408222489.
[8] W. Eckstein, R. Preuss, New fit formulae for the sputtering yield, Journal of Nuclear Materials 320 (2003) 209–213. https://doi.org/10.1016/S0022-3115(03)00192-2.
[9] C. García-Rosales, W. Eckstein, J. Roth, Revised formulae for sputtering data, Journal of Nuclear Materials 218 (1995) 8–17. https://doi.org/10.1016/0022-3115(94)00376-9.
[10] Q. Wei, K.-D. Li, J. Lian, L. Wang, Angular dependence of sputtering yield of amorphous and polycrystalline materials, J. Phys. D: Appl. Phys. 41 (2008) 172002. https://doi.org/10.1088/0022-3727/41/17/172002.
[11] J.E. Polk, A critical review and meta-analysis of xenon-on-carbon sputter yield data, Journal of Applied Physics 135 (2024) 040701. https://doi.org/10.1063/5.0180720.
[12] A.P. Yalin, J.D. Williams, V. Surla, K.A. Zoerb, Differential sputter yield profiles of molybdenum due to bombardment by low energy xenon ions at normal and oblique incidence, J. Phys. D: Appl. Phys. 40 (2007) 3194. https://doi.org/10.1088/0022-3727/40/10/025.
[13] J. Williams, M. Johnson, D. Williams, Differential Sputtering Behavior of Pyrolytic Graphite and Carbon-Carbon Composite Under Xenon Bombardment, in: 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, American Institute of Aeronautics and Astronautics, Fort Lauderdale, Florida, 2004. https://doi.org/10.2514/6.2004-3788.
[14] T.K. Chini, M. Tanemura, F. Okuyama, Angular distribution of sputtered Ge atoms by low keV Ar+ and Ne+ ion bombardment, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 119 (1996) 387–391. https://doi.org/10.1016/0168-583X(96)00366-7.
[15] G.K. Wehner, D. Rosenberg, Angular Distribution of Sputtered Material, Journal of Applied Physics 31 (1960) 177–179. https://doi.org/10.1063/1.1735395.
[16] Z.L. Zhang, L. Zhang, Anisotropic angular distribution of sputtered atoms, Radiation Effects and Defects in Solids 159 (2004) 301–307. https://doi.org/10.1080/10420150410001724495.
[17] M.W. Thompson, II. The energy spectrum of ejected atoms during the high energy sputtering of gold, The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics 18 (1968) 377–414. https://doi.org/10.1080/14786436808227358.
[18] W. Eckstein, Energy distributions of sputtered particles, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 18 (1986) 344–348. https://doi.org/10.1016/S0168-583X(86)80056-8.
[19] A.E. Ieshkin, A.V. Nazarov, A.A. Tatarintsev, D.S. Kireev, A.D. Zavilgelsky, A.A. Shemukhin, V.S. Chernysh, Energy distributions of the particles sputtered by gas cluster ions. Experiment and computer simulation, Surface and Coatings Technology 404 (2020) 126505. https://doi.org/10.1016/j.surfcoat.2020.126505.
[20] E. Oyarzabal, R.P. Doerner, M. Shimada, G.R. Tynan, Carbon atom and cluster sputtering under low-energy noble gas plasma bombardment, Journal of Applied Physics 104 (2008) 043305. https://doi.org/10.1063/1.2968549.
[21] J.F. Ziegler, The stopping of energetic ions in solids, Nuclear Instruments and Methods 168 (1980) 17–24. https://doi.org/10.1016/0029-554X(80)91225-2.
[22] J.F. Ziegler, M.D. Ziegler, J.P. Biersack, SRIM – The stopping and range of ions in matter (2010), Nuclear Instruments and Methods in Physics Research Section B: Beam

Interactions with Materials and Atoms 268 (2010) 1818–1823. https://doi.org/10.1016/j.nimb.2010.02.091.
[23] J.F. Ziegler, J.P. Biersack, The Stopping and Range of Ions in Matter, in: D.A. Bromley (Ed.), Treatise on Heavy-Ion Science: Volume 6: Astrophysics, Chemistry, and Condensed Matter, Springer US, Boston, MA, 1985: pp. 93–129. https://doi.org/10.1007/978-1-4615-8103-1_3.
[24] J. Drobny, A. Hayes, D. Curreli, D.N. Ruzic, F-TRIDYN: A Binary Collision Approximation code for simulating ion interactions with rough surfaces, Journal of Nuclear Materials 494 (2017) 278–283. https://doi.org/10.1016/j.jnucmat.2017.07.037.
[25] P.S. Szabo, D. Weichselbaum, H. Biber, C. Cupak, A. Mutzke, R.A. Wilhelm, F. Aumayr, Graphical user interface for SDTrimSP to simulate sputtering, ion implantation and the dynamic effects of ion irradiation, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 522 (2022) 47–53. https://doi.org/10.1016/j.nimb.2022.04.008.
[26] J.T. Yim, Yim, J.T. (2017). A Survey of Xenon Ion Sputter Yield Data and Fits Relevant to Electric Propulsion Spacecraft Integration., in: 35th International Electric Propulsion Conference, 2017.
[27] H. Hofsäss, K. Zhang, A. Mutzke, Simulation of ion beam sputtering with SDTrimSP, TRIDYN and SRIM, Applied Surface Science 310 (2014) 134–141. https://doi.org/10.1016/j.apsusc.2014.03.152.
[28] H. Tran, H.B. Chew, Surface morphology and carbon structure effects on sputtering: Bridging scales between molecular dynamics simulations and experiments, Carbon 205 (2023) 180–193. https://doi.org/10.1016/j.carbon.2023.01.015.
[29] J.P. Greene, J. Nemanich, G.E. Thomas, S.L. Schiel, Noble gas sputtering calculations using TRIM, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 397 (1997) 91–98. https://doi.org/10.1016/S0168-9002(97)00578-0.
[30] X. Li, X. Zhang, Y. Xu, G. Lei, S. Liu, H. Li, Z. Cui, Y. Zhu, J. Hu, S. Geng, X. Chen, H. Liu, X. Wang, J. Huang, H. Liu, J. Cheng, J. Shen, H. Lan, C. Tang, Molecular dynamics study of surface binding energy and sputtering in W-V alloys, Fusion Engineering and Design 195 (2023) 113971. https://doi.org/10.1016/j.fusengdes.2023.113971.
[31] Q. Liu, L. Li, Y.-R. Jeng, G. Zhang, C. Shuai, X. Zhu, Effect of interatomic potentials on modeling the nanostructure of amorphous carbon by liquid quenching method, Computational Materials Science 184 (2020) 109939. https://doi.org/10.1016/j.commatsci.2020.109939.
[32] J. Fischer, M. Wendland, On the history of key empirical intermolecular potentials, Fluid Phase Equilibria 573 (2023) 113876. https://doi.org/10.1016/j.fluid.2023.113876.
[33] A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in 't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, R. Shan, M.J. Stevens, J. Tranchida, C. Trott, S.J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Computer Physics Communications 271 (2022) 108171. https://doi.org/10.1016/j.cpc.2021.108171.
[34] H.M. Urbassek, Molecular-dynamics simulation of sputtering, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 122 (1997) 427–441. https://doi.org/10.1016/S0168-583X(96)00681-7.

[35] J.T. Yim, M.L. Falk, I.D. Boyd, Modeling low energy sputtering of hexagonal boron nitride by xenon ions, Journal of Applied Physics 104 (2008) 123507. https://doi.org/10.1063/1.2987090.
[36] S.-P. Kim, H.B. Chew, E. Chason, V.B. Shenoy, K.-S. Kim, Nanoscale mechanisms of surface stress and morphology evolution in FCC metals under noble-gas ion bombardments, Proc. R. Soc. A. 468 (2012) 2550–2573. https://doi.org/10.1098/rspa.2012.0042.
[37] R. Deltschew, M. Tartz, V. Plicht, E. Hartmann, H. Neumann, Sputter characteristics of carbon-carbon compound material, in: 27th International Electric Propulsion Conference, 2001.
[38] M. Tartz, Pyrolytic Graphite and Carbon-carbon Sputter Behaviour Under Xenon Ion Incidence, in: 29th International Electric Propulsion Conference, 2005: p. 10.
[39] R. Kolasinski, J. Polk, D. Goebel, L. Johnson, Carbon Sputtering Yield Measurements at Grazing Incidence, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, Sacramento, California, 2006. https://doi.org/10.2514/6.2006-4337.
[40] N.P. Brown, M.L.R. Walker, Review of Plasma-Induced Hall Thruster Erosion, Applied Sciences 10 (2020) 3775. https://doi.org/10.3390/app10113775.
[41] R.B. Lobbia, J.E. Polk, R.R. Hofer, V.H. Chaplin, B. Jorns, Accelerating 23,000 hours of Ground Test Backsputtered Carbon on a Magnetically Shielded Hall Thruster, in: AIAA Propulsion and Energy 2019 Forum, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-3898.
[42] R.M. Bradley, J.M.E. Harper, Theory of ripple topography induced by ion bombardment, Journal of Vacuum Science & Technology A 6 (1988) 2390–2395. https://doi.org/10.1116/1.575561.
[43] P.D. Shipman, R.M. Bradley, Theory of nanoscale pattern formation induced by normal-incidence ion bombardment of binary compounds, Phys. Rev. B 84 (2011) 085420. https://doi.org/10.1103/PhysRevB.84.085420.
[44] R.M. Bradley, G. Hobler, Sputter yields of surfaces with nanoscale textures: Analytical results and Monte Carlo simulations, Journal of Applied Physics 133 (2023) 065303. https://doi.org/10.1063/5.0137324.
[45] O. Bobes, K. Zhang, H. Hofsäss, Ion beam induced surface patterns due to mass redistribution and curvature-dependent sputtering, Phys. Rev. B 86 (2012) 235414. https://doi.org/10.1103/PhysRevB.86.235414.
[46] S. Habenicht, W. Bolse, K.P. Lieb, K. Reimann, U. Geyer, Nanometer ripple formation and self-affine roughening of ion-beam-eroded graphite surfaces, Phys. Rev. B 60 (1999) R2200–R2203. https://doi.org/10.1103/PhysRevB.60.R2200.
[47] S. Habenicht, Morphology of graphite surfaces after ion-beam erosion, Phys. Rev. B 63 (2001) 125419. https://doi.org/10.1103/PhysRevB.63.125419.
[48] S. Habenicht, K.P. Lieb, Diffusion and roughening during ion beam erosion of graphite surfaces, AIP Conference Proceedings 576 (2001) 951–954. https://doi.org/10.1063/1.1395461.
[49] M.P. Harrison, D.A. Pearson, R.M. Bradley, Emergence and detailed structure of terraced surfaces produced by oblique-incidence ion sputtering, Phys. Rev. E 96 (2017) 032804. https://doi.org/10.1103/PhysRevE.96.032804.
[50] A.S. Shumilov, I.I. Amirov, Profile of Silicon Nanostructures in Plasma Sputtering, J. Synch. Investig. 14 (2020) 935–943. https://doi.org/10.1134/S1027451020050195.

[51] I. Bizyukov, A. Mutzke, R. Schneider, J. Davis, Evolution of the 2D surface structure of a silicon pitch grating under argon ion bombardment: Experiment and modeling, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 268 (2010) 2631–2638. https://doi.org/10.1016/j.nimb.2010.06.035.
[52] K.-K. Hung, I. Chasiotis, Control of Surface Wrinkling through Compliant Nanostructured Interfaces, Advanced Materials Interfaces 9 (2022) 2101583. https://doi.org/10.1002/admi.202101583.
[53] G. Hobler, D. Kovač, Dynamic binary collision simulation of focused ion beam milling of deep trenches, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 269 (2011) 1609–1613. https://doi.org/10.1016/j.nimb.2010.12.076.
[54] I. Bizyukov, A. Mutzke, R. Schneider, A.M. Gigler, K. Krieger, Morphology and changes of elemental surface composition of tungsten bombarded with carbon ions, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 266 (2008) 1979–1986. https://doi.org/10.1016/j.nimb.2008.03.211.
[55] W. Boxleitner, G. Hobler, FIBSIM – dynamic Monte Carlo simulation of compositional and topography changes caused by focused ion beam milling, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 180 (2001) 125–129. https://doi.org/10.1016/S0168-583X(01)00406-2.
[56] D. Kunder, E. Baer, M. Sekowski, P. Pichler, M. Rommel, Simulation of focused ion beam etching by coupling a topography simulator and a Monte-Carlo sputtering yield simulator, Microelectronic Engineering 87 (2010) 1597–1599. https://doi.org/10.1016/j.mee.2009.11.007.
[57] M.V. Skachkov, Application of the Monte Carlo Method for Simulation of Pattern Formation by Ion-Beam Sputtering of Amorphous Bodies, Math Models Comput Simul 10 (2018) 551–563. https://doi.org/10.1134/S2070048218050113.
[58] A. Mutzke, G. Bandelow, R. Schneider, Sputtering of mixed materials of beryllium and tungsten by hydrogen and helium, Journal of Nuclear Materials 467 (2015) 413–417. https://doi.org/10.1016/j.jnucmat.2015.05.052.
[59] A.S. Shumilov, I.I. Amirov, Morphology simulation of the surface subjected to low-energy ion sputtering, Tech. Phys. 60 (2015) 1056–1062. https://doi.org/10.1134/S1063784215070245.
[60] G.Z. Li, R.E. Wirz, Persistent Sputtering Yield Reduction in Plasma-Infused Foams, Phys. Rev. Lett. 126 (2021) 035001. https://doi.org/10.1103/PhysRevLett.126.035001.
[61] C.E. Huerta, T.S. Matlock, R.E. Wirz, View factor modeling of sputter-deposition on micron-scale-architectured surfaces exposed to plasma, Journal of Applied Physics 119 (2016) 113303. https://doi.org/10.1063/1.4944035.
[62] D. Angelis, F. Sofos, T.E. Karakasidis, Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives, Arch Computat Methods Eng 30 (2023) 3845–3865. https://doi.org/10.1007/s11831-023-09922-z.
[63] Y. Cui, H.B. Chew, Machine-Learning Prediction of Atomistic Stress along Grain Boundaries, Acta Materialia 222 (2022) 117387. https://doi.org/10.1016/j.actamat.2021.117387.
[64] W. Noh, H.B. Chew, Dislocation descriptors of low and high angle grain boundaries with convolutional neural networks, Extreme Mechanics Letters 68 (2024) 102138. https://doi.org/10.1016/j.eml.2024.102138.

CHAPTER 3: SURFACE MORPHOLOGY AND CARBON STRUCTURE EFFECTS ON SPUTTERING: MOLECULAR DYNAMICS SIMULATIONS [1]

[1] *Extracts of this chapter can be found in* Tran and Chew, Carbon, 205, 180-193

In this section, we perform molecular dynamics (MD) simulations on the xenon ion bombardment of carbon substrates across an extensive range of ion energies and ion incidence angles, to resolve the uncertainties in the experimental sputter measurements. We describe the MD modeling in Section 3.1, and elucidate the atomistic mechanisms of sputter erosion, in relation to the initial carbon structure and sputtering conditions in Section 3.2. In Section 3.3, we upscale our sputter yield predictions from MD to account for surface morphology effects, and adopt a Bayesian approach to calibrate the parameters of a semi-empirical sputter yield formula based on our results. Section 3.4 discusses the implications of our simulation results in the context of sputtering experiments in the literature. We conclude in Section 3.5 with a summary.

3.1 Molecular Dynamics Modeling

We perform MD simulations on the xenon ion bombardment of carbon substrates using LAMMPS [1]. The C-C interatomic interactions are governed by an Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) potential [2]. This potential allows for bond breaking and reforming, and accurately describes the sp-sp^2-sp^3 bond hybridization in carbon structures [3], along with the van der Waals long-range interactions typical for multilayer graphene (MLG) structures [4]. Separately, the Xe-Xe and Xe-C interactions are governed by a Ziegler-Biersack-Littmark (ZBL) potential [5], to account for screened nuclear repulsion associated with the high-energy collision between Xe-Xe or Xe-C atoms. As shown by our density functional theory (DFT) [6–8] calculations in Fig. 3.1, this ZBL potential accurately captures the interactions between

xenon atom and various configurations of carbon atoms since the attractive potential of ~0.5 eV between C-Xe for configurations 1, 3, and 5 is relatively small compared to the xenon ion energy.

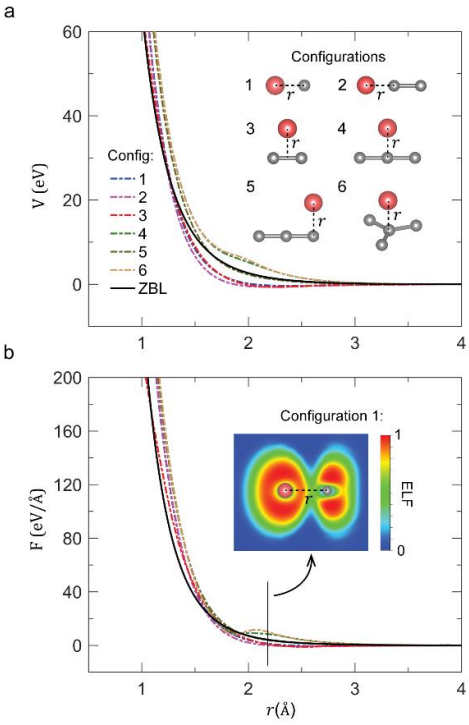

Figure 3.1: (**a**) Comparison of the interatomic interaction energy V as a function of interatomic distance r predicted by the ZBL potential (solid black line) versus DFT computations using VASP (dash lines). The (atomic mass, atomic number) for xenon and carbon atoms are taken to be $(131.293\ \text{g mol}^{-1}, 54\ \text{amu})$ and $(12.011\ \text{g mol}^{-1}, 6\ \text{amu})$ for the ZBL potential. Each dashed line in the DFT computations corresponds to the potential energy function between xenon atom (red atom) and the various s, sp, and sp^2 configurations of carbon (grey atoms) along the distance r as shown in the inset. (**b**) The interatomic force $F = -\frac{\partial V}{\partial r}$ (eV/Å) as a function of distance r(Å) based on the ZBL potential (solid black line) and DFT computations of the respective configurations in (**a**) (dashed lines). Contours of the electron localization function (ELF) for configuration 1 depicted in inset suggests negligible bonding between xenon (red) and carbon (grey) atoms.

Majority of our ion bombardment simulations are performed on horizontally-oriented MLG (h-MLG), with the ABA stacking arrangement oriented normal to the vertical (z) axis (Fig. 3.2a). Our simulation box is periodic in the in-plane (x-y) directions with dimensions of 5.1×4.9 nm^2. We fix the bottom graphene layer throughout our simulations and designate the next two layers above as the damping layer to absorb the stress waves caused by ion impact. This follows with four heat bath layers above, which we introduce to equilibrate the system to the targeted temperature. Depending on the xenon ion incidence energy (E) and incidence angle (θ), we include multiple graphene layers above (9 layers in case of $E = 500$ eV in Fig. 3.2a) to ensure that the damage caused by the incident xenon ions is confined within the active graphene layers above the heat bath. In addition to h-MLG, we also perform ion bombardment simulations on diamond and

vertically-oriented MLG (v-MLG). For the former, we assign one, two and four unit cells of diamond (lattice parameter of 3.567 Å) as the fixed, damping, and heat bath layers, with an active diamond substrate comprising of 14×15×8 unit cells above. Our ion bombardment simulations of v-MLG comprise of 15 graphene layers each of (*y*-*z*) dimensions 4.9×5.4 nm^2. Atoms from within the bottom 7 Å and the next 14 Å of these graphene layers are designated as the damping and heat bath zones, respectively. In addition, we attach a <001> face-centered-cubic (FCC) Cu lattice structure to the bottom of these graphene layers, to provide structural rigidity and to capture the penetrated Xe ions channeling between these vertical graphene layers. The interatomic interactions between Cu-Cu, Cu-C, and Xe-Cu atoms are governed by an Embedded Atom Method (EAM) potential [9], a Lennard-Jones (LJ) potential, and a ZBL potential, respectively, though the exact choice of these potentials will not affect the sputtering simulations since this Cu lattice is sufficiently far from the v-MLG surface.

Prior to initiating the bombardment sequence, we adopt a time step of 1 fs and subject the entire carbon structure sans the fixed layer to an NVT ensemble maintained at a temperature of 400 K by a Berendsen thermostat for 30 ps, which is a typical surface temperature from a beam dump experiment [10]. We then switch off the thermostat and perform the ion bombardment simulations using a time-accelerated scheme as follows. For each ion bombardment, we deposit one xenon atom randomly above the substrate surface with an initial ion velocity (v_x, v_y, v_z) constrained to have the designated ion (kinetic) energy, E, of between 75 and 2000 eV, and ion incidence angle, $\theta = \text{atan}\left(\sqrt{v_x^2 + v_y^2}/v_z\right)$, of between 0° and 75°. The exact choice of (v_x, v_y, v_z) is randomly selected at the start of the simulation but is fixed between the successive bombardments of a given set of (E, θ) to simulate a constant ion beam source. We allow the system to equilibrate without a thermostat for the first 1 ps, with a time step of 0.1 fs, to resolve the initial

impact dynamics associated with the deposition process. Using a time step of 1 fs, we then switch on the thermostat in the heat-bath region and set it to the target temperature of 400 K for the next 20 ps, before quenching the entire system to 400 K for a further 20 ps. We then repeat this entire process for the next ion bombardment. By accelerating the time between each bombardment sequence, our simulation predictions (e.g. sputter yield) are now in terms of the ion fluence, defined as the number of bombarded xenon ions per unit in-plane (*x*-*y*) area.

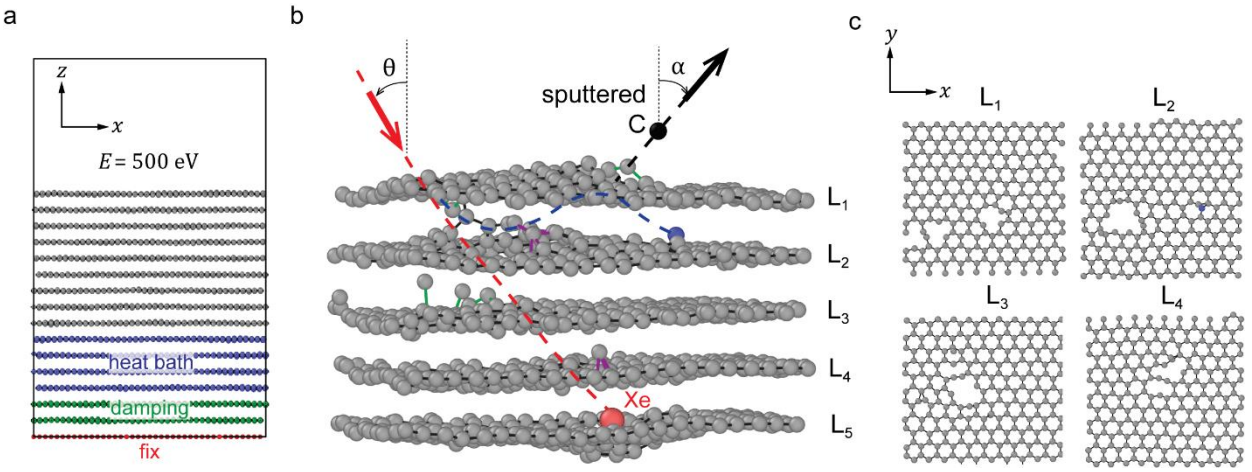

Figure 3.2: (a) Horizontally-oriented multilayer graphene (h-MLG) with designated fixed, damping, and heat bath layers colored in red, green, and blue, respectively. **(b)** Side view of the atomistic configuration of h-MLG after the first xenon ion impact at an energy of E=500 eV and incidence angle of θ=30°. Green, black, and purple short lines denote sp, sp2, and sp3 bonds. Dashed red, blue, and black lines denote trajectory of the xenon ion, a knock-off carbon atom, and a sputtered carbon surface atom with sputtered angle, α. **(c)** Top-view of the atomistic configurations of the first four graphene layers corresponding to **(b)**.

3.2 Atomistic Sputtering Mechanisms

Our MD simulations show that the penetration of xenon ions within the graphite layers at high velocities causes significant disruption to the atomic arrangement of carbon atoms along its path. Figure 3.2b shows the side view of the atomic configuration of the graphitic substrate after the first xenon ion impact with an ion energy of $E = 500$ eV and ion incidence angle of $\theta = 30°$. The heavier and significantly larger xenon atom knocks-off multiple carbon atoms, leading to a

collision cascade along its path (red dashed line), before residing between the L_4 and L_5 graphene interlayers. The subsequent collision of one of these knock-off carbon atoms (colored in blue with trajectory in blue dashed line) leads to sputtering of a surface carbon atom (black) with sputtered angle α and a kinetic energy of ~20 eV. The bombardment process results in the breaking of sp^2 bonds (black) to form sp bonds (green), as well as a small percentage of sp^3 bonds (purple). In addition, multiple pores are created within the first four graphene interlayers (L_1 to L_4 in Fig. 3.2c) along both the xenon ion path and at the location of the sputtered carbon atom (L_1).

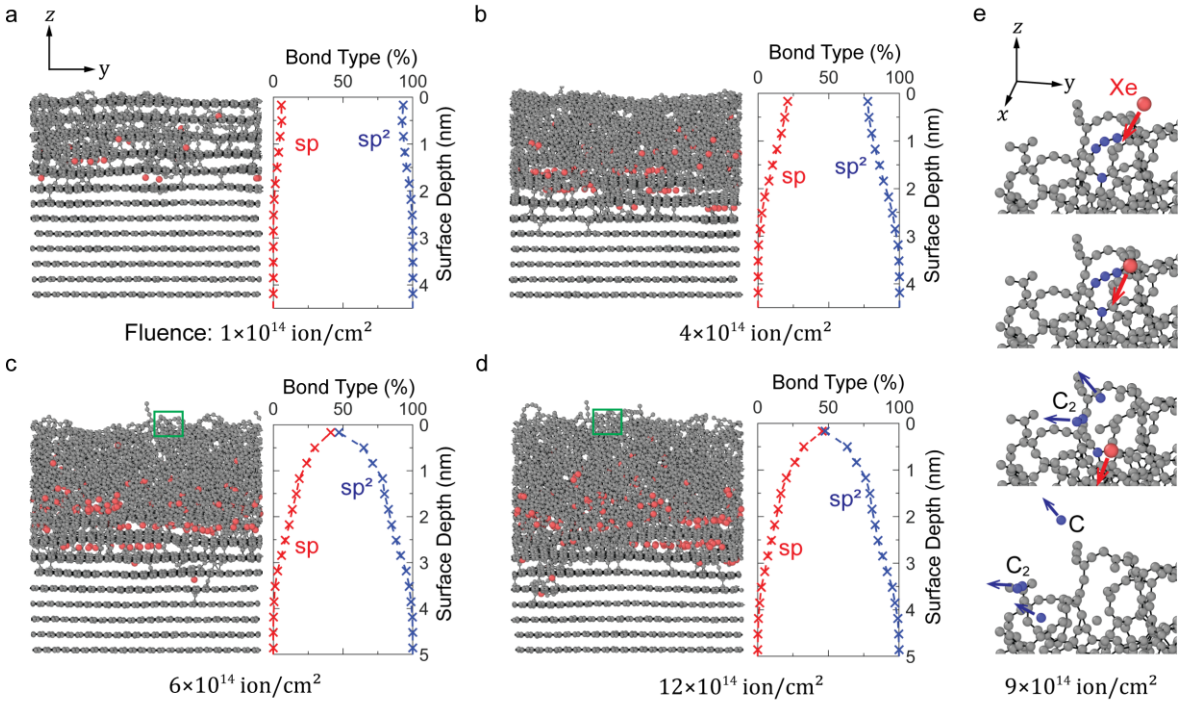

Figure 3.3: (**a-d**) Atomistic configuration (left) and corresponding spatial distribution of sp and sp^2 bond proportions (right) of h-MLG subjected to ion energy of $E = 500$ eV and ion incidence angle of $\theta = 30°$ at four ion fluences. (**e**) Close-up view [green box in (**c,d**)] depicting the sputtering sequence initiating with scission of sp and sp^2 bonded C-C chains (blue atoms) along the xenon ion path (red atom), leading to the ejection of 2 single and 1 duo carbon atoms from the surface. Arrows denote trajectory of xenon (red) and sputtered carbon atoms (blue).

Repeating the bombardment process above increases the density of sp bonds at the expense of sp^2 bonds, and results in amorphization of the graphitic subsurface spanning the penetration depth of the xenon ion. We partition the evolving atomistic structure into vertical bins of 0.335 nm

representing the interlayer spacing, and average the proportion of sp and sp^2 bonds within each bin. Figure 3.3a-d shows the atomistic structure and the spatial distribution of sp versus sp^2 bond proportions corresponding to four fluences. At low ion fluence of 1×10^{14} ion/cm^2, chains of sp-bonded C-C atoms form across the damaged interlayers, but the layered h-MLG structure is still distinct (Fig. 3.3a). Increasing the fluence to 4×10^{14} ion/cm^2 leads to a more porous and homogeneous carbon structure, with increasing (decreasing) proportion of sp (sp^2) bonds towards the exposed surface (Fig. 3.3b). These trends continue until an almost equal proportion of sp and sp^2 bonds are formed at the surface (Fig. 3.3c). Beyond this fluence, steady-state sputtering is achieved, and the spatial distribution of sp versus sp^2 bond proportions, indicative of the structural characteristics of the amorphous subsurface, remains unchanged under further ion bombardment (compare Fig. 3.3c and d). Figure 3.3e shows snapshots of the typical steady-state sputtering sequence, caused by scission of both sp and sp^2 bonded C-C chains (blue atoms) along the xenon

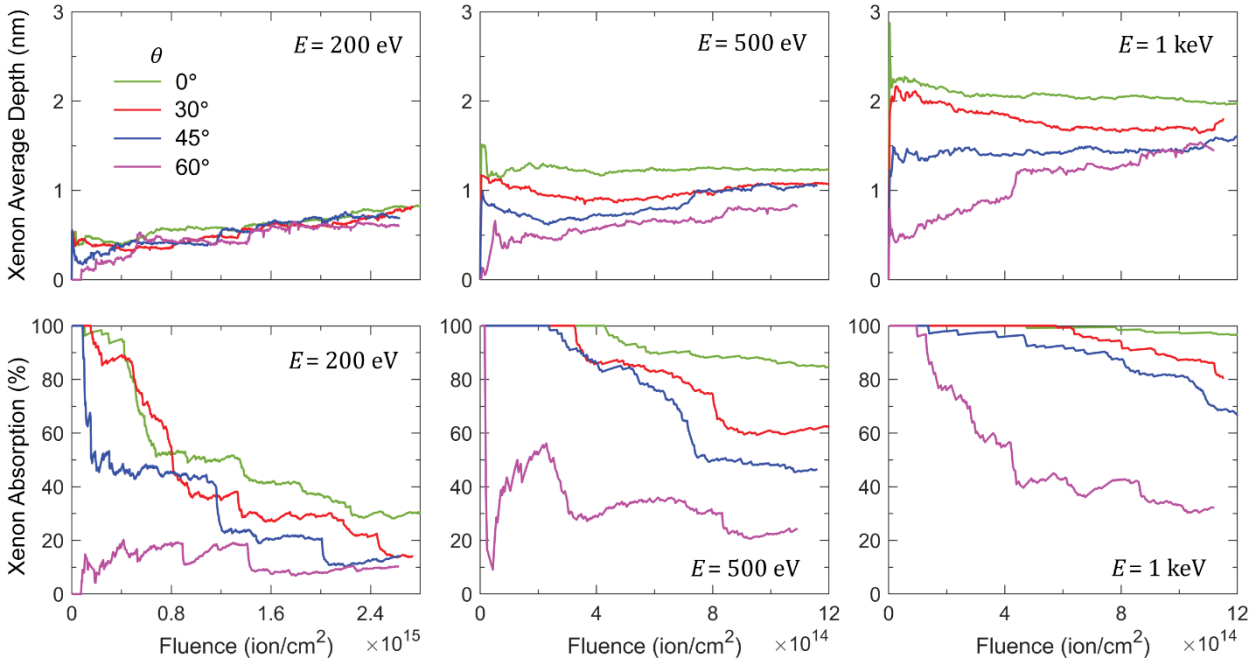

Figure 3.4: Average penetration depth (nm) of xenon ions (above), and the absorption rate (%) of xenon ions depicting the ratio of implanted versus total bombarded xenon ions (below), as a function of ion fluence for $E = 200$, 500, and 1000 eV, and at $\theta = 0°$, 30°, 45° and 60°.

ion path (red atom, arrow), leading to the ejection of 2 single and 1 duo carbon atoms from the surface. Similarly, we have traced the origin of the sputtered carbon species under steady-state bombardment across different E and have found that the sputtered species consistently originate from the exposed surface due to the higher proportion of weaker sp bonds. Unlike metallic materials, the carbon substrate traps xenon ions within the subsurface to a much higher degree, with the xenon absorption rate ranging from ~20% at $E = 500$ eV, $\theta = 60°$ to ~100% at $E = 1$ keV, $\theta = 0°$ (Fig. 3.4). This high xenon absorption rate will significantly bias mass loss measurements for quantifying the sputtering yield in experiments [11–13].

We average the percentage of sp and sp^2 bonds, as well as the atomic density, within the volume of the amorphized region bounded by the average xenon ion penetration depth (Fig. 3.4) and α-shapes surface reconstruction [14] at each fluence, to quantify the evolving bond fraction and porosity in the amorphous subsurface. Figures 3.5 shows these evolving structural characteristics with ion fluence for three representative ion energies of $E = 200$, 500 and 1000 eV, and at four ion incidence angles of $\theta = 0°$, 30°, 45° and 60°. In all cases, the decrease in the sp^2 bond percentage with ion fluence is accompanied by a corresponding increase in the sp bond percentage (Fig. 3.5a) – the percentage of both these bond types under steady-state bombardment (arrows) sum to 96%, with the remaining 4% attributed to sp^3 bonds. In addition to these changes in the bond types, the structure also becomes increasingly porous with bombardment, as shown by the decrease in atomic density ρ (Fig. 3.5b). Beyond a certain fluence (arrows in Figs. 3.5), both the sp/sp^2 bond fractions and ρ reach a plateau, with nearly identical steady-state values for different E and θ. This suggests that the amorphous subsurface structures created by different ion energies and ion incidence angles are virtually indistinguishable from one another under steady-

state bombardment. We have repeated these calculations across a wider range of E and θ, and have summarized the near-uniform steady-state sp/sp^2 bond fractions and ρ in Fig. 3.6.

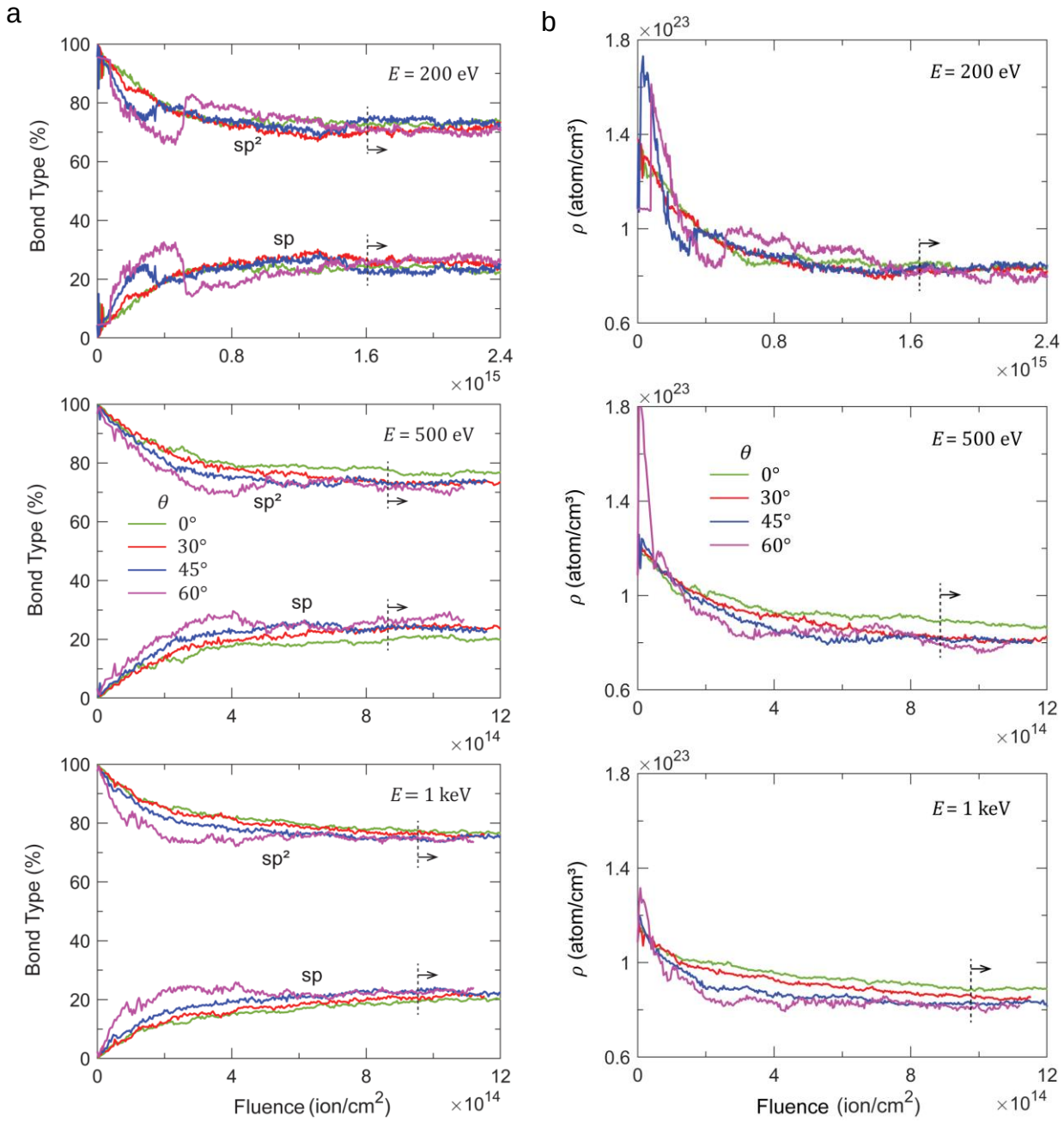

Figure 3.5: Evolution of **(a)** the average percentage of sp and sp^2 bonds and **(b)** the atomic density, ρ, within the amorphized subsurface with ion fluence for ion energies of $E = 200$, 500, and 1000 eV, and at ion incidence angles of $\theta = 0°$, 30°, 45° and 60°. Arrows denote steady-state sputtering regime.

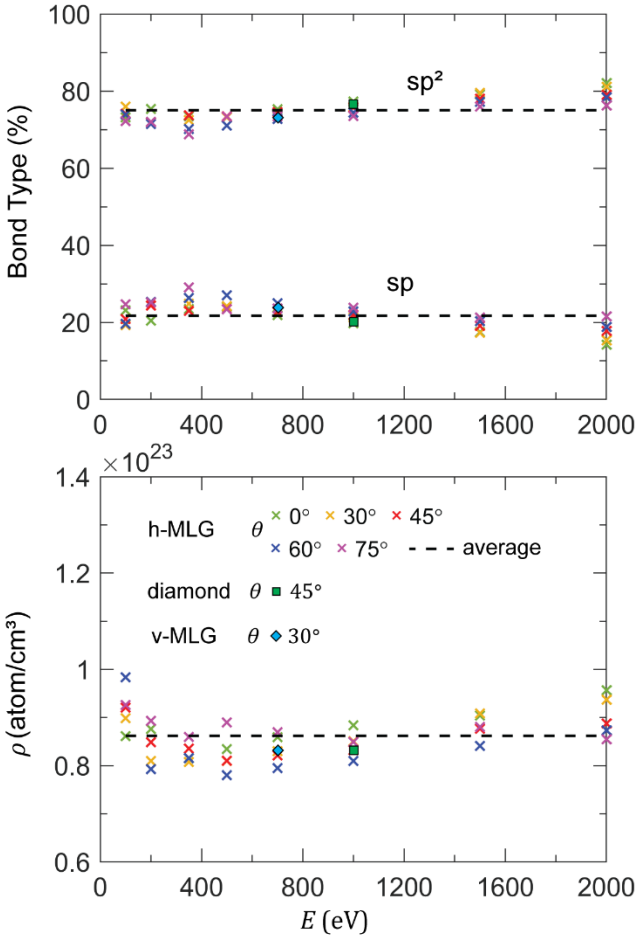

Figure 3.6: Average percentage of sp and sp^2 bond proportions (**a**) and atomic density, ρ, (**b**) within the amorphous subsurface across different E and θ for different initial crystal structures: h-MLG (cross symbol), v-MLG (diamond symbol), diamond (square symbol). Black dashed line denotes average structural characteristics for h-MLG. Observed scatter in (**a**) and (**b**) about the black dashed line can be attributed to stochasticity in the ion bombardment simulations and uncertainties in estimation of the amorphized layer thickness.

Our amorphous subsurface structures above are created by the xenon ion bombardment of h-MLG. We repeat our ion bombardment simulations on sp^3-bonded crystalline diamond with $E =$ 1 keV, $\theta = 45°$ (Fig. 3.7a), as well as v-MLG with $E = 700$ eV, $\theta = 30°$ (Fig. 3.7b). The initial evolution of the sp/sp^2 bond proportion (Fig. 3.7c) and the atomic density ρ (Fig. 3.7d) is sensitive to the initial crystal structure. For v-MLG (blue curves), the increase in sp bond proportion with fluence is accompanied by a decrease in sp^2 bond proportion, which is similar to h-MLG. In contrast, for the diamond substrate (red curves), both proportions of sp and sp^2 bonds are converted from sp^3 bonds during the bombardment process, and monotonically increase with fluence. In addition, because of the more-density packed nature of sp^3-bonded diamond, the drop in ρ

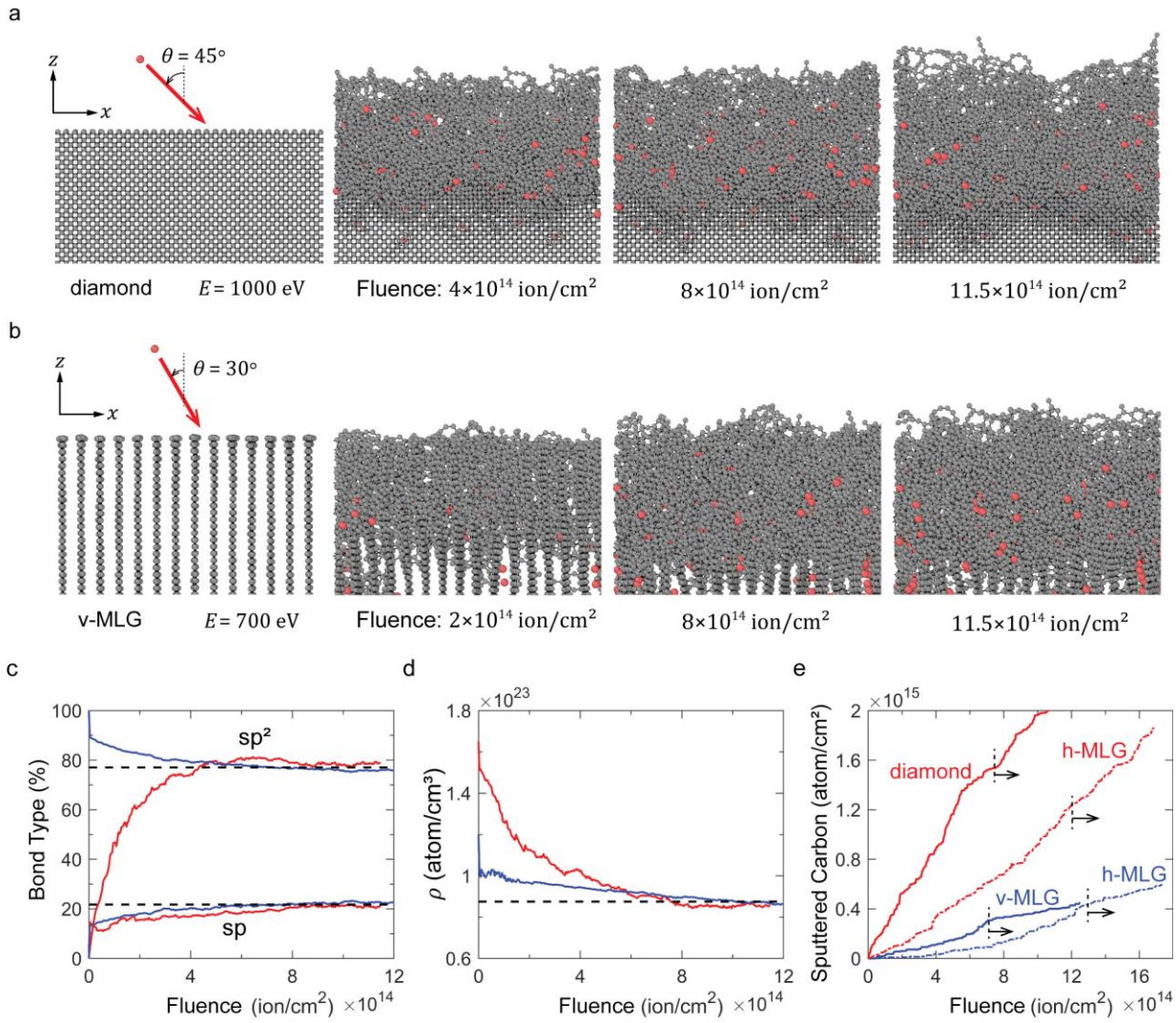

Figure 3.7: (**a,b**) Evolution of the atomistic configurations with ion fluence for diamond with $E = 1$ keV, $\theta = 45°$ (**a**), and for vertically-oriented multilayer graphene (v-MLG) with $E = 700$ eV, $\theta = 30°$ (**b**). (**c,d**) Evolution of the proportion of sp and sp^2 bonds (**c**) and atomic density ρ (**d**) within the amorphous subsurface for diamond (red) and v-MLG (blue). Dashed black lines denote average values for h-MLG under steady-state bombardment. (**e**) Total quantity of sputtered carbon atoms with fluence for diamond (red) and v-MLG (blue), as compared to h-MLG under the same E, θ (corresponding dashed red, and blue lines). Arrows denote the steady-state sputtering regime.

(increase in porosity) for the diamond substrate with fluence is also more dramatic compared to v-MLG. Nevertheless, under steady-state bombardment, both the proportion of sp/sp^2 bonds and the atomic densities plateau to the exact same values as for h-MLG (average steady-state values across

all h-MLG cases simulated denoted by dashed black lines). Thus, the steady-state amorphous carbon subsurface structures created by ion bombardment appear to be independent of *both* the sputtering conditions (E, θ), as well as the initial carbon structure. We track the accumulated quantity of sputtered carbon as a function of ion fluence for both diamond (red solid line) and v-MLG (blue solid line) in Fig. 3.7e and include the results for h-MLG under the same sputtering conditions as for diamond (red dashed line) and v-MLG (blue dashed line). While the diamond structure has the highest initial sputtering yield, as defined by the slope of the curves in Fig. 3.7e, the sputtering yields under steady-state conditions (arrows) are nearly identical for diamond and h-MLG, as well as v-MLG and h-MLG, under the same E and θ. This confirms that the bombardment process creates a unique amorphous subsurface structure that is independent of both the initial carbon structure (e.g., diamond, v-MLG, and h-MLG), and the sputtering history (prior E and θ). Consequently, the measured steady-state sputtering yield from all these different initial carbon structures will be identical.

We summarize the steady-state sputtering yield from MD simulations across different E and θ in Fig. 3.8 (blue symbols). Note that these MD results are based on the xenon ion bombardment of an initial h-MLG structure, with exception for $E = 75$ eV (all θ) where we adopt the steady-state amorphous structure created from h-MLG subjected to $E = 200$ eV ($\theta = 45°$) as the starting initial configuration for our ion bombardment simulations to reduce computational time. In fact, starting with h-MLG at $E = 75$ eV may not result in carbon sputtering because of the strong in-plane sp^2 bonds. We include the sputtering yield from existing experimental studies in the literature (black symbols), as well as our calculations from SRIM models (purple symbols [15,16]). As prior studies have shown, SRIM calculations based on binary collision approximation tend to be inaccurate at ion energies of below ~1 keV, due to the inability to handle many-body collisions of

arbitrarily many atoms [17,18]. Similarly, our sputtering yield predictions from SRIM tend to deviate from MD predictions at $E < \sim 1$ keV. More importantly, our MD results on the sputtering yield are generally in good agreement with experimental measurements [11,12,19–24] at intermediate θ across the ion energy range of 75 eV $\leq E \leq$ 2 keV, but under and overpredicts the sputtering yield at $\theta = 0°$ and 60°, respectively. We attribute this to surface morphology effects, detailed later in Section 3, which are beyond the length-scales of MD simulations.

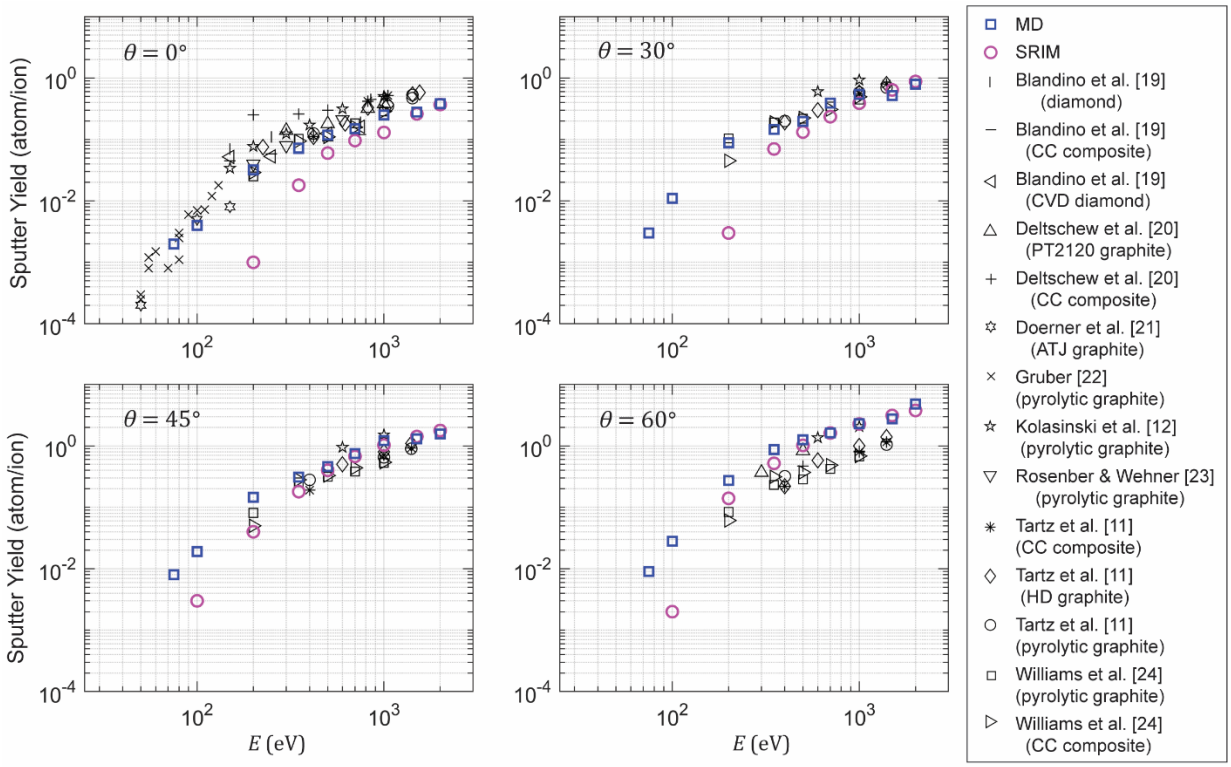

Figure 3.8: Steady-state sputtering yield from MD simulations (blue symbols) and SRIM calculations (purple symbols) versus experiments for different carbon structures (black symbols) across various E and θ.

In addition to the sputtering yield, the differential sputter yield profile, representing the probability density function (PDF) $f(\alpha)$ of the angle of the carbon sputterants, α, is equally essential to quantifying the contributions of redeposition (contamination) effects on plasma facing materials, including those of solar cells, fusion reactors, and electric propulsion facilities [25–30].

From MD simulations, we trace the local angle of emission α of the carbon sputterants (Fig. 3.2b) ranging from $-90°$ to $90°$ with respect to the surface normal under steady-state bombardment, to construct $f(\alpha)$ for sputterant emission in Fig. 3.9, at ion energies of $E = 200$, 500, 1000 and 2000 eV and at four ion incidence angles of $\theta = 0°$, $30°$, $45°$ and $60°$. Each symbol in Fig. 3.9 represents a cumulative probability density based on the sum of all sputterants across the azimuthal angles over a bin size of $\pm 10°$ from the indicated α value. Our results show that the profile of $f(\alpha)$, while primarily a function of θ, is independent of E. The PDF profile transitions from near-uniform distribution of sputterants across a wide range of α for $\theta = 0°$, to having a skewed distribution at higher θ favoring the forward scatter direction with maximum distribution of sputterants centered at $\alpha = \sim 50°$. In addition, $f(\alpha)$ at a specific θ can be well described by a linear combination of

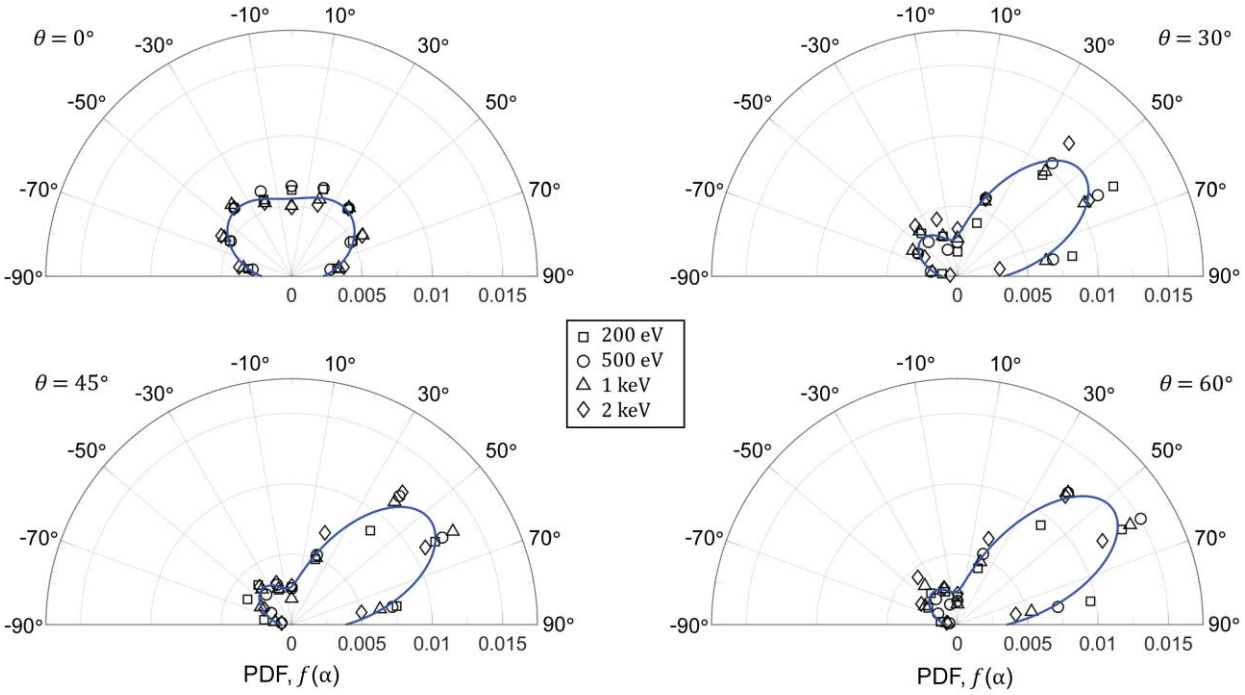

Figure 3.9: Probability density function (PDF) of the differential yield profile of the sputterants, $f(\alpha)$, from MD simulations, at $E = 200$, 500, 1000 and 2000 eV with $\theta = 0°$, $30°$, $45°$ and $60°$. Each symbol denotes the cumulative probability density over all azimuthal angles within a bin size of $\Delta\alpha = \pm 10°$. Blue curve shows the average fitting over all E for each θ, based on a linear combination of two Gaussian distribution functions.

two Gaussian function distributions (blue curve in Fig. 3.9), each representing the dominant forward ($0° < \alpha < 90°$) and backward ($-90° < \alpha < 0°$) scatter directions

$$f(\alpha) = \frac{A}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\alpha-\mu_1}{\sigma_1}\right)^2} + \frac{1-A}{\sigma_2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\alpha-\mu_2}{\sigma_2}\right)^2} \tag{3.1}$$

where μ_j represents the peak sputtered angle, σ_j the spread due to the inelastic collisions, and A the relative contribution of the forward $(j = 1)$ and backward $(j = 2)$ sputterants. The fitting parameters for μ_j, σ_j, A are summarized in Table 3.1.

Table 3.1: Fitting parameters for the linear combination of two Gaussian distribution functions (equation (3.1)) used to describe the PDF of the sputtered carbon direction (α) as a function of θ and $\frac{H}{\lambda}$; the PDF is assumed to be independent of E.

$\frac{H}{\lambda} = 0$ (MD)						$\frac{H}{\lambda} = 0.1$					
θ	μ_1	σ_1	μ_2	σ_2	A	θ	μ_1	σ_1	μ_2	σ_2	A
0°	39.59	36.16	-44.11	32.48	0.54	0°	43.90	31.15	-42.23	32.49	0.49
15°	49.78	29.20	-37.34	35.78	0.56	15°	55.49	23.86	-29.38	37.06	0.52
30°	49.96	25.21	-43.38	29.66	0.73	30°	57.06	22.94	-25.41	36.98	0.62
45°	52.86	24.34	-37.30	30.62	0.75	45°	57.77	22.97	-23.38	37.62	0.70
60°	53.33	22.18	-35.82	33.31	0.76	60°	56.45	21.83	-24.46	39.19	0.72
75°	54.61	20.02	-35.18	35.70	0.81	75°	44.95	20.69	-40.18	38.62	0.75
$\frac{H}{\lambda} = 0.2$						$\frac{H}{\lambda} = 0.3$					
θ	μ_1	σ_1	μ_2	σ_2	A	θ	μ_1	σ_1	μ_2	σ_2	A
0°	50.80	30.82	-42.05	38.90	0.43	0°	43.69	38.01	-39.32	40.44	0.47
15°	63.34	19.44	-13.54	45.06	0.39	15°	67.39	16.40	-0.64	47.09	0.24
30°	64.54	19.25	-8.73	40.30	0.50	30°	68.85	17.42	0.21	40.28	0.38
45°	63.47	19.05	-4.06	43.10	0.54	45°	58.69	26.39	-16.62	36.09	0.56
60°	48.48	22.06	-18.64	43.51	0.61	60°	38.05	34.11	-41.89	32.21	0.75
75°	34.30	24.11	-48.25	38.92	0.73	75°	30.24	26.86	-46.55	39.65	0.71

We also summarize the carbon cluster contribution in Table 3.2 below. Single Carbon is the most dominant form of carbon sputtering, following by C_2 clusters while C_3 cluster contributes to less than 3% of the total sputter yield. This results appears contradicts with experiment cluster analysis by Oyarzabal et. al. [31] for Xenon at 225 eV, where C_2 and C_3 clusters are the main contributions. However, more experiments are required to verify this information.

Table 3.2: Ratio of Carbon cluster contributions to the total sputtering yield for different xenon ion incidence energy E and angle θ

	$E = 100$ eV			200 eV			350 eV			500 eV		
θ	C	C_2	C_3	C	C_2	C_3	C	C_2	C_3	C	C_2	C_3
0	0.79	0.18	0.03	0.87	0.13	0.00	0.61	0.17	0.16	0.57	0.36	0.07
15	0.69	0.31	0.00	0.80	0.07	0.13	0.60	0.28	0.12	0.59	0.23	0.18
30	0.74	0.26	0.00	0.50	0.35	0.06	0.72	0.20	0.08	0.59	0.34	0.07
45	0.56	0.29	0.16	0.47	0.23	0.16	0.65	0.26	0.10	0.63	0.20	0.17
60	0.40	0.36	0.24	0.41	0.25	0.15	0.49	0.28	0.15	0.59	0.23	0.18
75	1.00	0.00	0.00	0.49	0.31	0.15	0.56	0.29	0.09	0.57	0.31	0.09
	$E = 700$ eV			1000 eV			1500 eV			2000 eV		
θ	C	C_2	C_3	C	C_2	C_3	C	C_2	C_3	C	C_2	C_3
0	0.69	0.11	0.20	0.57	0.27	0.16	0.76	0.21	0.03	0.85	0.15	0.00
15	0.65	0.21	0.14	0.69	0.29	0.03	0.73	0.12	0.15	0.83	0.17	0.00
30	0.66	0.18	0.10	0.60	0.26	0.14	0.85	0.15	0.00	0.82	0.18	0.00
45	0.55	0.27	0.18	0.61	0.25	0.14	0.60	0.28	0.12	0.60	0.26	0.14
60	0.58	0.26	0.16	0.57	0.26	0.11	0.54	0.26	0.20	0.56	0.30	0.14
75	0.52	0.32	0.13	0.54	0.30	0.16	0.57	0.30	0.07	0.69	0.21	0.10

3.3 Surface Morphology Modeling

Scanning electron microscopy (SEM) imaging of amorphous carbon films show the surface, while macroscopically smooth, appears very rough at lower (micrometer) length-scales (Fig. 3.10a) [12]. Atomic force microscopy (AFM) measurements of the surface height map (Fig. 3.10b) show the surfaces to have height variations (amplitude) on the order of $\frac{H}{2} = \sim 200$ nm. In addition, there are a total of 12 peaks within the 5×5 μm^2 surface height map, which infers a wavelength of $\lambda = \sim 2$ μm. For simplicity, we represent the surface morphology in one-dimension by a cos function

$$h = \frac{H}{2} \cos\left(\frac{2\pi x}{\lambda}\right) + \frac{H}{2} \tag{3.2}$$

As depicted schematically in Fig. 3.10c, the surface normal $\boldsymbol{n}$ now varies with location x along the surface. Hence, the local incident angle, θ_L, at the surface impact point is now distinct from the global ion incidence angle θ. Note that the sputtering yield predictions from MD (Fig. 3.8) are with respect to the surface normal $\boldsymbol{n}$, and hence are based on local ion incidence angle θ_L. The continuous variation of θ_L along the surface in turn causes the sputtering yield $Y_{\mathrm{MD}}(E, \theta_L)$ to vary even for ion bombardment with uniform θ. Because of the undulating surface, however, only a proportion K of the sputterants can escape from the surface (purple arrows in Fig. 3.10c), while the remaining proportion D $(= 1 - K)$ is redeposited back onto the surface (green arrows in Fig. 3.10c). The values of K and D are the cumulative probability of $f(\alpha)$ (Fig. 3.9) over the domains that can be delineated by lines joining the ion impact site, and the surface peaks within the field-of-view (dashed purple line in Fig. 3.9c). In addition, the undulating surface shields a portion of the surface, between the limits of $x_s < x < x_e$ in Fig. 3.10d, from the incoming ions. Accounting for all three surface morphology effects, the effective sputtering yield can be expressed as

$$Y\left(\frac{H}{\lambda}, E, \theta\right) = \frac{\int_0^{\lambda} K Y_{\mathrm{MD}}[1 - \mathrm{stp}(x - x_s) + \mathrm{stp}(x - x_e)]\zeta dx}{\int_0^{\lambda} \zeta dx} \tag{3.3}$$

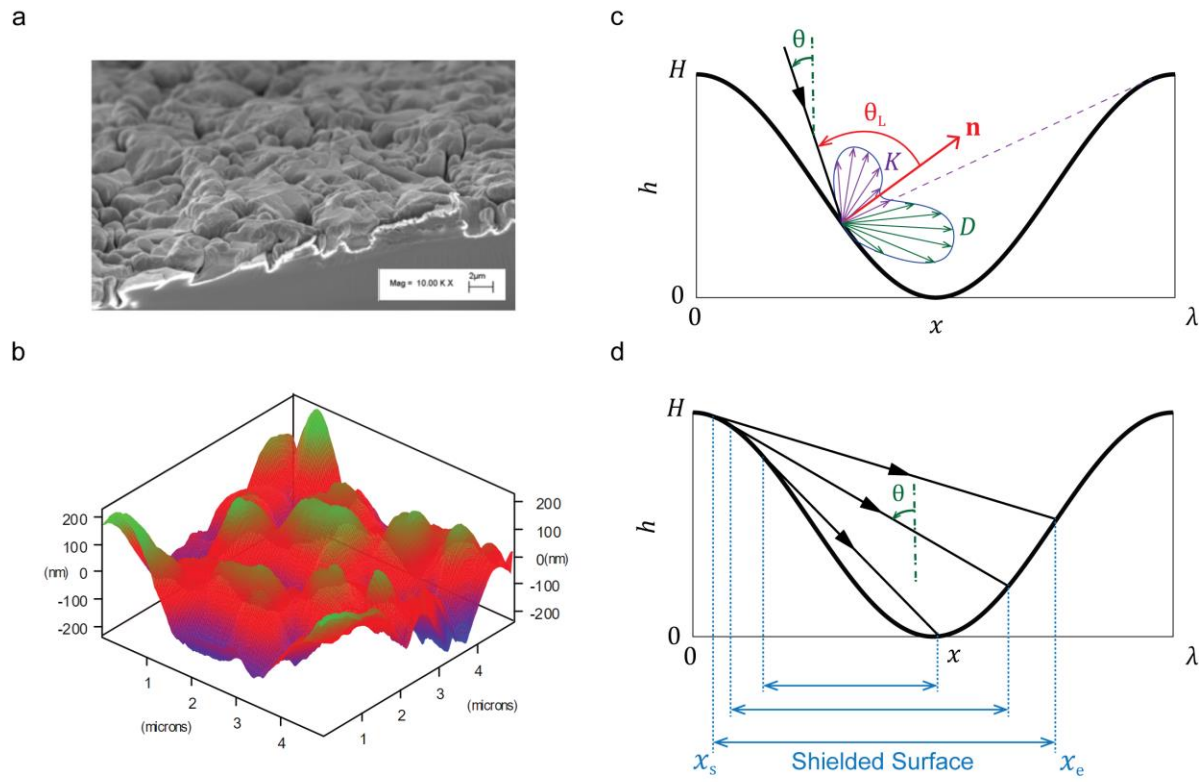

Figure 3.10: (**a,b**) Scanning electron microscopy (SEM) imaging (**a**) and atomic force microscopy (AFM) surface height measurements (**b**) of amorphous carbon films (Kolasinski et al., 2006). (**c,d**) Schematic of an idealized surface geometry represented by a cosine function with amplitude $H/2$ and wavelength λ, depicting changes in the local incidence angle θ_L from the global ion incidence angle θ at the impact site, redeposition of the sputterants D to only allow the proportion K to escape (**c**), along with the shielding of a portion of the surface, $x_s < x < x_e$, from the incoming ions (**d**).

where $\mathrm{stp}(x)$ denotes a unit step function, and $\zeta(\theta, x)$ is the local ion fluence representing the variation of ions impacting the undulating surface per unit elemental dimension dx. We remark that this simple model does not account for surface curvature effects, which are only apparent at high ion energies ($> \sim 5$ keV) and at very small λ ($< \sim 100$ nm) where the impact-induced damage can span across multiple nanoripples [31–33]. For a perfectly smooth surface, i.e. $\frac{H}{\lambda} = 0$, we obtain $Y = Y_{\mathrm{MD}}$. For $\frac{H}{\lambda} > 0$, we assume a constant incoming ion flux, and solve (3) numerically by discretizing the domain $0 \leq x \leq \lambda$ into individual elements, each with a constant K and Y_{MD} determined from the differential yield function $f(\alpha)$ obtained from MD (Fig. 3.9). We consider

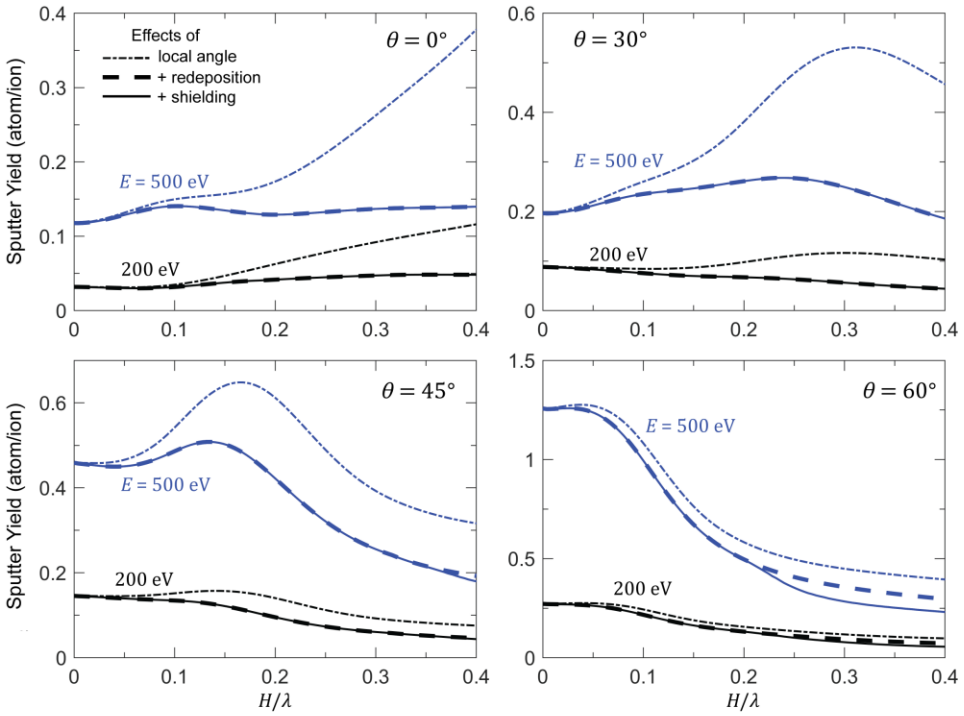

Figure 3.11: Changes in the effective steady-state sputter yield Y with $\frac{H}{\lambda}$ for $E = 200$ and 500 eV at $\theta = 0°$, 30°, 45° and 60°, considering the cumulative contributions of local incidence angle, surface redeposition, and surface shielding effects.

the cumulative contributions of local incidence angle, surface redeposition, and surface shielding effects on the effective steady-state sputtering yield Y as a function of $\frac{H}{\lambda}$ in Fig. 3.11. Results show that the changes in the local incidence angle (dashed dot line) significantly changes the sputtering yield across all θ, and represents the most significant effect of surface morphology. This inclusion of surface redeposition effect (thick dashed line), however, lowers the sputtering yield and partially mitigates the influence of local incidence angle changes. In contrast, surface shielding effect is only visible at high $\frac{H}{\lambda}$ coupled with high θ of 60°. When all three contributions are considered (solid line), our results show that the effects of $\frac{H}{\lambda}$ are especially dominant at higher ion incidence angles of $\theta = 60°$, where we observe a several-fold drop in the sputtering yield for $E = 500$ eV, at $\frac{H}{\lambda} = \sim 0.2$ representative of amorphous carbon films (Fig. 3.10a,b). We summarize our steady-state sputtering yield predictions, corrected for surface morphology effects ($\frac{H}{\lambda} = 0.2$) in

Fig. 3.12 (blue symbols). Our augmented MD predictions are now well within the experimental scatter across the range of ion energies and ion incidence angles.

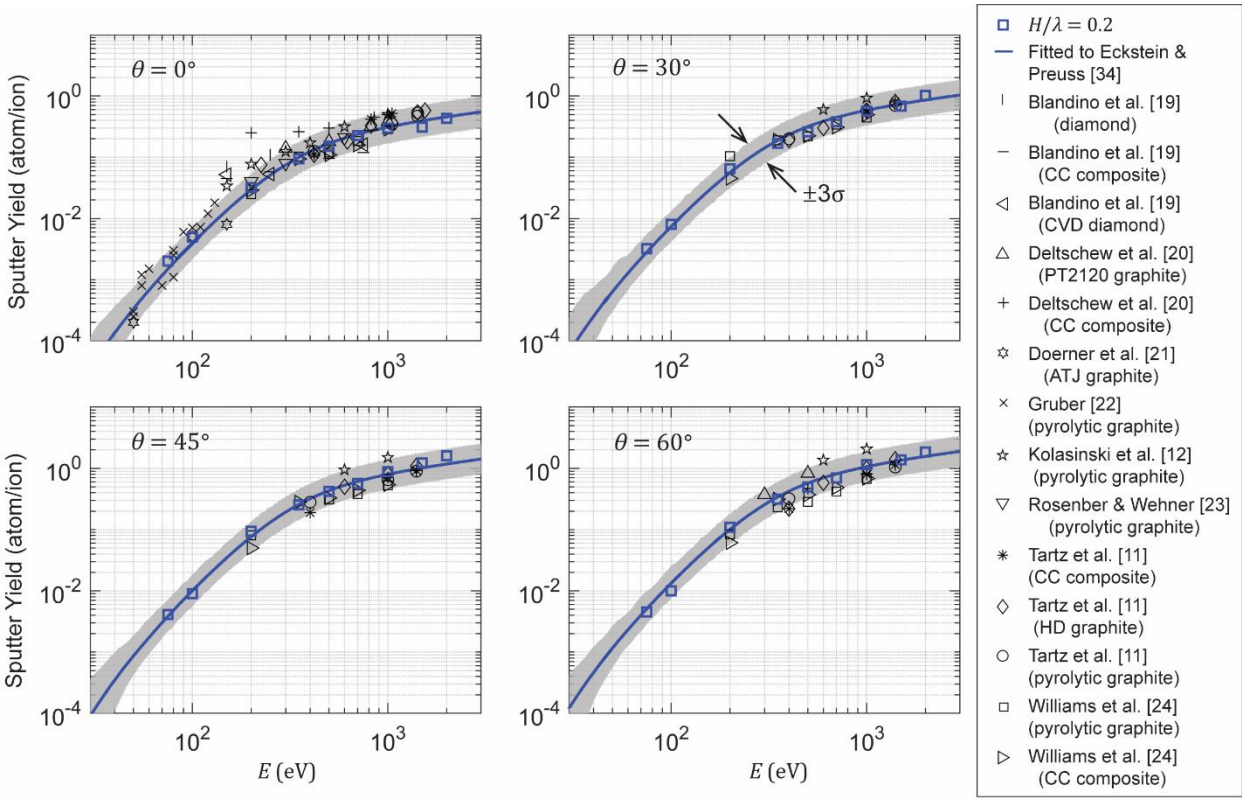

Figure 3.12: Steady-state sputtering yield from MD simulations augmented to account for surface morphology effects ($\frac{H}{\lambda} = 0.2$; blue symbols) versus experiments for different carbon structures (black symbols). Solid blue lines denote maximum likelihood predictions of the Eckstein & Preuss (2003) sputter yield model fitted to the MD results at $\frac{H}{\lambda} = 0.2$ (blue symbols), with the shaded grey region depicting the likelihood bounds of $\pm 3\sigma$, where σ is the standard deviation.

Empirical and semi-empirical models to describe the sputtering yield commonly assume that the energy and angular dependence are separable, i.e.

$$Y(E,\theta) = Y(E,0)Y'(\theta) \tag{3.4}$$

Here, we will adopt the formalism by Eckstein & Preuss [34], where the energy dependence of the sputtering yield at normal incidence can be expressed as

$$Y(E,0) = Qs_n \frac{\left(\frac{E}{E_{th}}-1\right)^{\mu}}{\frac{\lambda}{w}+\left(\frac{E}{E_{th}}-1\right)^{\mu}} \qquad (3.5)$$
$$w = \epsilon + 0.1728\sqrt{\epsilon} + 0.008\epsilon^{0.1504}$$

with the multiplicative factor for angular dependence

$$Y'(\theta) = (\cos(\theta^c))^{-f} \exp(b(1-(\cos(\theta^c))^{-1})) \qquad (3.6)$$

where s_n in (3.5) is the nuclear stopping cross section in the form of a Krypton-Carbon potential relevant to the low energy bombardment of xenon ions on carbon species [17,35], and is expressed

$$s_n = \frac{0.5\ln(1+1.2288\epsilon)}{\epsilon+0.1728\sqrt{\epsilon}+0.008\,\epsilon^{0.1504}} \qquad (3.7)$$

with the reduced nuclear stopping power

$$\epsilon = \frac{a_L}{Z_i Z_S}\frac{4\pi\varepsilon_0}{e^2}\frac{M_s}{M_i+M_s}E \qquad (3.8)$$

where $\varepsilon_0 = 1.42 \times 10^{-40}\ \mathrm{C^2 eV^{-1} Å^{-1}}$ is the vacuum permittivity, $e = 1.602 \times 10^{-19}\ \mathrm{C}$ is the elementary charge, (M_i, Z_i) and (M_S, Z_S) are the (atomic mass, atomic number) for xenon and carbon atoms and are taken to be $(131.293\ \mathrm{g\ mol^{-1}}, 54\ \mathrm{amu})$ and $(12.011\ \mathrm{g\ mol^{-1}}, 6\ \mathrm{amu})$, respectively, and a_L is the Lindhard screening length

$$a_L = \left(\frac{9\pi^2}{128}\right)^{\frac{1}{3}} a_0 \left(Z_i^{\frac{2}{3}} + Z_s^{\frac{2}{3}}\right)^{-\frac{1}{2}} \qquad (3.9)$$

with the Bohr radius, $a_o = 0.529$ Å.

The parameters $(Q, \lambda, \mu, E_{th})$ in (5) along with (b, c, f) in (3.6) can be regarded as physics-based fitting parameters for the energy and angular dependence portions of the semi-empirical sputtering model. Notably, Q is the scaling representing the spread of the impact energy between the ion and the target, λ, μ describes the strength and the onset of the sputtered yield at low ion energy, E_{th} denotes the threshold energy for sputtering, f scales with the proportion of the particle reflection coefficient, while b, c controls the peak sputter yield angle [34]. We employ Bayesian

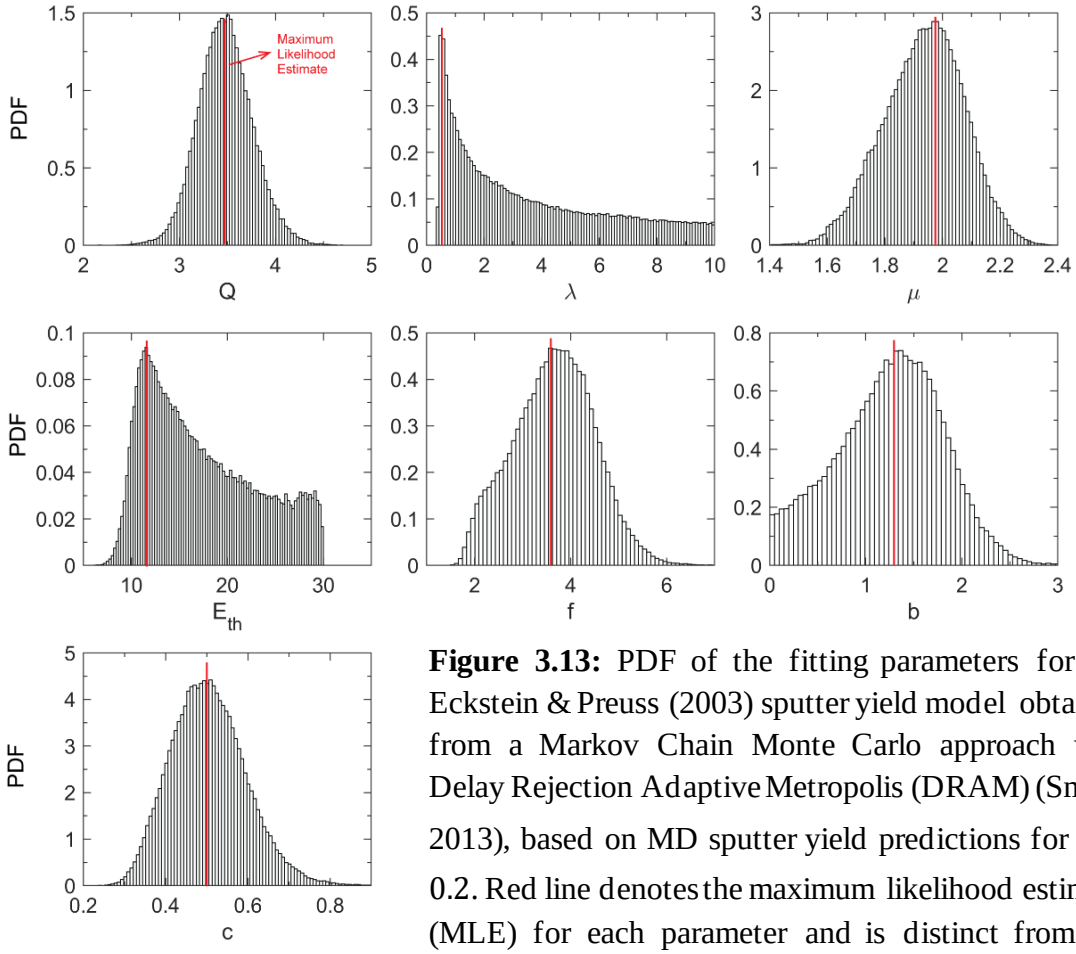

Figure 3.13: PDF of the fitting parameters for the Eckstein & Preuss (2003) sputter yield model obtained from a Markov Chain Monte Carlo approach with Delay Rejection Adaptive Metropolis (DRAM) (Smith, 2013), based on MD sputter yield predictions for $\frac{H}{\lambda} = 0.2$. Red line denotes the maximum likelihood estimate (MLE) for each parameter and is distinct from the parameters used to construct the MLE of the fitted sputter-yield response (Table 3.3).

parameter estimation, applied to the MD dataset of the sputtering yield augmented to account for surface morphology effects ($\frac{H}{\lambda}$), to generate posterior probability distributions for these semi-empirical parameters. Because the sputtering yield at our lowest simulated ion energy of $E = 75$ eV is several orders of magnitude smaller than the sputtering yield at $E = 2$ keV (Fig. 3.12), parameter estimation in the linear space would incorrectly negate the lower energy dataset in favor of the higher energy dataset because of their significantly higher sputtering rates. Instead, we transform (3.4)-(3.6) to the natural logarithmic space to allow the MD dataset across the entire ion energy range of $75 \text{ eV} \leq E \leq 2$ keV to be adequately considered in a Markov chain Monte Carlo algorithm (MCMC) with Delay Rejection Adaptive Method (DRAM) [36]. We summarize in

Table 3.3 the fitting parameters of the Eckstein & Preuss [34] sputter model that yield the maximum likelihood curve based on MD results for varying $\frac{H}{\lambda}$. For completeness, we also include parameters fitted to the experimental sputter yield data of the various polymorphs of carbon (black symbols in Fig. 3.12) at normal ion incidence angle from Yim [37]. We observe significant changes to these maximum likelihood parameters with $\frac{H}{\lambda}$, which demonstrates the sensitivity of the sputtering data to surface morphology. Based on the maximum likelihood parameters for $\frac{H}{\lambda} = 0.2$ (see PDF of the fitting parameters in 3.13), we include in Fig. 3.12 the sputtering yield from the newly-calibrated Eckstein & Preuss [34] model (blue curves), along with the likelihood bounds of $\pm 3\sigma$ (shaded grey), where σ is the standard deviation. The fitted results are in near perfect agreement with MD data across all E and θ, with the likelihood bounds encompassing most of the experimental data.

Table 3.3: Fitting parameters for the Eckstein & Preuss [34] sputter yield formula based on MD-derived data.

Parameters	Surface Morphology, H/λ				Experiment (Yim, 2017 [37])
	0 (MD)	**0.1**	**0.2**	**0.3**	
Q (eV^{-1})	2.64	3.04	3.48	4.00	4
λ	3.96	3.80	4.01	3.86	0.8
μ	2.08	2.01	1.93	1.71	1.8
E_{th} (eV)	11.2	11.9	12.5	13.8	21
f	6.42	6.03	3.66	1.98	-
c	2.17	2.21	1.25	0.57	-
b	0.77	0.71	0.50	0.72	-

In addition to changes in the sputter yield predictions, the effects of surface morphology can induce changes to the apparent differential yield profile with respect to the global (x, y, z)

coordinate system. Here, we quantify the local sputter yield profile over all discretized surface elements between 0 and λ, to obtain the effective PDF of the differential yield profile $f(\alpha)$ for different $\frac{H}{\lambda}$ in Fig. 3.14. We summarize the parameters of (1) for different $\frac{H}{\lambda}$ in Table. Our results generally show that surface morphology has minimal effects on $f(\alpha)$, except at very high $\frac{H}{\lambda}$ of 0.3 with ion incidence angles of $\theta \geq 30°$, where a higher proportion of sputterants with $|\alpha| < 20°$ is favored since redeposition effects at higher $|\alpha|$ are exacerbated due to the steeper surface.

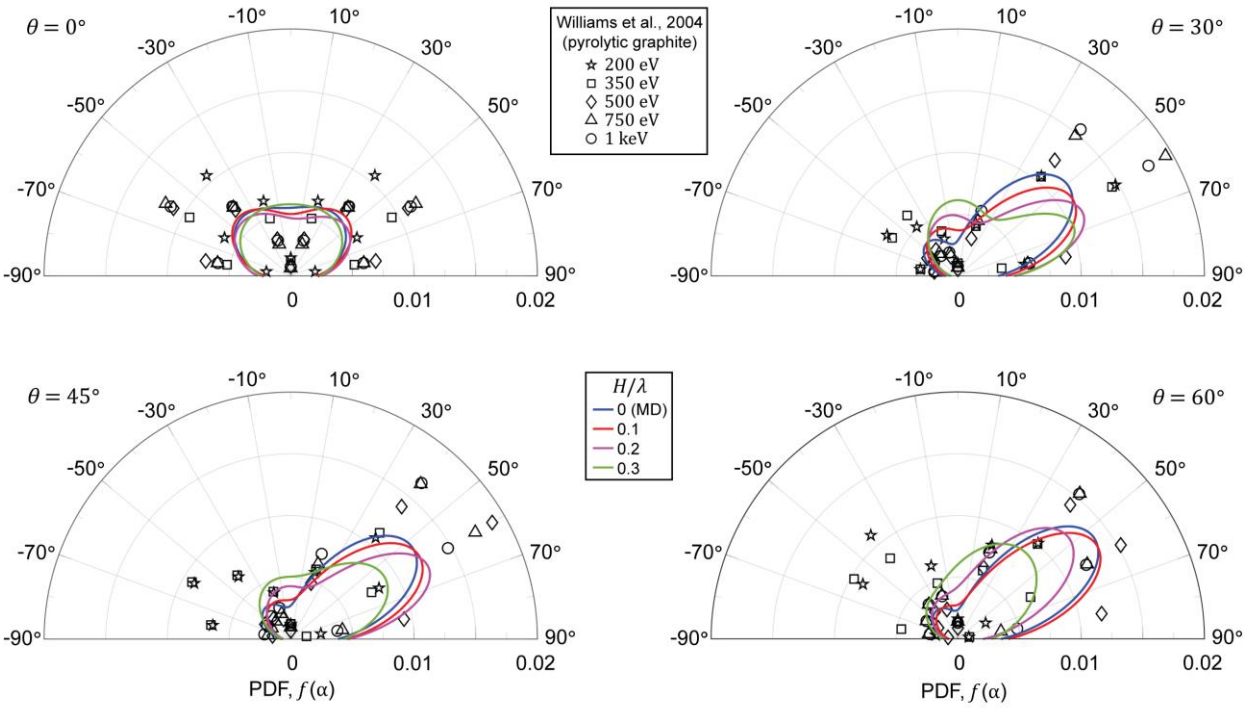

Figure 3.14: Changes in the PDF of the differential yield profile of the sputterants, $f(\alpha)$, for different $\frac{H}{\lambda}$ at $\theta =$ 0°, 30°, 45° and 60°. Open black symbols denote the experimentally-measured probability distribution with a bin size of $\pm$ 10° for five different ion energies [24].

As aforementioned, differential yield profile measurements for carbon sputtering within the low ion energy regime ($\leq$ 1 keV) of interest are extremely limited. Using a Quartz Crystal Microbalance (QCM), Williams et al. [24] reported the differential yield profile as a function of

polar angle α by measuring the quantity of detected carbon sputterants per incident xenon ion per unit steradian. However, the measured data for each polar angle α was only collected at a fixed azimuthal angle, and a uniform sputterant distribution was assumed along the azimuthal direction. We include in Fig. 3.14 the equivalent $f(\alpha)$ from data provided by Williams et al. [24] at various E (black symbols). The experimental data for $E \geq 500$ eV are in reasonable agreement with our MD-predicted $f(\alpha)$ spanning the range of possible surface morphologies ($0 \leq \frac{H}{\lambda} \leq 0.3$). The experimental data, however, deviates from our simulation results in the backward scatter directions (i.e., $\alpha < 0°$) for $E \leq 350$ eV, and is in contrast to our MD results which show $f(\alpha)$ to be independent of E (Fig. 3.9). Presumably, this discrepancy arises from the assumption of uniform distribution of sputterants along the azimuthal angle in the experimental measurements, which artificially enforces $f(\alpha) \to 0$ when $\alpha \to 0°$. In comparison, we account for all detected carbon sputterants along the azimuthal angles for each polar angle data in our construction of $f(\alpha)$ from MD simulations (Fig. 3.9).

3.4 Discussions

A starting point of predicting the sputtering yields for any ion-target combination is through semi-empirical analytical formulas, including those by Sigmund [38], Yamamura & Shindo [39], and Eckstein & Preuss [34]. The free parameters of these formulas are commonly fitted to models based on binary collision approximation, such as SRIM or TRIM [5,15]. However, the differences between the sputtering yield data on carbon from experiments (black symbols in Fig. 3.8) and those from SRIM or TRIM models (purple symbols in Fig. 3.8) can differ by several orders of magnitude at lower ion energies of ≤ 300 eV. Studies have also fitted these semi-analytical formulas to experimental sputtering yields on carbon under low ion energies [12,37]. However, the experimental data for several polymorphs of carbon (black symbols in Fig. 3.8) shows a large

spread between measurements conducted in different laboratories. In addition, there is also no obvious correlation between the sputter yield data and the various polymorphs of carbon, including graphite, diamond, and carbon composites. This is counterintuitive, in view of the large differences in sp^3 (711 kJ/mol) versus sp^2 (524 kJ/mol) bond strengths, coupled with the anisotropy character of the various forms and orientations of graphite caused by the relatively weak bonding between hexagonal planes of atoms. In fact, Tartz et al. [11] found no appreciable differences between the sputtering yield measurements of carbon-carbon, pyrolytic graphite, and high-density graphite of various grain sizes, with xenon ion energies ranging between 400 eV and 1.4 keV. Similarly, William et al. [24] reported nearly identical differential and total sputter yield measurements of carbon-carbon composite and pyrolytic graphite across $200 \text{ eV} \leq E \leq 1 \text{ keV}$ and $0° \leq \theta \leq 60°$.

Through MD simulations on the xenon ion bombardment of h-MLG, v-MLG, and diamond, we show that the bombardment process creates the same amorphous subsurface for each of these carbon structures once steady-state sputtering is achieved. Each of these amorphous subsurface structures possess similar atomic densities (porosities) and proportions of $sp/sp^2/sp^3$ bond types, and hence are virtually indistinguishable from one another. Their structural characteristics also remain unchanged with further ion bombardment. This explains why the sputtering yield under steady-state bombardment is independent of the initial carbon crystal structure reported in experiments [11,24]. Furthermore, our simulations show that the same unique amorphous subsurface structure is attained under different ion energies and incidence angles, implying that the sputtering measurements at steady-state are also independent of the prior sputtering history. Ultimately, these results suggest that our steady-state sputtering measurements from MD are applicable to a wide array of carbon structures subjected to sputter erosion. In the application of electric space propulsion, this includes pyrolytic graphite used for gridded ion optics, diamond

films for coatings of ion thruster electrodes, isotropic graphite for pole covers of Hall effect thrusters, and poco or anisotropic graphite used to line the panels of ground-based testing facilities of ion thrusters [24,28,40,41].

Performing large-scale atomistic simulations to account for microscale surface morphology effects is challenging because of the disparate length-scales. Instead, we idealize the surface geometry and consider changes to the sputtering yield (obtained from MD) caused by (a) changes to the local ion incidence angle at each impact point along the surface, (b) redeposition of sputterants which impact with surface features, as well as (c) partial shielding of the surface from incoming ions by surface features leading to a shadowing or masking effect. By augmenting the MD predictions to account for surface morphology effects of amorphous carbon films, our sputter yield predictions are well within the experimental scatter across the range of ion energies and incidence angles (Fig. 3.12). This allows us to calibrate semi-empirical sputter yield models with MD data for the first time, which is especially critical at lower ion energies because of the large scatter in experimental data caused by low sputter yields. Building on the high accuracy of our MD sputtering yield predictions, we further obtain differential yield profiles of the sputterants, which are necessary for quantifying the contribution of facility effects associated with carbon backsputtering. This differential yield data from MD complements the extremely limited set of experimental data available in the literature, which are highly non-trivial to obtain [24,30,42].

Surface morphology effects have been proposed to explain the differing sputtering yield data by Deltschew et al. [20] versus Williams et al. [24] and Tartz et al. [11] on carbon-carbon composites. While Deltschew et al. [20] demonstrated a two-fold higher sputtering yield for carbon-carbon composite compared to graphite, Williams et al. [24] and Tartz et al. [11] reported very similar sputtering yields between these two carbon structures – a conclusion more in line with

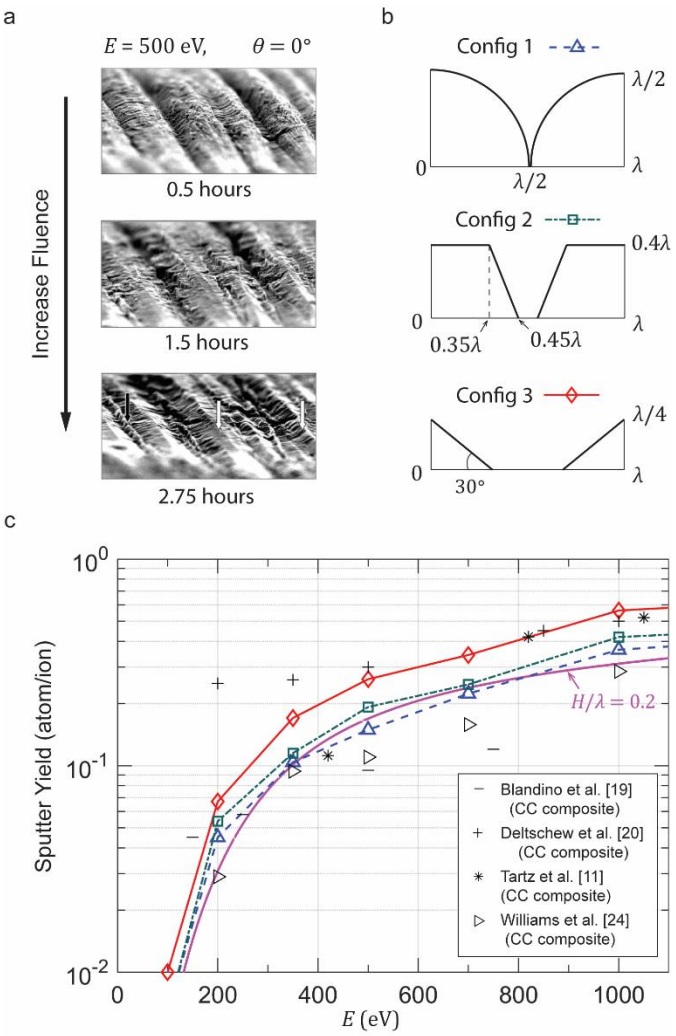

Figure 3.15: (**a**) SEM imaging of the evolving surface morphology of the exposed carbon fiber under xenon ion bombardment with $E = 500$ eV and $\theta = 0°$ [20]. (**b**) Idealization of the evolving surface morphology in (**a**). (**c**) Steady-state sputtering yield across different E with $\theta = 0°$, comparing experiments for Carbon-Carbon composites with MD predictions based on the varying surface configurations in (**b**). Purple line denotes the maximum likelihood predictions of the Eckstein & Preuss [34] sputter yield model at $\frac{H}{\lambda} = 0.2$ (**Fig. 3.13**).

our MD results. Unlike the carbon composite targets used by Williams et al. [24], however, the composite material characterized by Deltschew et al. [20] was not infiltrated with carbon or pyrolytically coated, and consequently, the carbon fibers were directly exposed to the ion beam. Thus, it was assumed by Williams et al. [24] that the exceptionally higher sputter yield for the carbon composite targets by Deltschew et al. [20] was due to the cylindrical fiber morphology (Fig. 3.15a and 3.15b – configuration 1). We show in Fig. 3.15c, however, that the sputtering yield caused by this cylindrical fiber morphology (blue dashed line), obtained by modifying (3) to account for the surface morphology in Fig. 3.15b – configuration 1, is not too different from that associated with a cosine surface morphology defined by $\frac{H}{\lambda} = 0.2$ (purple solid curve) which we

have assumed for the various carbon polymorphs. However, Deltschew et al. [20] showed that the surface morphology of these carbon fibers also evolves with ion fluence (Fig. 3.15a). By idealizing these evolving surface geometries in Fig. 3.15b and modifying equations (2) and (3) accordingly, we show that the evolving sputter yield predictions transcending through these surface configurations appear to agree quantitatively with the experimental data of Deltschew et al. [20], at least for the higher ion energy values ($E \geq 500$ eV). We remark that at lower ion energies, the evolving morphology can be different from those reproduced in Fig. 3.15a for $E = 500$ eV.

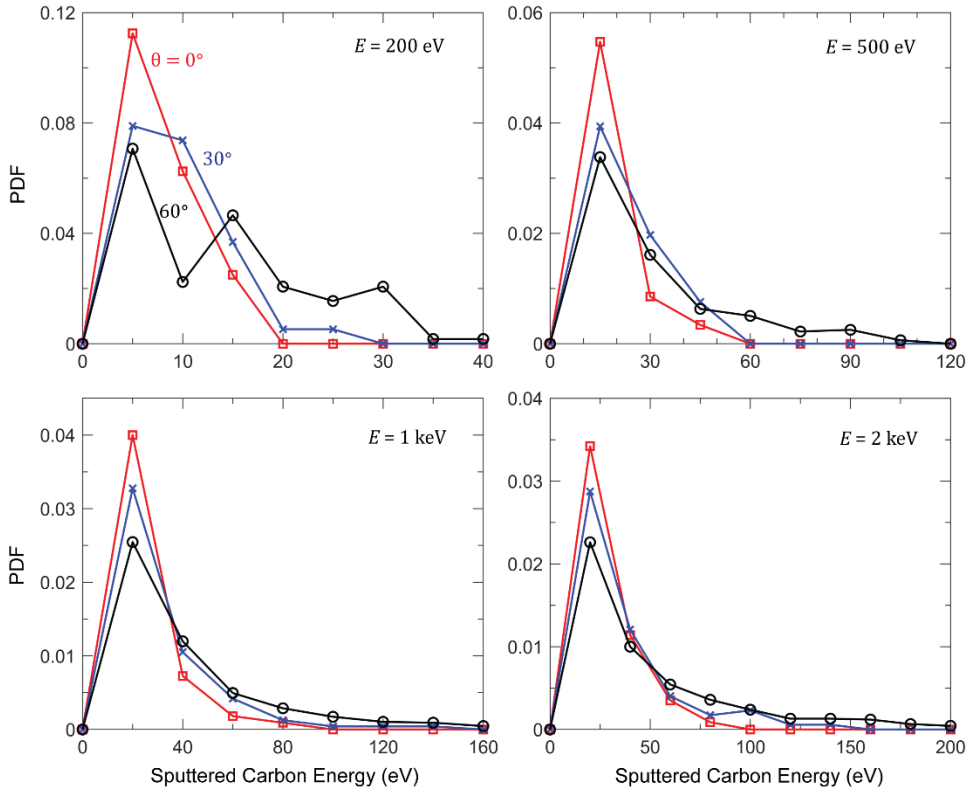

Figure 3.16: Probability distribution function (PDF) of the energy of the sputtered carbon atoms for $E = 200$, 500, 1000, and 2000 eV with $\theta = 0°$, 30° and 60°.

Our above analysis suggests the importance of accounting for the evolving surface morphology with ion fluence. Certainly, the surface features will change with the removal or redeposition of carbon sputterants, especially under high fluence (Fig. 3.15a). In addition, our MD simulations show that the energy of the carbon sputterants can range from 5 to 40 eV for $E = 200$ eV and as high as 25 to 200 eV for $E = 2$ keV (see sputterant energy distributions in Fig.

3.16); the impact of the higher energy carbon sputterants on surface features can initiate further sputtering. In addition, the surface morphology may be affected by the entrapment of xenon ions within the subsurface, as quantified by the xenon ion absorption rate (Fig. 3.4). Surface roughening caused by these effects may in turn be offset by smoothening due to surface diffusion operative at longer time-scales [31,43]. Accounting for these complexities require Monte Carlo simulations, which is presented in Chapter 4.

3.5 Conclusions

Large-scale, massively-parallel MD simulations were conducted to elucidate the sputtering yield of carbon substrates across the range of ion energies and incidence angles of relevance to surface erosion in ion thrusters for electric space propulsion. Our extensive simulation data complements the sparse experimental datasets of carbon sputtering yield and differential yield profile, in an ion energy regime (< 1 keV) where obtaining statistically-accurate sputtering measurements becomes challenging in view of the very low carbon sputter rates. Our simulations uncovered the fundamental mechanism underpinning similar differential yield profiles and sputtering yields associated with different carbon target materials reported in experiments. We show that the bombardment of low energy heavy xenon ions creates an amorphous subsurface which, under steady-state conditions, possess the same structural characteristics (atomic density, $sp/sp^2/sp^3$ bond proportion) regardless of the initial carbon structure or the sputtering conditions. By upscaling our MD predictions to account for surface morphology effects, our sputter yield predictions were well within the experimental scatter, and could quantitatively explain the contrasting experimental results associated with the unique and evolving surface morphology of exposed carbon fibers. Parameters of a semi-empirical sputter-yield model were calibrated using this MD dataset, to provide a reduced order representation of the sputtering yield for plasma-surface interactions.

3.6 References

[1] A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in 't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, R. Shan, M.J. Stevens, J. Tranchida, C. Trott, S.J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Computer Physics Communications 271 (2022) 108171. https://doi.org/10.1016/j.cpc.2021.108171.

[2] S.J. Stuart, A.B. Tutein, J.A. Harrison, A reactive potential for hydrocarbons with intermolecular interactions, J. Chem. Phys. 112 (2000) 6472–6486. https://doi.org/10.1063/1.481208.

[3] C. Qian, B. McLean, D. Hedman, F. Ding, A comprehensive assessment of empirical potentials for carbon materials, APL Materials 9 (2021) 061102. https://doi.org/10.1063/5.0052870.

[4] M. Wen, E.B. Tadmor, Hybrid neural network potential for multilayer graphene, Phys. Rev. B 100 (2019) 195419. https://doi.org/10.1103/PhysRevB.100.195419.

[5] J.F. Ziegler, J.P. Biersack, The Stopping and Range of Ions in Matter, in: D.A. Bromley (Ed.), Treatise on Heavy-Ion Science: Volume 6: Astrophysics, Chemistry, and Condensed Matter, Springer US, Boston, MA, 1985: pp. 93–129. https://doi.org/10.1007/978-1-4615-8103-1_3.

[6] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Computational Materials Science 6 (1996) 15–50. https://doi.org/10.1016/0927-0256(96)00008-0.

[7] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169–11186. https://doi.org/10.1103/PhysRevB.54.11169.

[8] G. Kresse, J. Hafner, Ab initio molecular dynamics for liquid metals, Phys. Rev. B 47 (1993) 558–561. https://doi.org/10.1103/PhysRevB.47.558.

[9] M.S. Daw, M.I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Phys. Rev. B 29 (1984) 6443–6453. https://doi.org/10.1103/PhysRevB.29.6443.

[10] R.B. Lobbia, J.E. Polk, R.R. Hofer, V.H. Chaplin, B. Jorns, Accelerating 23,000 hours of Ground Test Backsputtered Carbon on a Magnetically Shielded Hall Thruster, in: AIAA Propulsion and Energy 2019 Forum, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-3898.

[11] M. Tartz, Pyrolytic Graphite and Carbon-carbon Sputter Behaviour Under Xenon Ion Incidence, in: 29th International Electric Propulsion Conference, 2005: p. 10.

[12] R. Kolasinski, J. Polk, D. Goebel, L. Johnson, Carbon Sputtering Yield Measurements at Grazing Incidence, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, Sacramento, California, 2006. https://doi.org/10.2514/6.2006-4337.

[13] R.D. Kolasinski, J.E. Polk, D. Goebel, L.K. Johnson, Sputtering yield measurements at glancing incidence using a quartz crystal microbalance, Journal of Vacuum Science & Technology A 25 (2007) 236–245. https://doi.org/10.1116/1.2435375.

[14] H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–72. https://doi.org/10.1145/174462.156635.

[15] J.F. Ziegler, M.D. Ziegler, J.P. Biersack, SRIM – The stopping and range of ions in matter (2010), Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 268 (2010) 1818–1823. https://doi.org/10.1016/j.nimb.2010.02.091.
[16] J.F. Ziegler, The stopping of energetic ions in solids, Nuclear Instruments and Methods 168 (1980) 17–24. https://doi.org/10.1016/0029-554X(80)91225-2.
[17] W.D. Wilson, L.G. Haggmark, J.P. Biersack, Calculations of nuclear stopping, ranges, and straggling in the low-energy region, Phys. Rev. B 15 (1977) 2458–2468. https://doi.org/10.1103/PhysRevB.15.2458.
[18] M. Titze, J.L. Pacheco, T. Byers, S.B. Van Deusen, D.L. Perry, D. Weathers, E.S. Bielejec, Evaluation of the accuracy of stopping and range of ions in matter simulations through secondary ion mass spectrometry and Rutherford backscattering spectrometry for low energy heavy ion implantation, Journal of Vacuum Science and Technology A 39 (2021). https://doi.org/10.1116/6.0001406.
[19] J.J. Blandino, D.G. Goodwin, C.E. Garner, Low energy sputter yields for diamond, carbon–carbon composite, and molybdenum subject to xenon ion bombardment, Diamond and Related Materials 9 (2000) 1992–2001. https://doi.org/10.1016/S0925-9635(00)00350-2.
[20] R. Deltschew, M. Tartz, V. Plicht, E. Hartmann, H. Neumann, Sputter characteristics of carbon-carbon compound material, in: 27th International Electric Propulsion Conference, 2001.
[21] R.P. Doerner, D.G. Whyte, D.M. Goebel, Sputtering yield measurements during low energy xenon plasma bombardment, Journal of Applied Physics 93 (2003) 5816–5823. https://doi.org/10.1063/1.1566474.
[22] J. Gruber, Low-Energy Sputter Erosion of Various Materials in a T 5, in: 27th International Electric Propulsion Conference, 2001.
[23] D. Rosenberg, G.K. Wehner, Sputtering Yields for Low Energy He^+-, Kr^+-, and Xe^+-Ion Bombardment, Journal of Applied Physics 33 (1962) 1842–1845. https://doi.org/10.1063/1.1728843.
[24] J. Williams, M. Johnson, D. Williams, Differential Sputtering Behavior of Pyrolytic Graphite and Carbon-Carbon Composite Under Xenon Bombardment, in: 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, American Institute of Aeronautics and Astronautics, Fort Lauderdale, Florida, 2004. https://doi.org/10.2514/6.2004-3788.
[25] J. Bohdansky, Important sputtering yield data for tokamaks: A comparison of measurements and estimates, Journal of Nuclear Materials 93–94 (1980) 44–60. https://doi.org/10.1016/0022-3115(80)90302-5.
[26] M.J. Pellin, R.B. Wright, D.M. Gruen, Laser fluorescence spectroscopy of sputtered zirconium atoms, J. Chem. Phys. 74 (1981) 6448–6457. https://doi.org/10.1063/1.440983.
[27] S. Hadrath, J. Ehlbeck, G. Lieder, F. Sigeneger, Determination of absolute population densities of eroded tungsten in hollow cathode lamps and fluorescent lamps by laser-induced fluorescence, J. Phys. D: Appl. Phys. 38 (2005) 3285. https://doi.org/10.1088/0022-3727/38/17/S33.
[28] G.C. Soulas, The Impact of Back-Sputtered Carbon on the Accelerator Grid Wear Rates of the NEXT and NSTAR Ion Thrusters, in: Washington, D.C., 2013. https://ntrs.nasa.gov/citations/20150021367 (accessed October 31, 2022).

[29] C.E. Huerta, T.S. Matlock, R.E. Wirz, View factor modeling of sputter-deposition on micron-scale-architectured surfaces exposed to plasma, Journal of Applied Physics 119 (2016) 113303. https://doi.org/10.1063/1.4944035.
[30] M.J. Schaible, C.A. Dukes, A.C. Hutcherson, P. Lee, M.R. Collier, R.E. Johnson, Solar Wind Sputtering Rates of Small Bodies and Ion Mass Spectrometry Detection of Secondary Ions, Journal of Geophysical Research: Planets 122 (2017) 1968–1983. https://doi.org/10.1002/2017JE005359.
[31] R.M. Bradley, J.M.E. Harper, Theory of ripple topography induced by ion bombardment, Journal of Vacuum Science & Technology A 6 (1988) 2390–2395. https://doi.org/10.1116/1.575561.
[32] E. Oyarzabal, R.P. Doerner, M. Shimada, G.R. Tynan, Carbon atom and cluster sputtering under low-energy noble gas plasma bombardment, Journal of Applied Physics 104 (2008) 043305. https://doi.org/10.1063/1.2968549.
[33] S. Habenicht, Morphology of graphite surfaces after ion-beam erosion, Phys. Rev. B 63 (2001) 125419. https://doi.org/10.1103/PhysRevB.63.125419.
[34] W. Eckstein, R. Preuss, New fit formulae for the sputtering yield, Journal of Nuclear Materials 320 (2003) 209–213. https://doi.org/10.1016/S0022-3115(03)00192-2.
[35] C. García-Rosales, W. Eckstein, J. Roth, Revised formulae for sputtering data, Journal of Nuclear Materials 218 (1995) 8–17. https://doi.org/10.1016/0022-3115(94)00376-9.
[36] R.C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, Society for Industrial and Applied Mathematics, USA, 2013.
[37] J.T. Yim, Yim, J.T. (2017). A Survey of Xenon Ion Sputter Yield Data and Fits Relevant to Electric Propulsion Spacecraft Integration., in: 35th International Electric Propulsion Conference, 2017.
[38] P. Sigmund, Sputtering by ion bombardment theoretical concepts, in: R. Behrisch (Ed.), Sputtering by Particle Bombardment I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981: pp. 9–71. https://doi.org/10.1007/3540105212_7.
[39] Y. Yamamura, S. Shindo, An empirical formula for angular dependence of sputtering yields, Radiation Effects 80 (1984) 57–72. https://doi.org/10.1080/00337578408222489.
[40] P.M. Sforza, Chapter 14 - Space Propulsion, in: P.M. Sforza (Ed.), Theory of Aerospace Propulsion, Butterworth-Heinemann, Boston, 2012: pp. 541–565. https://doi.org/10.1016/B978-1-85617-912-6.00014-1.
[41] D.M. Goebel, B. Jorns, R.R. Hofer, I.G. Mikellides, I. Katz, Pole-piece Interactions with the Plasma in a Magnetically Shielded Hall Thruster, in: 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, American Institute of Aeronautics and Astronautics, 2014. https://doi.org/10.2514/6.2014-3899.
[42] A.P. Yalin, J.D. Williams, V. Surla, K.A. Zoerb, Differential sputter yield profiles of molybdenum due to bombardment by low energy xenon ions at normal and oblique incidence, J. Phys. D: Appl. Phys. 40 (2007) 3194. https://doi.org/10.1088/0022-3727/40/10/025.
[43] R.L. Headrick, H. Zhou, Ripple formation and smoothening on insulating surfaces, J Phys Condens Matter 21 (2009) 224005. https://doi.org/10.1088/0953-8984/21/22/224005.

CHAPTER 4: MORPHOLOGY EVOLUTION OF CARBON SURFACES UNDER ION BOMBARDMENT: MONTE CARLO SIMULATIONS [1]

[1] *Extracts of this chapter can be found in* Tran and Chew, Acta Materialia, 263, 119498

In Chapter 3, molecular dynamics (MD) simulations have established the sputtering yields of carbon under the low-energy bombardment of noble gas ions. An analytical model has also been proposed to account for changes in the sputtering yield and differential yield associated with a *stationary* sinusoidal surface morphology, which does not evolve with ion fluence. In this chapter, we adopt a more realistic approach, by extending the length- and time-scales of these MD simulations through a Monte Carlo model to elucidate the ion-surface sputtering mechanisms, *evolving* surface morphology, and resulting sputtering yield associated with micro-scale surface roughness (Fig. 4.1a) under sputtering conditions relevant to spacecraft electric propulsion. Section 4.1 describes the Monte Carlo (MC) modeling approach and the underlying MD-derived sputtering characteristics. We detail in Section 4.2 the fundamental ion-surface erosion and deposition mechanisms leading to surface texturing under normal and oblique ion incidence angles and present the transient and steady-state sputtering yields associated with the evolving surface morphologies in Section 4.3. We expand our MC simulations to include a wide range of initial surface morphologies in Section 4.4, and explore design strategies to control and abate sputtering by surface texturing. Finally, we discuss the implications of our Monte Carlo simulations in the context of prior experiments in Section 4.5 and conclude with a summary in Section 4.6.

4.1 Simulation Methodology

A variety of physical processes can occur when an incoming xenon ion impinges on a carbon surface, as outlined in Fig. 4.1b. The xenon ion (red arrow) can penetrate the surface or recoil from

the surface, and cause scission of carbon-carbon bonds in the process to initiate sputtering [1]. These (primary) carbon sputterants (blue arrows) can escape from the surface, or impact with other surface features and potentially trigger the emission of secondary carbon sputterants (purple arrows). In turn, this process can trigger the emission of tertiary carbon sputterants. However, the secondary carbon sputterants tend to get absorbed upon impact with surface features due to their low energies (~1 eV). Here, we will adopt a Monte Carlo approach to simulate the evolving surface morphology from two competing mechanisms: (i) surface recession due to sputtering induced by

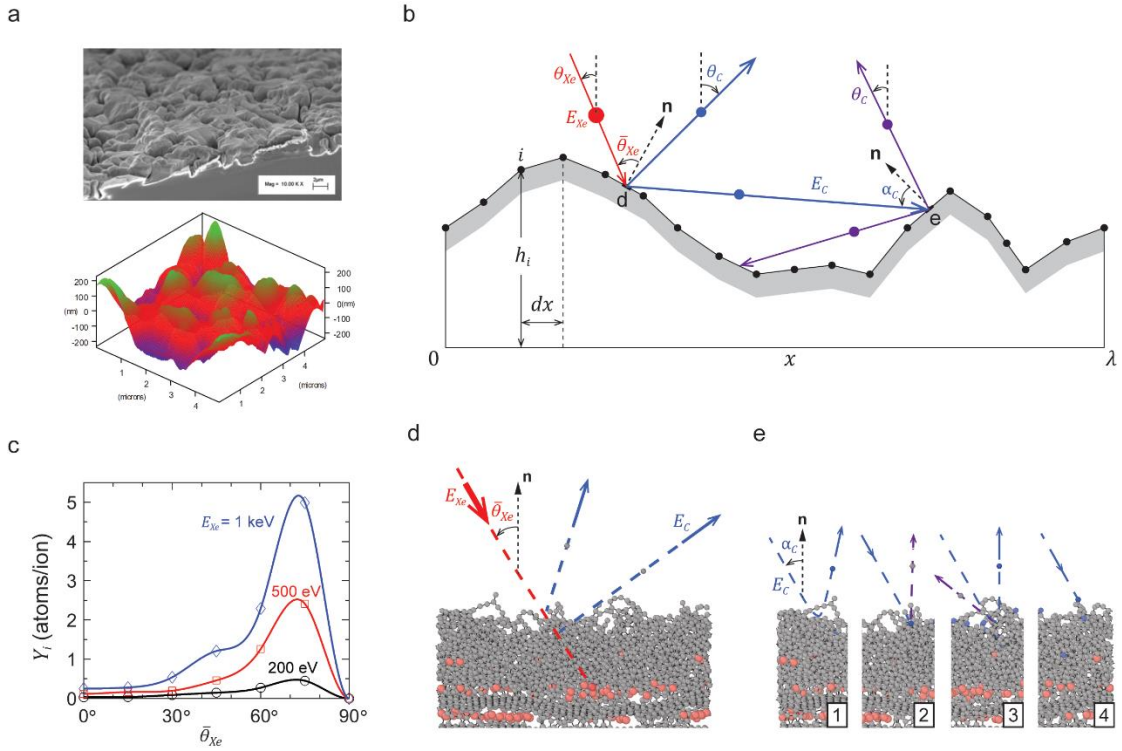

Figure 4.1: (a) Surface morphology of amorphous carbon characterized by scanning electron microscopy (SEM) and atomic force microscopy (AFM) [2]. **(b)** Monte Carlo modeling of sputtering erosion cd by the impingement of xenon ions (red) on an undulating carbon surface, knocking-off primary (blue) and secondary (purple) carbon sputterants. **(c)** Elemental sputtering yield (Y_i) as a function of local ion incidence angle, $\bar{\theta}_{Xe}$, for three ion incidence energies, $E_{Xe} = 200, 500, 1000$ eV; symbols denote the steady-state sputtering yield from MD simulations connected with a spline fitting, with exception for $\bar{\theta}_{Xe} = 90°$ where $Y_i = 0$ is assumed. **(d,e)** MD simulations of the energetic bombardment of amorphous carbon substrates with xenon ions (**d**) and carbon atoms (**e**), to quantify the sputtering characteristics of primary (**d**) and secondary carbon sputterants (**e**), which govern the underlying elemental properties of the MC model.

surface impact of xenon ions and primary carbon sputterants, and (ii) surface growth due to deposition of primary and secondary carbon sputterants.

On one hand, studies focusing on nanoscale texturing have uncovered a number of other physical phenomena that can significantly influence surface morphology evolution and surface patterning at the nanoscale (roughness wavelength of $< \sim$100 nm), including ion-induced mass redistribution, curvature-dependent sputtering, and ion-induced viscous flow [3–5]. In particular, mass redistribution is an important destabilizing effect that contributes to the formation of nanoscale surface ripples for sufficiently high angles of ion incidence [4,6]. On the other hand, our previous MD simulations on the sputtering of carbon surfaces also reveal that both ion-induced mass redistribution and ion-induced viscous flow contribute to rapid amorphization of the carbon subsurface, but the structural characteristics ($sp/sp^2/sp^3$ bond proportions, atomic density) eventually plateau once steady-state bombardment of the MD simulation box (with 5×5 nm^2 exposed surface to the incident ions) is attained at the time-scale of $\sim10^{14}$ ions/cm^2. This time-scale is several orders of magnitude smaller than the time-scale required to evolve the carbon surfaces with micron-scale undulations (Fig. 4.1a) to reach steady-state in experiments ($\sim10^{19}$ to 10^{20} ions/cm^2) [7,8], which is the focus of our studies. This difference allows us to delineate the local nanoscale effects of ion-induced mass redistribution and viscous flow from microscale surface morphology evolution. Moreover, we show that we attain the same amorphous carbon structure regardless of the ion incidence energies and angles. Therefore, we homogenize the nanoscale sputtering response (and consequently, the physical effects of ion-induced mass redistribution and viscous flow) from MD [1] into a single representative element in our Monte Carlo approach (Fig. 4.1b), where the sputtering yield of each element is based on the steady-state sputtering response from MD.

Scanning electron microscopy (SEM) and atomic force microscopy (AFM) measurements of amorphous carbon structures (Fig. 4.1a) show the surfaces to have height variations, H, of ~200 nm, with micro-scale roughness wavelength, λ, of ~2 μm, which infers a surface roughness of $\frac{H}{\lambda} =$ ~0.2 to 0.25). We consider these microscale morphology as one-dimensional surfaces that are periodic along the x-direction. We represent each surface within its period length λ with n two-noded elements, each element of uniform dimension, $dx = \frac{\lambda}{n}$ along the x-direction (Fig. 4.1b); the first ($i = 1$) and last nodes ($i = n + 1$) are tied together to enforce surface continuity across the periodic domain. (a) Incident xenon atoms (red arrows in Fig. 4.1) are introduced into the system at random positions above the carbon surface with assigned kinetic energy E_{Xe}, and are propagated along the assigned global ion incidence angle θ_{Xe} until the ion path intersects with a surface element, indicating a point of impact. (b) Since each element represents the homogenized response from MD [1], we compute the number of sputtered primary carbon atoms based on the steady-state sputtering yield data, Y_i, from MD (Fig. 4.1c), which is a function of both E_{Xe} and the local ion incidence angle $\bar{\theta}_{Xe}$ at the impact site [1]. For example, a sputter yield of 1.2 atoms/ion implies a 100% probability of sputtering a first primary carbon atom, and a 20% probability of sputtering a second primary carbon atom from the impact site. (c) For each sputtered primary carbon atom (blue arrows, Fig. 4.1d), we also compute the corresponding energy E_C and trajectory, which are statistically selected based on the cumulative distribution functions of the sputtered carbon angles [1] and energies (see Table 4.1 and Fig. 4.2) from prior MD simulations. (d) Tracing the trajectory of each of these primary carbon sputterants (blue arrows, Fig. 4.1b), we determine if they intersect another surface element or if they escape from the simulation box. (e) In the case of the former, we compute the probability of emission of secondary carbon sputterants from the impact site and sample the trajectories of these secondary sputterants (purple arrows, Fig. 4.1a) from a cosine

distribution [9]. (f) Should these secondary carbon sputterants impact another surface element, we assume the sputterants to be absorbed at the impact site.

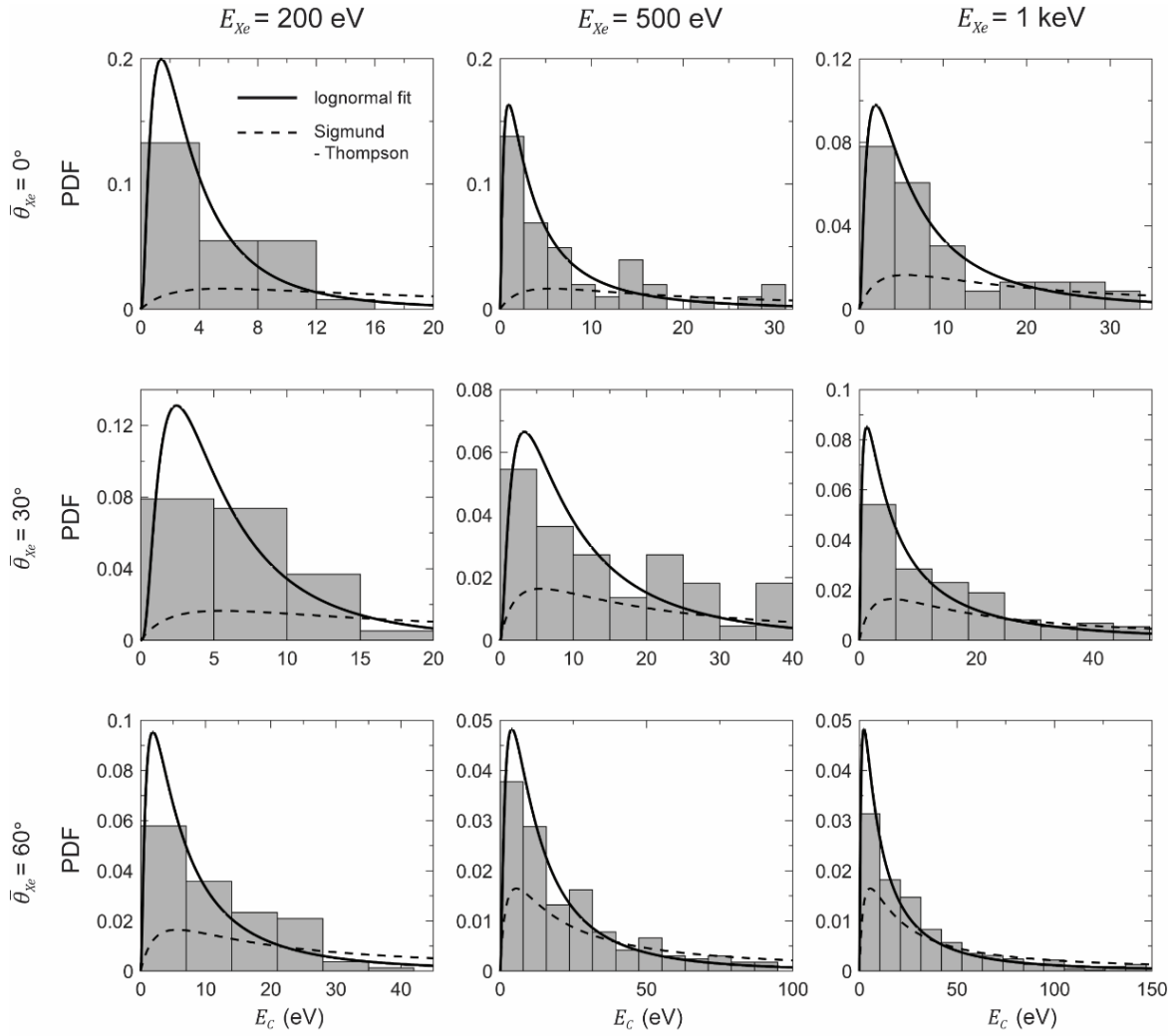

Figure 4.2: Histogram of the probability distribution function (PDF) of the energy of the sputtered primary carbon atoms (E_C) under a xenon ion fluence with $E_{Xe} = 200, 500, 1000$ eV and $\bar{\theta}_{Xe} = 0°, 30°, 60°$ from molecular dynamics (MD) simulations [1]. Dash curves denote the PDF based on Sigmund-Thompson model [10], which assumes E_C is independent of E_{Xe} and $\bar{\theta}_{Xe}$. The obtained E_C in the Monte Carlo model is based on distribution fits of the MD-derived PDFs for each $(E_{Xe}, \bar{\theta}_{Xe})$ pair to a log-normal functional form (solid curves)

$$\frac{1}{E_C \sigma_s \sqrt{2\pi}} e^{-\frac{\ln(E_C)-\mu_s}{\sigma_s}} \tag{4.1}$$

where σ_s and μ_s are fitting parameters summarized in **Table 4.1**.

Table 4.1: Fitting parameters (μ_s, σ_s) as a function of local xenon ion incidence angle ($\bar{\theta}_{Xe}$) and energy (E_{Xe}) for the log normal distribution function in Eq. (S1), which is used to describe the PDF of the primary sputtered carbon energy, E_C, in the Monte Carlo simulations.

μ_S						σ_S					
E_{Xe} (eV) / $\bar{\theta}_{Xe}$	**200**	**350**	**500**	**1000**	**1500**	E_{Xe} (eV) / $\bar{\theta}_{Xe}$	**200**	**350**	**500**	**1000**	**1500**
0°	1.197	0.953	1.437	1.919	2.168	**0°**	0.924	1.004	1.222	1.119	1.280
15°	1.151	1.054	1.317	1.971	2.446	**15°**	1.157	1.250	1.171	1.200	1.224
30°	1.634	1.896	2.293	2.181	2.620	**30°**	0.860	1.075	1.047	1.390	1.195
45°	1.493	2.004	2.424	2.526	2.793	**45°**	1.052	1.188	0.944	1.370	1.226
60°	1.952	2.296	2.624	2.740	2.811	**60°**	1.149	1.124	1.108	1.377	1.334
75°	2.367	2.774	2.846	3.009	3.293	**75°**	0.876	0.959	1.048	1.224	1.225

We remark that process (e) above necessitates new MD simulations to elucidate the sputtering yield of secondary carbon sputterants caused by the bombardment of primary carbon sputterants on amorphous carbon substrates. Following the same MD modeling approach as before [1], we subject an amorphous carbon substrate, created by the bombardment of xenon ions under steady-state sputtering conditions, to the bombardment of carbon atoms with incidence energies E_C of 1 to 150 eV and across local ion incidence angles α_C of $0°, 30°, 60°, 75°$. Note that this E_C range corresponds to the energy range reported for primary carbon sputterants (Fig. 4.2). As shown in Fig. 4.1e, the primary carbon atoms (blue spheres along the blue trajectories) can recoil upon impact or can be deposited onto the surface; these processes can potentially trigger the release of secondary carbon sputterants (purple trajectories). For simplicity, we do not distinguish between these processes, and compute the probability of secondary sputterant emission as the ratio of carbon atoms exiting the MD simulation box (regardless of the carbon atom source) to the bombarded primary carbon sputterants. For each combination of E_C and α_C, we perform 30

individual bombardments and summarize our results in Table 4.2. As expected, increasing E_C and α_C tends to increase the probability of secondary carbon sputterant emission.

Table 4.2: Emission probability of (secondary) carbon sputterants, under the impact of (primary) carbon atoms with incidence energy E_C and local incidence angle α_C on amorphous carbon substrates, from MD simulations.

E_C (eV) / α_C	**1**	**2**	**5**	**15**	**30**	**50**	**80**	**100**	**150**
0°	0.03	0.13	0.13	0.07	0.10	0.10	0.07	0.13	0.30
30°	0.10	0.20	0.13	0.10	0.17	0.07	0.23	0.43	0.37
60°	0.1	0.07	0.27	0.30	0.47	0.40	0.60	0.73	0.90
75°	0.07	0.03	0.23	0.30	0.40	0.53	0.83	0.67	1.40

To reduce computational time and to allow for parallelization in our Monte Carlo simulations, we deposit 10^4 xenon ions over the periodic surface length, λ (cm), at each time instant, to obtain a fixed incremental fluence of $= 10^4/\lambda$ ($\frac{\text{ions}}{\text{cm}^2}$), assuming unit displacement (cm) in the out-of-plane direction. We do not distinguish between the primary and secondary sputterants and compute the total number and trajectory of carbon atoms escaping from the simulation domain to obtain the average sputtering yield $\hat{Y}$ (atoms/ion) and differential yield profile $f(\theta_C)$ at this fluence ξ.

During each sputter or deposition event at the impact site of each element, we linearly interpolate the decrease or increase in the number of carbon atoms from the impact site to the corresponding nodes using linear finite element shape functions. The global sputtering yield $\hat{Y}$ can also be obtained at the elemental level by summing the sputtering yield Y_i at each node

$$\hat{Y} = \sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (S_i - D_i) \tag{4.2}$$

where S_i and D_i are the number of sputtered and deposited carbon atoms for node i, respectively, per incremental fluence $d\xi$ and λ. Assuming unit displacement (cm) in the out-of-plane direction, the deposition (sputtering) of a single carbon atom at the node increases (decreases) its height h_i (cm) by $(\rho\ dx)^{-1}$ where ρ is the atomic number density of the carbon surface material. After the dynamic stochastic processes have been resolved at each $d\xi$, we update the vertical height, h_i, of each node by

$$dh_i = \frac{(D_i - S_i)\lambda d\xi}{\rho dx} \tag{4.3}$$

to "evolve" the surface. The repeated ion bombardment creates a dual layer carbon structure, comprising of an amorphous subsurface layer (shaded grey in Fig. 4.1b) of atomic number density, $\rho = 0.8 \times 10^{23}$ atoms/cm3, with thickness spanning the penetration depth (~3 nm) of the xenon ion, followed by the undamaged carbon substrate beneath [1]. The continued removal of carbon atoms from the amorphous layer by sputtering will allow xenon ions to penetrate deeper into the substrate, effectively shifting this amorphous subsurface layer of constant thickness downwards. *As such, ρ in (2) is effectively a constant.*

In theory, we are free to select any element size dx in our Monte Carlo simulations. However, a large $\frac{dx}{\lambda}$ typically results in significant numerical errors because it cannot adequately represent an undulating surface geometry, while a small $\frac{dx}{\lambda}$ leads to physical traits that are more subtle. The underlying bases of our Monte Carlo simulations are the sputtering predictions (sputtering yield, differential yield profile, sputterant energy, etc.) where the sputtering characteristics of each element are homogenized from MD simulations, i.e., all the nanoscale sputtering mechanisms (ion-induced mass redistribution and viscous flow [3,4,6]) are averaged over a 5×5 nm^2 exposed surface area in MD [1]. As such, dx represents the characteristic length-scale of local surface

features and it has to be several-folds larger than the size of the MD simulation box (~5 nm) used to obtain the steady-state sputtering yield response. AFM measurements of the surface height map of amorphous carbon films show the surfaces to have height variations (amplitude) of ~200 nm, with peak-to-peak wavelength of $\lambda = \sim 2\ \mu$m [2] (Fig. 4.1a), which suggests $\frac{dx}{\lambda} \gg 0.0025$. Here, we adopt a fixed element size of $\frac{dx}{\lambda} = 0.01$ $(dx = 20\ (nm)$ for $\lambda = 2\ \mu$m in our Monte Carlo simulations. Regardless, our numerical studies further show that the choice of $\frac{dx}{\lambda}$, at least within the bounds of 0.005 and 0.025, does not significant change the overall surface topology evolution (see Fig. 4.3).

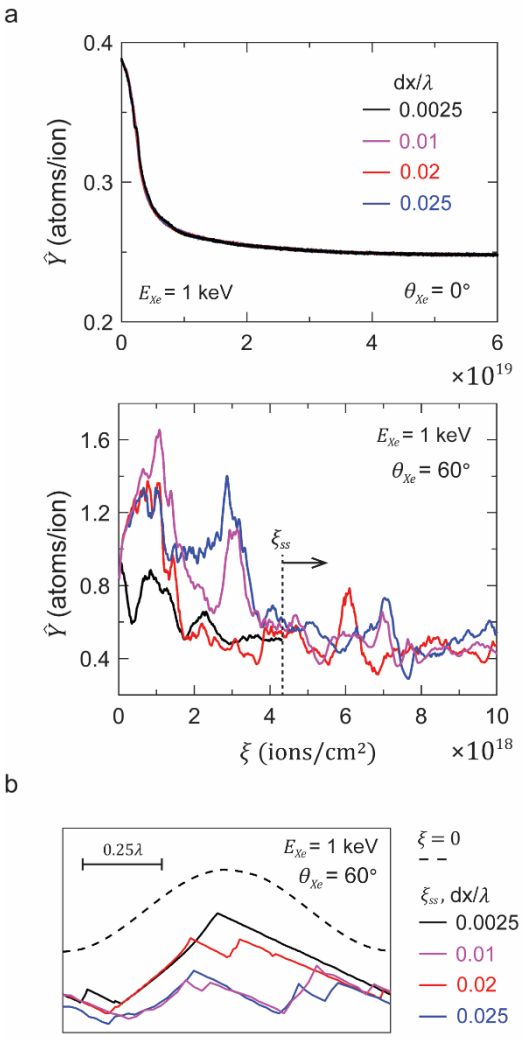

Figure 4.3: Effects of element size, $\frac{dx}{\lambda}$, on the evolution of the sputtering yield ($\hat{Y}$) with ion fluence (ξ) under $E_{Xe} = 1\ keV$, $\theta_{Xe} = 0°, 60°$ (**a**) and associated surface morphology evolutions under steady-state sputtering, $\xi = \xi_{ss}$, from the initial sinusoidal morphologies at $\xi = 0$ (**b**). Changing $\frac{dx}{\lambda}$ has negligible impact on the sputtering yields, and the morphology evolutions from sinusoidal to flat surface for $\theta_{Xe} = 0°$ and the formation of surface steps for $\theta_{Xe} = 60°$.

In our Monte Carlo simulations, we do not consider the impact of recoiling xenon ions with surface features. Our prior MD simulations show that majority of the incident xenon ions are embedded within the carbon substrate at depths of 1 to 3 nm [1]. While some of these entrapped xenon ions may diffuse out of the substrate, they do so at relatively low energies and cannot induce any further breaking of carbon-carbon bonds. We also do not consider the effects of surface reconstruction or carbon atom diffusion in our Monte Carlo simulations. We have performed MD simulations where we subject the amorphous carbon structure from xenon ion bombardment to elevated temperatures of ~2,000 K, maintained by a Berendsen thermostat for 10 ns. We observe negligible movement or reconfiguration of carbon atoms, which confirms the slow thermal-induced diffusion characteristics expected of a covalently-bonded system.

The Monte Carlo code discussed and used in this Chapter is provided in Appendix A.1.

4.2 Surface Evolution Mechanism

We elucidate the general mechanisms of sputtering for a periodic undulating surface idealized by an initial sinusoidal topology along the x-direction

$$h(x) = H\left[1 - \cos\left(\frac{2\pi x}{\lambda}\right)\right] \tag{4.4}$$

where H denotes the amplitude, and the surface roughness can be quantified by the normalized parameter $\frac{H}{\lambda}$. Motivated by AFM surface roughness measurements of amorphous carbon films [2], we adopt $\frac{H}{\lambda} = 0.25$ as our initial configuration, and quantify the morphology evolution as a function of xenon ion fluence ξ for a fixed ion energy of $E_{Xe} = 500$ eV under normal and oblique ion incidence angles ($\theta_{Xe} = 0°, 60°$). We have also repeated our simulations under different ion energies ($E_{Xe} = 200, 1000$ eV) and incidence angles ($\theta_{Xe} = 0°, 30°, 60°$) and have observed similar morphological transitions.

4.2.1 Normal ion incidence

Figure 4.4a-top shows the evolving surface configurations 1 to 5 under a normal ion incidence. We color the surface to denote the expected elemental sputter yield $Y(x) = S(x) - D(x)$ obtained by freezing the current surface to compute with Monte Carlo the average yield of sputterants S and deposited atoms D for each element over a large ion fluence to ensure statistical independence; we depict the spatial distribution of these quantities in Fig. 4.4a-bottom. For the initial geometry (Fig. 4.4a-1), the locations with the highest erosion ($Y_{\max}$) coincide with $\frac{x}{\lambda} = 0.25, 0.75$ where the local ion incidence $\bar{\theta}_{Xe}$ is the highest (~75°). Comparatively, the crest and trough ($\bar{\theta}_{Xe} = \sim 0°$) have much lower Y, particularly at the trough where $Y_{\min}$ is observed because of the confined geometry which promotes sputterant redeposition (higher D), versus the unobstructed geometry of the crest where $D = \sim 0$.

The accelerated erosion at local regimes with high $\bar{\theta}_{Xe}$ results in tapering and sharpening of the trough along with flattening of the crest (Fig. 4.4a-2). A symmetrical bilinear surface structure with bilinear distributions of $S, D,$ and Y, now develops: a perfectly flat, low erosion, regime centered at the former trough, transitioning to an inclined, higher erosion, regime with a constant slope, which peaks at the former crest (Fig. 4.4a-3). In the absence of substantial variations in D, a flat regime will always remain flat under a normal ion incidence. Continued sputtering therefore causes the self-similar size reduction of this triangular protrusion (Fig. 4.4a-4) and a perfectly microscopic flat surface (distinguished from those flat surfaces with nanoscale ripples) is eventually obtained (Fig. 4.4a-5). This flattening of surface features after long time exposure to xenon ions has been reported experimentally, as shown by SEM images in Fig. 4.4b for the sputtering of pyrolytic graphite under a normal xenon incidence ion with energy of 1 keV [8].

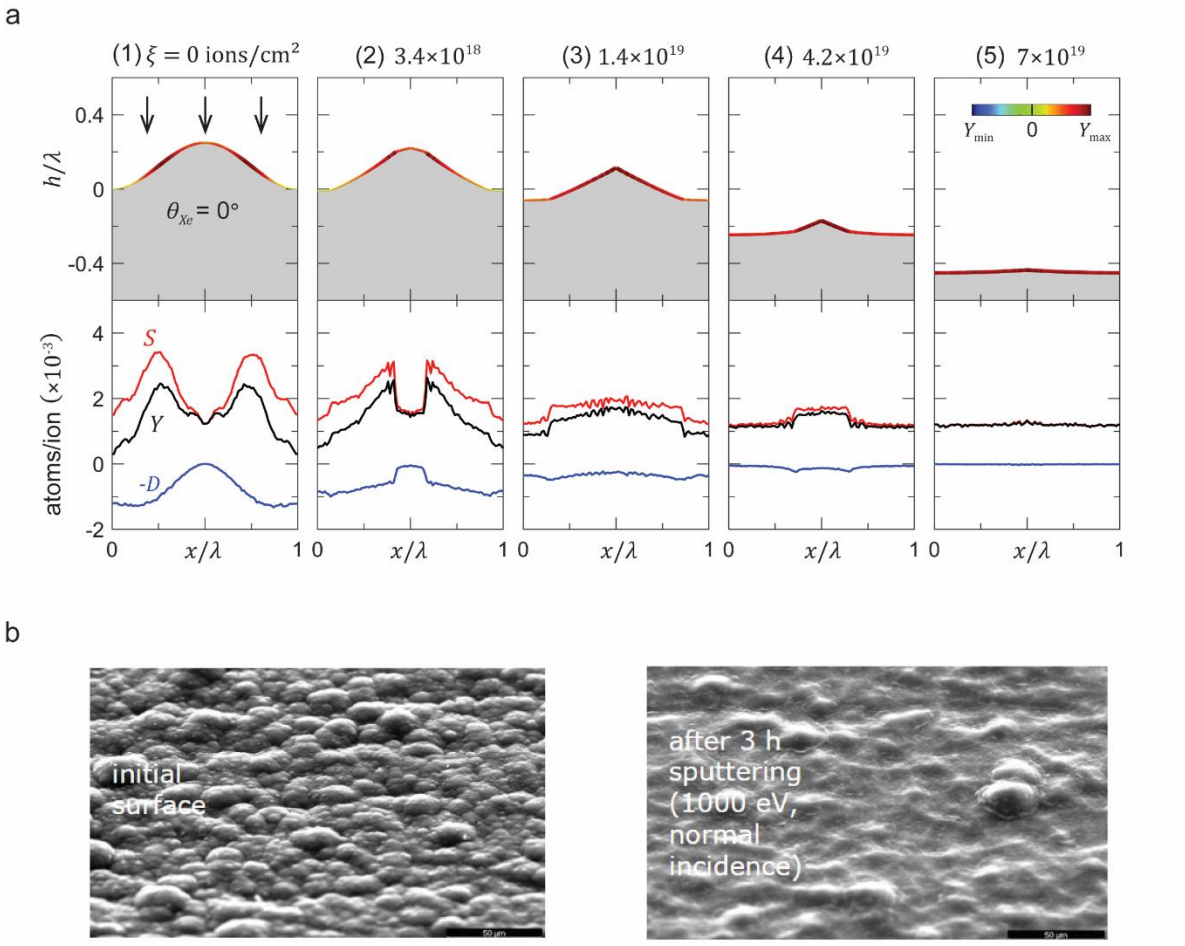

Figure 4.4: (a) Sputtering of an initial sinusoidal surface with $\frac{H}{\lambda} = 0.25$ under an ion fluence with $E_{Xe} = 500$ eV, $\theta_{Xe} = 0°$. Top: morphology evolution with ion fluence ξ; surface contours delineate elements with high yields of deposited atoms ($Y_{\min}$) and sputterants ($Y_{\max}$), respectively. Bottom: spatial distributions of the elemental yields of sputterants S and deposited atoms D, and elemental sputtering yields $Y = S - D$. (**b**) SEM imaging of pyrolytic graphite before and after three hours of sputtering with $E_{Xe} = 1$ keV, $\theta_{Xe} = 0°$, depicting distinct surface smoothening [8].

4.2.2 Oblique ion incidence

The sputtering process becomes more complicated under a highly oblique ion incidence, triggering multiple erosion and deposition mechanisms at three distinct length-scales depicted in Fig. 4.5. This is in part because of the activation of surface shielding effects which are absent under a normal ion incidence. At the length-scale spanning several elements, the competing sputtering (red

arrows) and sputter deposition process (blue arrows) can introduce small variations in the local yield, which increases the local surface (elemental) undulations (Fig. 4.5a). Relative to the back-faces of these elemental undulations that are (partially) shielded from the incoming ions, the ion-facing sides are subjected to high ion fluxes but can also trap a considerable amount of sputtered carbon species. The elemental undulations are smoothened to recover a flattened surface when sputtering effects are dominant but grow into a more distinctive microscale surface feature spanning multiple (10 or more) elements when deposition effects are dominant (Fig. 4.5a). The latter results in roughening of the overall surface topography at the micro-scale.

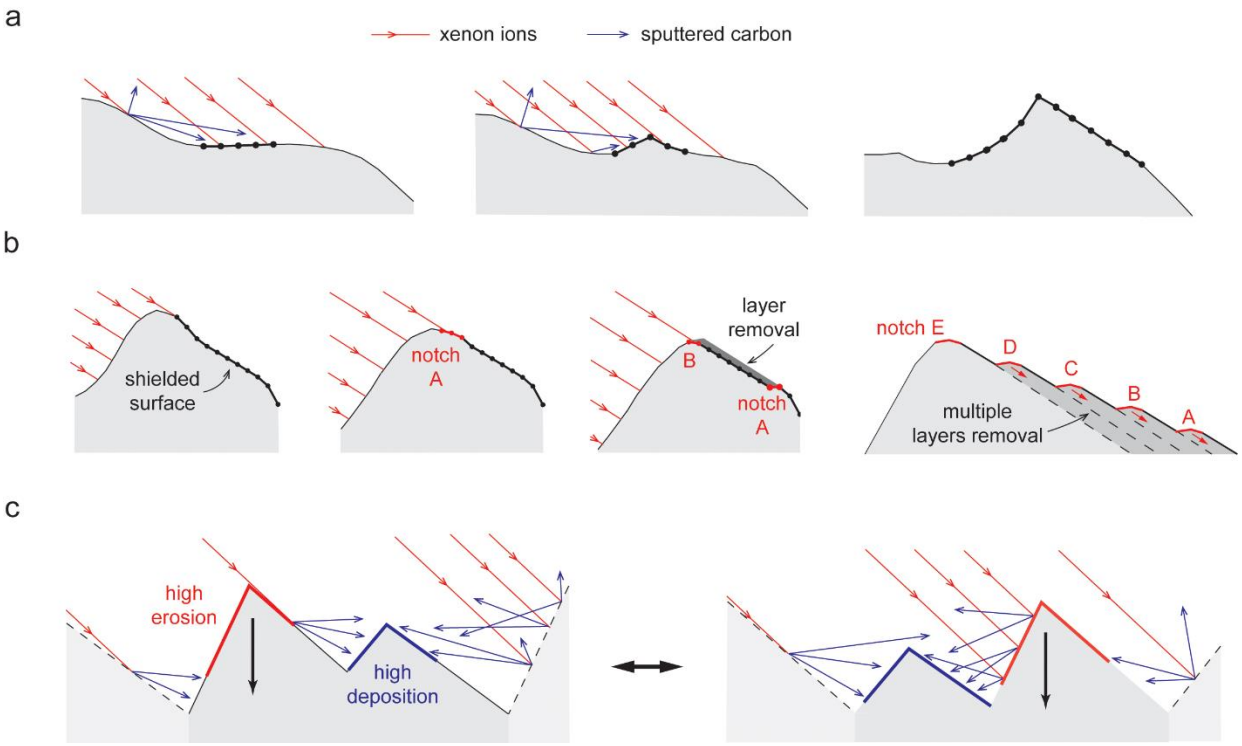

Figure 4.5: Overview of the multiscale sputter-erosion mechanisms of an undulating surface under an oblique ion fluence, caused by competing sputter erosion (red) and sputter deposition (blue) processes. **(a)** Nucleation of elemental undulations to form microscale undulations spanning multiple surface elements. **(b)** Erosion of microscale undulations to form surface steps, with the ion- and back-facing surfaces oriented perpendicular and parallel to the beam direction. Concentration of sputtering at the notch-tips resulting in simultaneous layer-by-layer erosion on the back-faces. **(c)** Concomitant decay of neighboring surface steps, resulting from the alternating and fluctuating deposition versus sputtering rates among these surface steps.

The competing sputtering and sputter-deposition mechanisms, resulting in roughening and smoothening and the eventual development of distinctive morphology changes at the microscale are observed in our Monte Carlo simulations for $\theta_{Xe} = 60°$ in Fig. 4.6a. Because the maximum sputtering yield is attained at $\bar{\theta}_{Xe} = \sim 75°$, sputter erosion is concentrated at the crest of the sinusoidal morphology (Fig. 4.6a-1), leading to flattening of the peak (Fig. 4.6a-2). In the process, a small undulation now develops at 'A' in Fig. 4.6a-2, with high and low sputtering rates on the ion- and back-facing sides of this undulation, but this peak is eventually flattened with continued bombardment (Fig. 4.6a-3). However, three other new undulations develop at 'B', 'C', and 'D' in Fig. 4.6a-3, and eventually grow into distinctive microscale undulations (Fig. 4.6a-4), which span over half of the simulation domain.

All these microscale undulations have a common characteristic: the ion- and back-facing sides are normal and parallel to the incidence ion flux, respectively, to resemble surface steps (see Fig. 4.6a-5). The flattening of the ion-facing surface to attain $\bar{\theta}_{Xe} = \sim 0°$ is akin to the morphology transitions from a rough to uniformly flat surface under a normal ion incidence (Section 4.2.1), and the ion-facing surface remains flat barring significant sputterant deposition. On the back-faces, any elemental undulations are rapidly eroded by the ion flux to create a flat, parallel surface that is shielded from the incidence ions by the tip of the surface steps, as illustrated in Fig. 4.5b. However, the high curvature at the tip facilitates rapid erosion to form a notch in the back-face, which propagates downhill with subsequent ion erosion. In turn, the nucleation and propagation of new notches lead to simultaneous removal of multiple layers of material on the back-face of each surface step. The erosion fronts (notches) of these material layers are characterized by intense but localized sputtering, as shown by the multiple dark red spots in the surface morphology contours (top), accompanied by rapid fluctuations in the sputtering signatures (bottom) on the

back-faces of surface steps ‘B’, ‘C’, ‘D’, and ‘E’ in Fig. 4.6a-4 to 4.6a-10. Similar micro-scale surface step features have been observed in the sputtering of exposed carbon fiber in carbon-carbon composites under the same xenon ion bombardment conditions of $E_{Xe} = 500$ eV, $\theta_{Xe} = 60°$, as shown by SEM images in Fig. 4.6b [7].

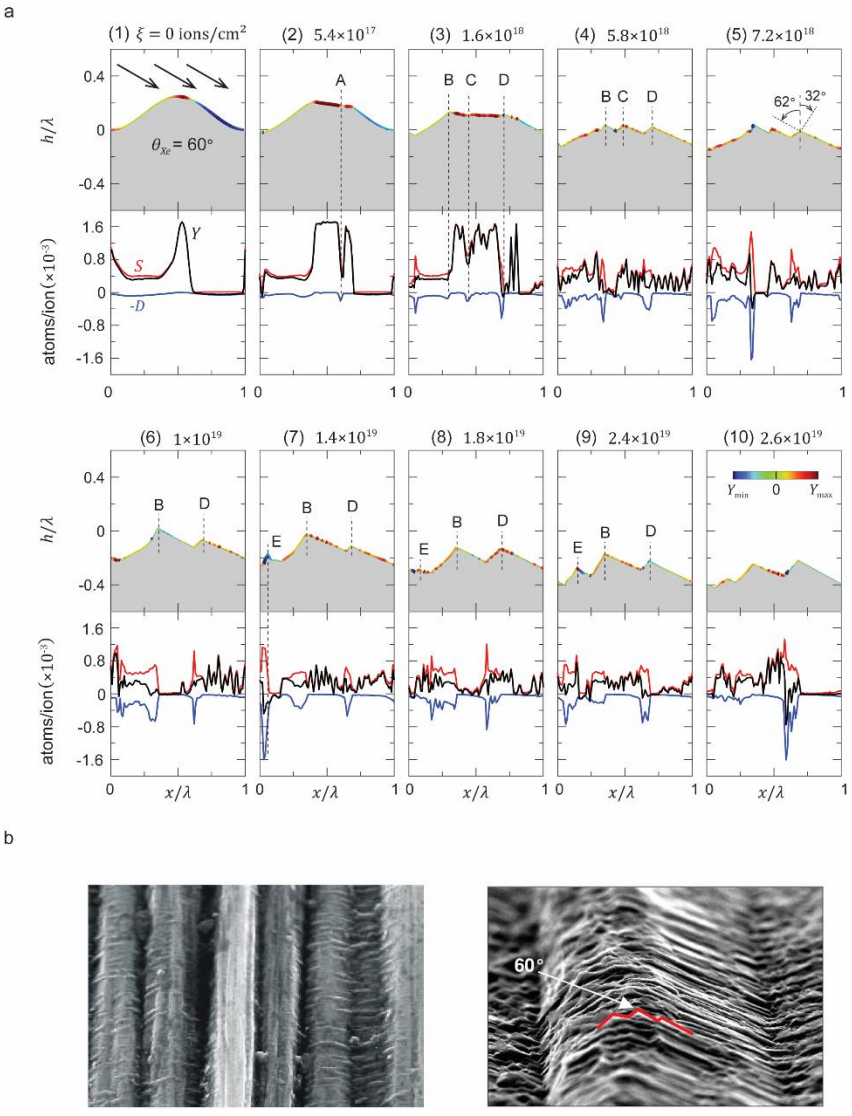

Figure 4.6: (a) Sputtering of an initial sinusoidal surface with $\frac{H}{\lambda} = 0.25$ under an ion fluence with $E_{Xe} = 500$ eV, $\theta_{Xe} = 60°$. Top: morphology evolution with ion fluence ξ; surface contours delineate elements with high yields of deposited atoms ($Y_{\min}$) and sputterants ($Y_{\max}$), respectively. Bottom: spatial distributions of the elemental yields of sputterants S and deposited atoms D, and elemental sputtering yields $Y = S - D$. **(b)** SEM imaging of an exposed carbon fiber before and after four hours of sputtering with $E_{Xe} = 500$ eV, $\theta_{Xe} = 60°$, depicting the formation of surface steps [7].

We remark that sputter erosion of these microscale surface steps does not occur in isolation. As illustrated in Fig. 4.5c, when a surface step is neighboring to, or sandwiched between, higher surface step(s), the deposition of sputterants from the neighboring surface step(s) could offset its

own sputter erosion and forestall its decay. Once the neighboring surface step(s) is now eroded to below a critical height, the reverse occurs where the decay of the neighboring surface step(s) is now forestalled at the expense of the now-taller surface step. This process allows for the concomitant decay of two or more surface steps and is the mechanism underpinning the alternate and fluctuating decay rates among the surface steps 'B', 'D', and 'E' in Fig. 4.5a-7 to 4.5a-10.

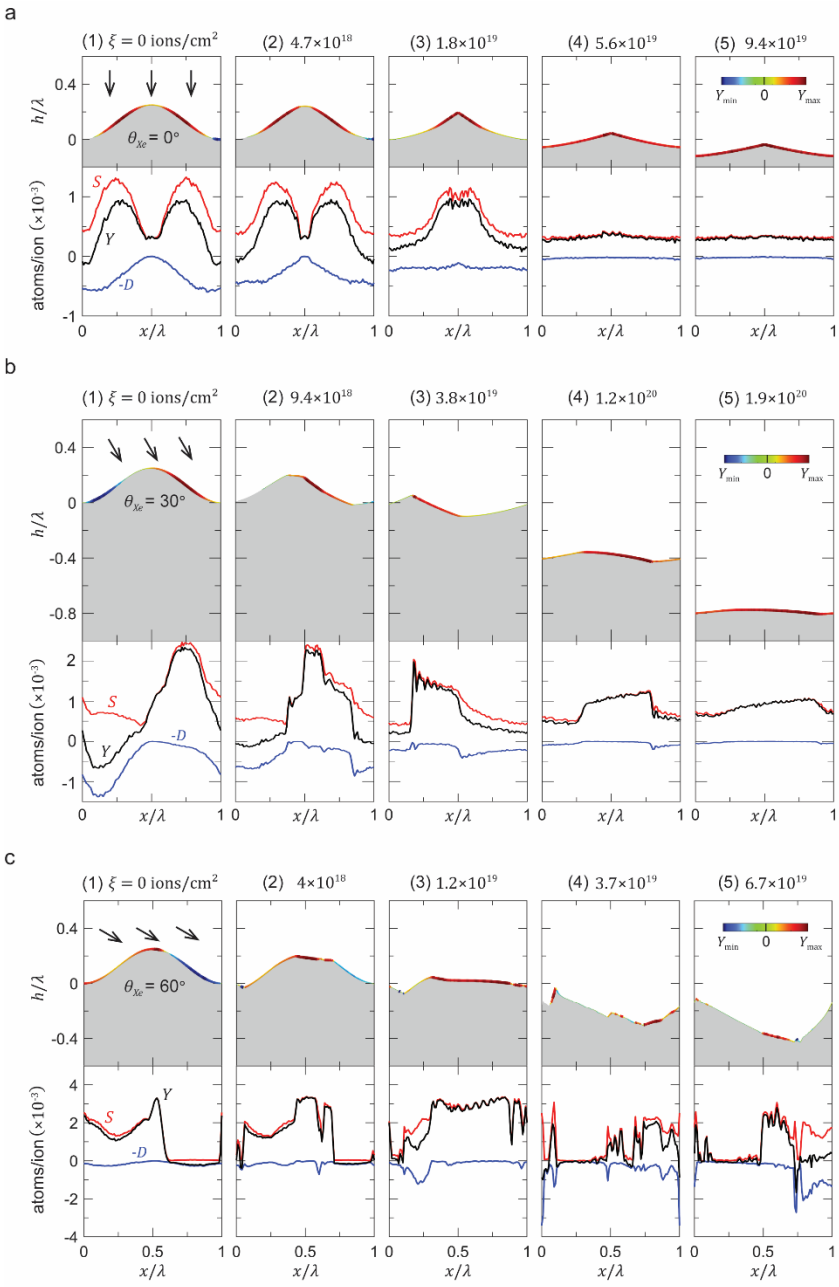

Figure 4.7: Sputtering of an initial sinusoidal surface under an ion fluence with $E_{Xe} = 200$ eV, **(a-c)** $\theta_{Xe} = 0°, 30°, 60°$. Top: morphology evolution with ion fluence ξ; surface contours delineate elements with high yields of deposited atoms ($Y_{\min}$) and sputterants ($Y_{\max}$), respectively. Bottom:spatial distributions of the elemental yields of sputterants S and deposited atoms D, and elemental sputtering yields $Y = S - D$.

We have also repeated our simulations under different ion energies ($E_{Xe} = 200, 1000$ eV) and incidence angles ($\theta_{Xe} = 0°, 30°, 60°$) and have observed similar morphological transitions (Figs. 4.7 and 4.8).

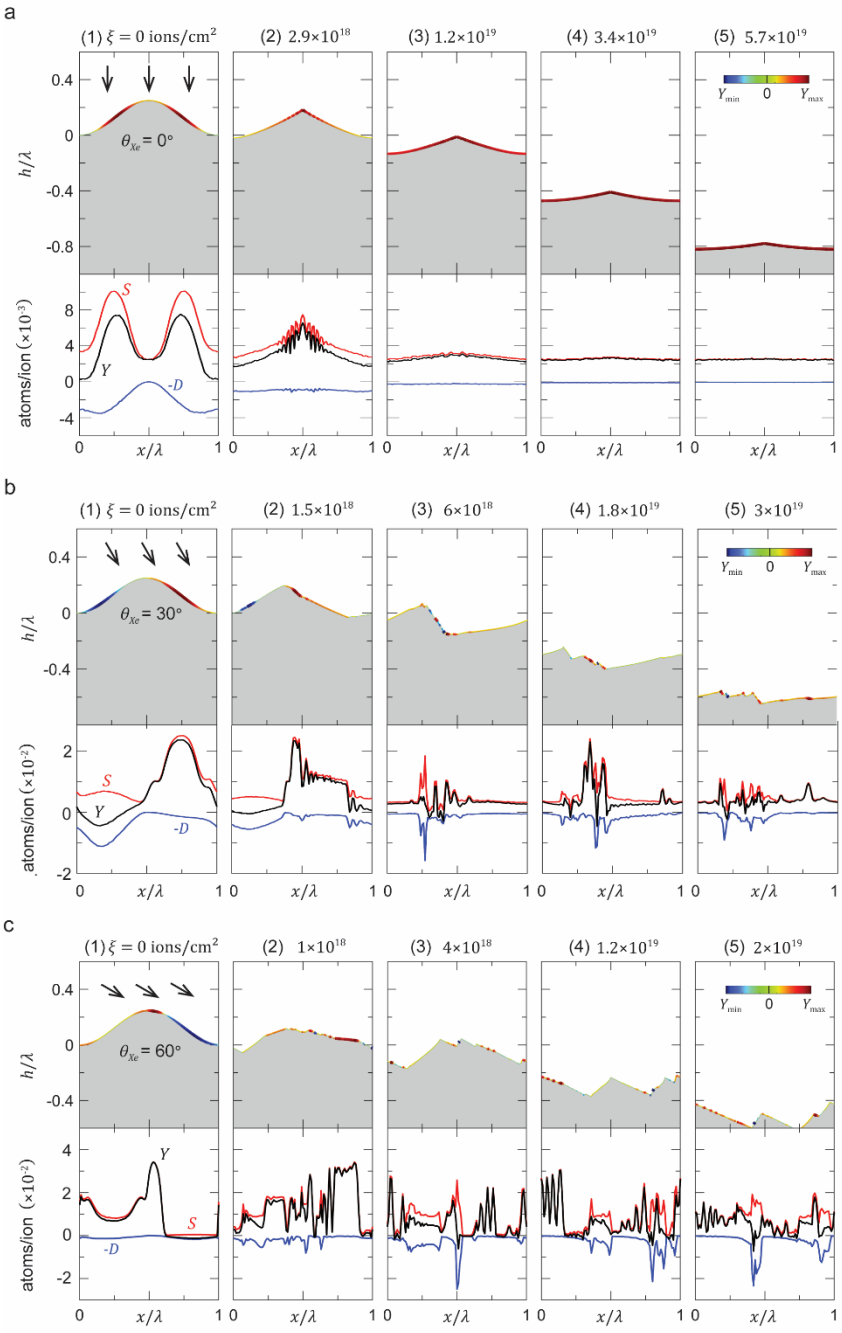

Figure 4.8: Sputtering of an initial sinusoidal surface under an ion fluence with $E_{Xe} = 1$ keV, **(a-c)** $\theta_{Xe} = 0°, 30°, 60°$. Top: morphology evolution with ion fluence ξ; surface contours delineate elements with high yields of deposited atoms ($Y_{\min}$) and sputterants ($Y_{\max}$), respectively. Bottom:spatial distributions of the elemental yields of sputterants S and deposited atoms D, and elemental sputtering yields $Y = S - D$.

4.3 Transient to Steady-state Sputtering Yield

The morphology evolution under xenon ion fluence (ξ) is accompanied by changes to the global sputtering yield ($\hat{Y}$), as shown in Fig. 4.9 for $E_{Xe} = 500$ eV with $\theta_{Xe} = 0°, 30°, 60°$. Under a normal ion fluence ($\theta_{Xe} = 0°$), the sputter yield monotonically decreases with ξ following the transition from a sinusoidal morphology to a sharp triangular morphology, and the subsequent self-similar erosion of this triangular protrusion. Beyond a fluence of ξ_{ss}, steady-state sputtering yield is reached as a flat surface morphology is maintained. Under oblique incidence angles, highly oscillating sputtering yields, particularly for $\theta_{Xe} = 60°$, are observed during the transient phase due to competing and evolving smoothening and roughening mechanisms at both the elemental- and micro-scales, resulting in the formation of surface steps (Section 4.2.2). Even beyond the steady-state fluence ξ_{ss} (operationally defined as a regime where $\Delta\hat{Y} = \hat{Y}_{\max} - \hat{Y}_{\min} \leq 10\%$), the concomitant interaction resulting in the alternating decay of neighboring surface steps at the mesoscale, leads to a fluctuating steady-state sputtering yield.

We summarize the above transitions and fluctuations in the sputtering yield with a box and whisker plot (inset in Fig. 4.9). Here, we use the whisker limits (a,e) to represent the maximum and minimum $\hat{Y}$ across the entire range of ξ, while the three box markers (b,c,d) are taken to represent the maximum, mean, and minimum values of $\hat{Y}$ beyond ξ_{ss}. Figure 4.10 compares our Monte Carlo sputtering yield predictions (blue box and whisker plots) across E_{Xe} of 200 to 1500 eV with $\theta_{Xe} = 0°, 30°, 60°$, against prior MD simulations [1] (red symbols) and existing sputtering yield measurements for various allotropes of carbon (black symbols) [2,7,8,11–15]. Under a normal ion incidence, the consistently smooth transition of $\hat{Y}$ to its steady-state value infers that (b,c,d,e) in each box and whisker plots collapse to a single point representing the lower limit of $\hat{Y}$. Across all ion energies, this lower limit coincides with the sputtering yield predictions from MD

since we ultimately attain a uniformly flat surface at steady-state, while the upper whisker limits represent the initial sputtering yields of the assumed sinusoidal morphologies with $\frac{H}{\lambda} = 0.25$. The box and whisker representations become more distinctive under an oblique ion fluence because of the increased scatter in $\hat{Y}$. The steady-state sputtering yield predictions now differ significantly from MD predictions because of the formation of *characteristic surface steps. In particular, MD predictions are now outside the whisker limits of the Monte Carlo simulations for* $\theta_{Xe} = 60°$ *because of this increased surface roughness.*

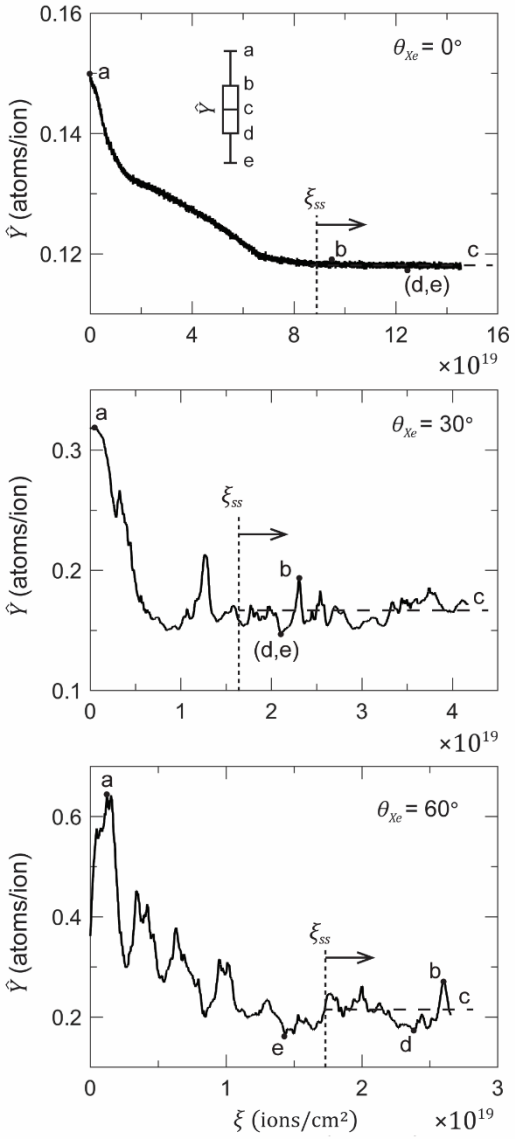

Figure 4.9: Evolution of the sputtering yield ($\hat{Y}$) with ion fluence (ξ) under $E_{Xe} = 500$ eV, $\theta_{Xe} = 0°, 30°, 60°$. Inset: box and whisker plot summarizing fluctuations in $\hat{Y}$ within the transient ($\xi < \xi_{ss}$) and steady-state ($\xi > \xi_{ss}$) regimes; whisker limits (a,e): maximum and minimum $\hat{Y}$ across the entire ξ; box markers (b,c,d): maximum, mean, and minimum $\hat{Y}$ within $\xi > \xi_{ss}$.

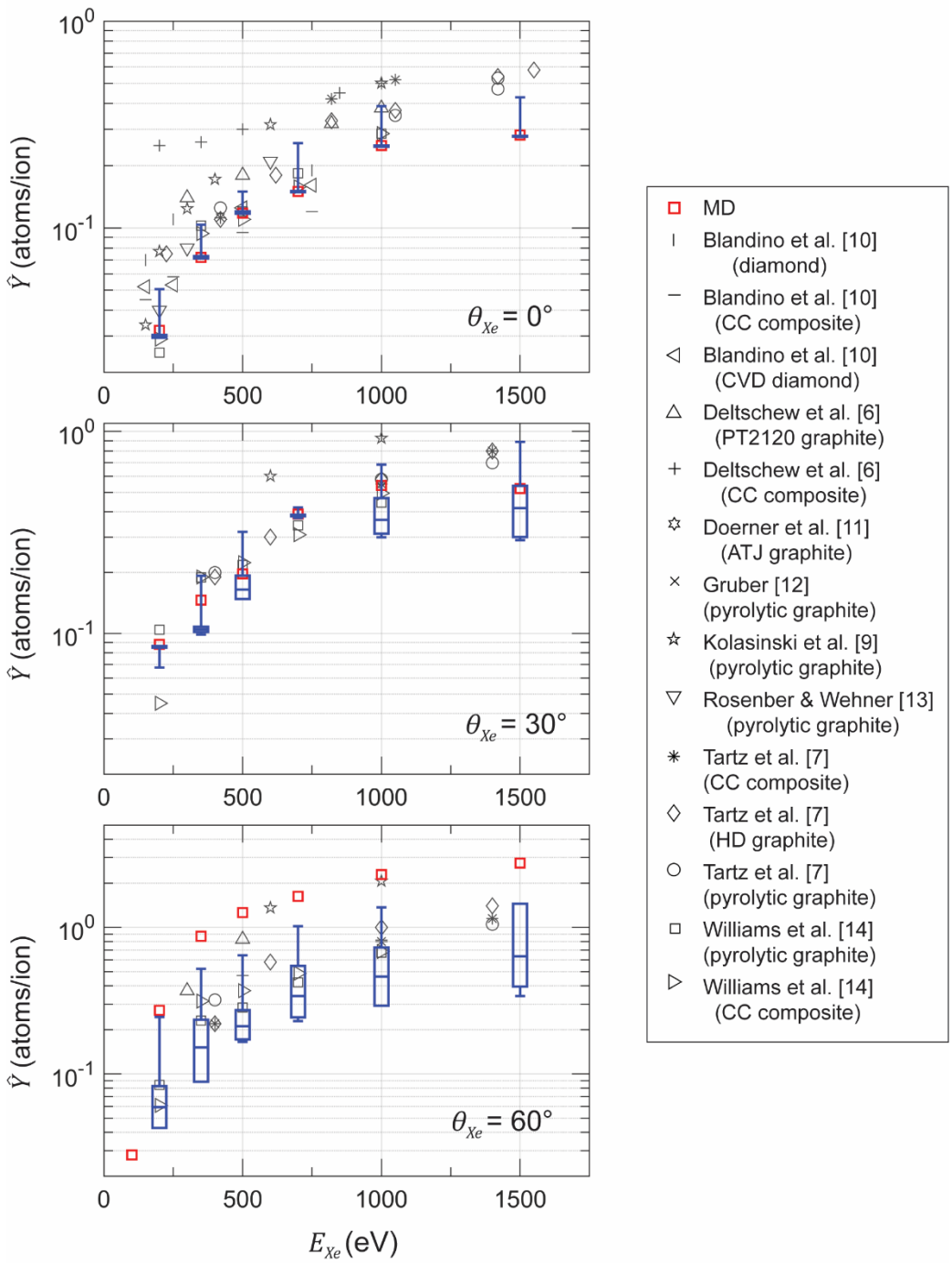

Figure 4.10: Comparison of sputtering yield ($\hat{Y}$) predictions from Monte Carlo simulations (blue box and whisker plots) versus MD simulations (red square) and sputtering experiments (black symbols), across various E_{Xe} and θ_{Xe}.

The sputtering yield is highly dependent on the evolving surface morphology within the transient regime, and an accurate and consistent measurement of the sputtering yield can only be

established within the steady-state regime. This presents a challenge to experimental measurements of the sputtering yield, since one cannot ascertain the value of ξ_{ss} a-priori. The upper and lower whisker limits from our Monte Carlo simulations in Fig. 4.10 define the bounds of the expected sputtering yield (albeit within the assumption of an initial sinusoidal morphology), with the steady-state values (box markers) constituting the lower end of the experimental measurements [2,7,8,11–15]. Our computations show that ξ_{ss} is at least several-folds higher for $\theta_{Xe} = 0°$ than for $\theta_{Xe} = 30°, 60°$ (Fig. 4.11), which could explain the closer proximity of the experimental measurements to the upper whisker limits for the former. Nevertheless, most of the experimental scatter across the range of E_{Xe} and θ_{Xe} are well-contained within the whisker limits. Notably, the box plots depicting sputter yield variations at steady-state encompass nearly all of the experimental data by Williams et al. [15], which utilizes a more accurate method of quartz crystal microbalance (QCM) [16–18] to measure the steady-state sputtering yield versus mass loss measurements.

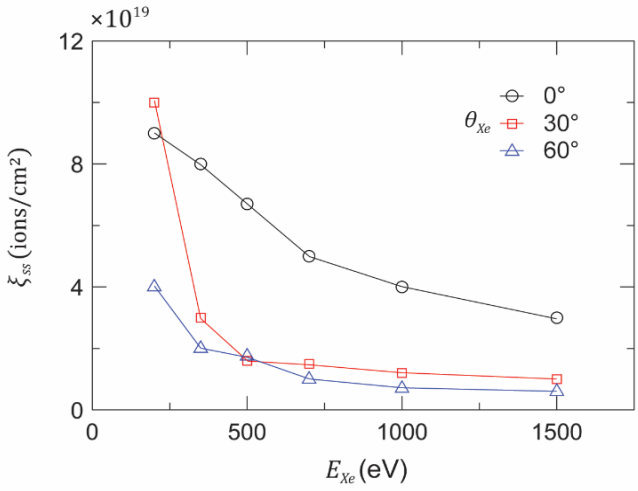

Figure 4.11: Critical fluence ξ_{ss} to attain steady-state sputtering yield versus ion energy E_{Xe} for $\theta_{Xe} = 0°, 30°, 60°$.

To elucidate the sensitivity of the differential sputter yield profile to the evolving surface morphology with ion fluence, we show in Fig. 4.12 the probability density function (PDF) of the angular distributions of carbon sputterants, $f(\theta_C)$, for $E_{Xe} = 500$ eV and $\theta_{Xe} = 0°, 60°$, at various

ξ with '(x)' corresponding to the respective topographies in Figs. 4.4- and 4.6-x. No significant change to the differential profile is observed under a normal ion incidence angle. Under an oblique ion incidence of $\theta_{Xe} = 60°$, however, the differential yield distinctly favors the front-scatter direction (50° to 60°) for the initial sinusoidal morphology, but increasing amount of backward scatter is observed with the development of surface steps, and $f(\theta_C)$ transitions closer to a cosine angle distribution (dashed curve) [27] at steady-state. In fact, the differential yield alternates between the blue and purple profiles corresponding to Fig. 4.6-9 and 4.6-10 at steady-state due to the alternating decay of neighboring surface steps. Similar relationships between the evolving surface morphology and differential yields are observed at $E_{Xe} = 200$ eV and 1 keV (Figs. 4.13 and 4.14 of the Supplementary Materials).

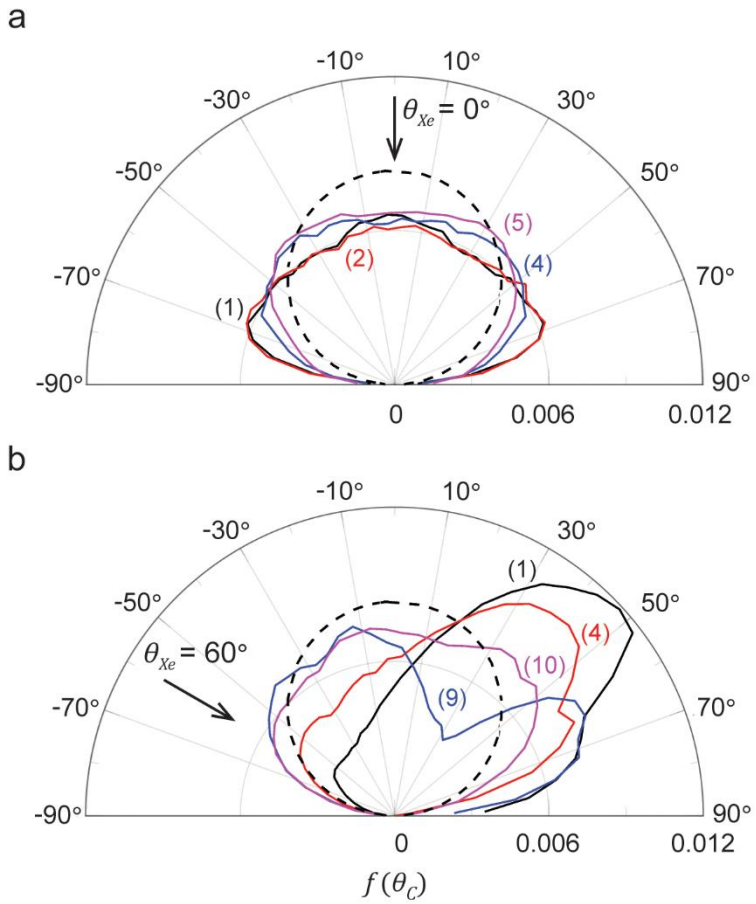

Figure 4.12: Probability density function (PDF) of the differential yield profiles of carbon sputterants for $E_{Xe} = 500$ eV, $\theta_{Xe} = 0°, 60°$. Each (x) in (**a**) and (**b**) corresponds to the respective surface morphologies at various ξ in Figs. 4.4-x and 4.6-x, respectively. Dashed curves denote the PDF of the cosine angle distribution [9].

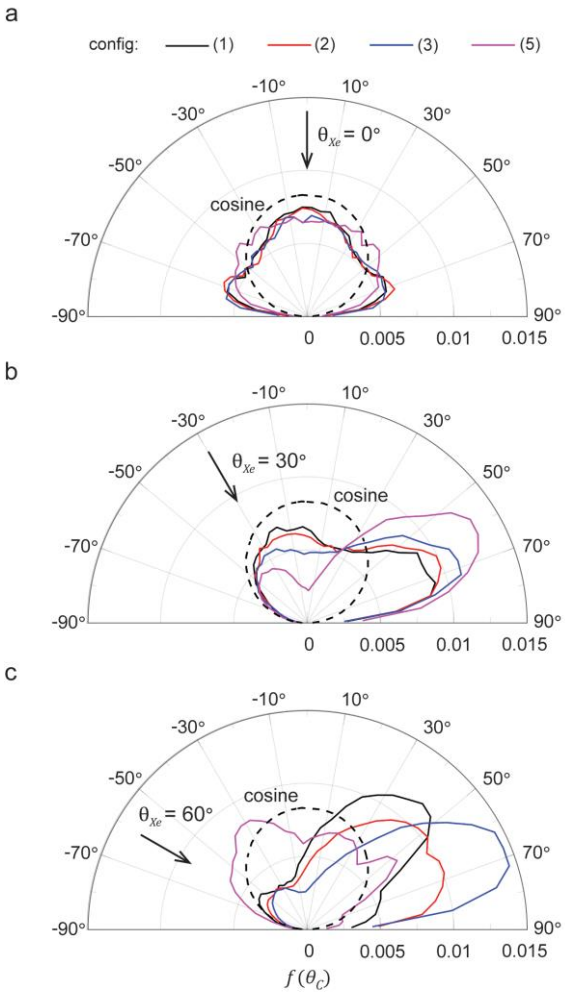

Figure 4.13: Probability density function (PDF) of the differential yield profiles of the carbon sputterants for $E_{Xe} = 200$ eV, $\theta_{Xe} = 0°, 30°, 60°$. Each (x) corresponds to the respective surface morphologies at various fluence ξ in Fig. 4.7. Dashed curves denote the PDF of the cosine angle distribution [9].

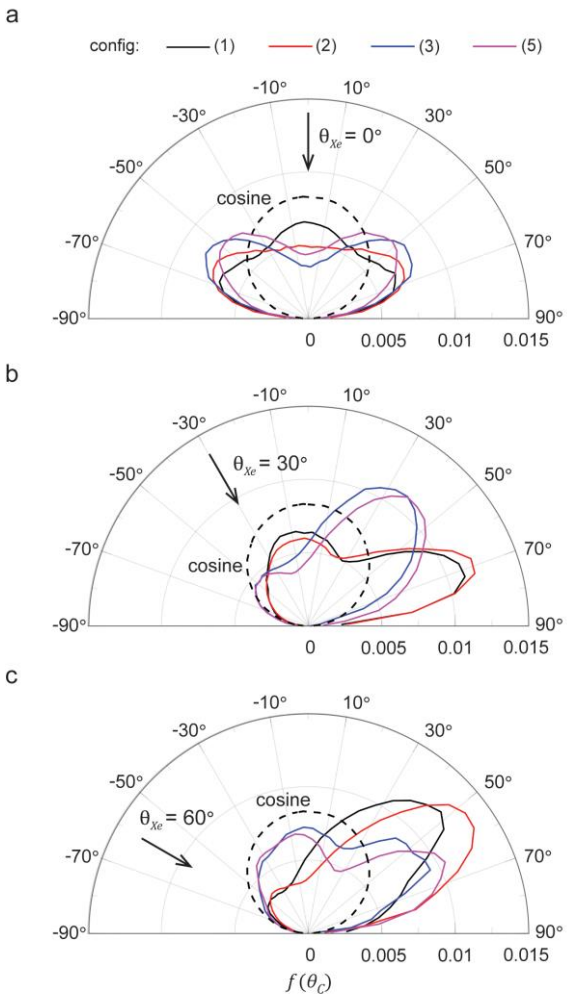

Figure 4.14: Probability density function (PDF) of the differential yield profiles of the carbon sputterants for $E_{Xe} = 1$ keV, $\theta_{Xe} = 0°, 30°, 60°$. Each (x) corresponds to the respective surface morphologies at various fluence ξ in Fig. 4.8. Dashed curves denote the PDF of the cosine angle distribution [9].

4.4 Sputtering-by-design

We expand our studies to include the effects of varying initial surface morphologies to gain insights into design strategies for controlling the sputtering yield. Our earlier analyses are based on an initial sinusoidal morphology with $\frac{H}{\lambda} = 0.25$. Figure 4.15 examines the effects of $\frac{H}{\lambda}$ on the

morphology evolution and the sputtering yield for $E_{Xe} = 500$ eV with $\theta_{Xe} = 0°, 30°, 60°$. When the surface is sufficiently rough ($\frac{H}{\lambda} \geq 0.2$), the surface morphologies at steady-state (purple, blue, and red curves in Fig. 4.15a) are virtually independent of the initial topology (black curves in Fig. 4.15a): we obtain a uniformly flat surface at $\theta_{Xe} = 0°$, and characteristic surface steps at $\theta_{Xe} = 30°$ and $60°$. For a sufficiently smooth initial surface ($\frac{H}{\lambda} = 0.1$), however, elemental undulations cannot develop if the sputtering yield is nearly spatially homogeneous under $\theta_{Xe} = 30°$. Meanwhile, multiple elemental undulations can still develop under a high ion incidence angle of $\theta_{Xe} = 60°$, giving rise to the formation of multiple, albeit smaller-dimension, surface steps at steady-state. The development of these multiple surface steps is responsible for the significant variations in the sputtering yield during the transition to reach steady-state (whisker height). As shown in Fig. 4.15b, increasing $\frac{H}{\lambda}$ dramatically reduces these variations. The opposite trend is

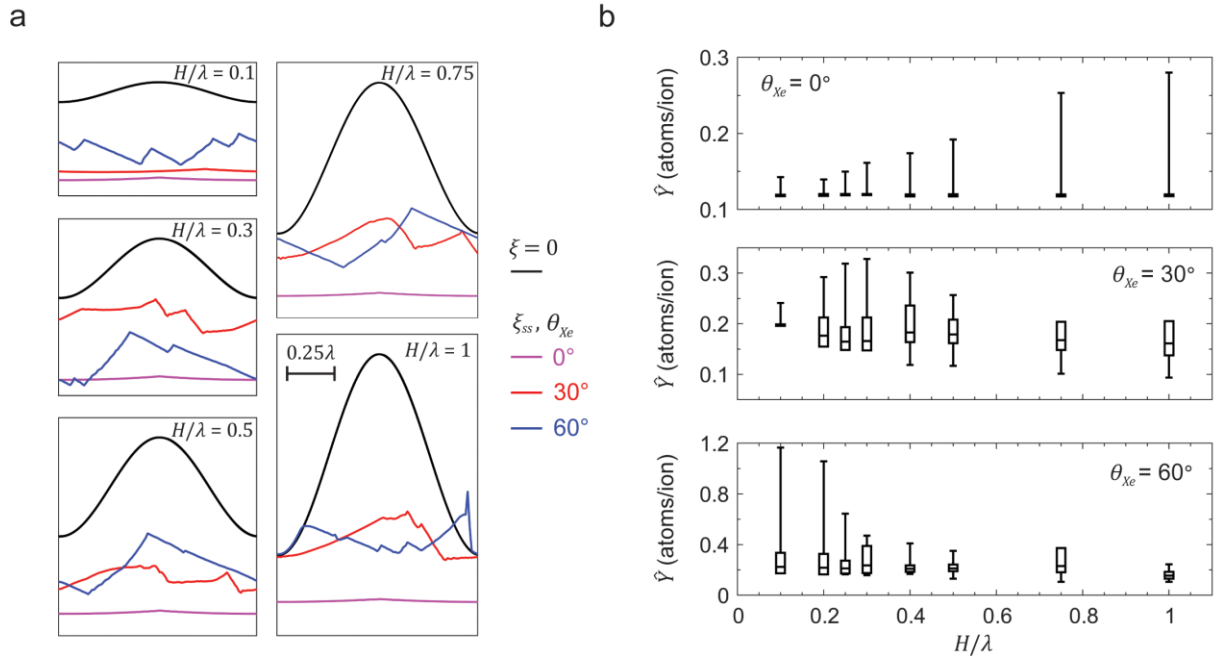

Figure 4.15: Effects of initial surface roughness $\frac{H}{\lambda}$ under $E_{Xe} = 500$ eV, $\theta_{Xe} = 0°, 30°, 60°$. (**a**) Surface morphology evolutions at steady-state sputtering (colored curves), $\xi = \xi_{ss}$, from the initial sinusoidal morphologies at $\xi = 0$ (black curves). (**b**) Box and whisker summary of the sputtering yields ($\hat{Y}$).

observed for $\theta_{Xe} = 0°$ because more material has to be eroded for larger $\frac{H}{\lambda}$ to attain a morphologically flat surface. Nevertheless, with the exception for $\frac{H}{\lambda} = 0.1, \theta_{Xe} = 30°$, the steady-state sputtering yield (box plots) is virtually independent of $\frac{H}{\lambda}$ since morphologically similar structures are obtained.

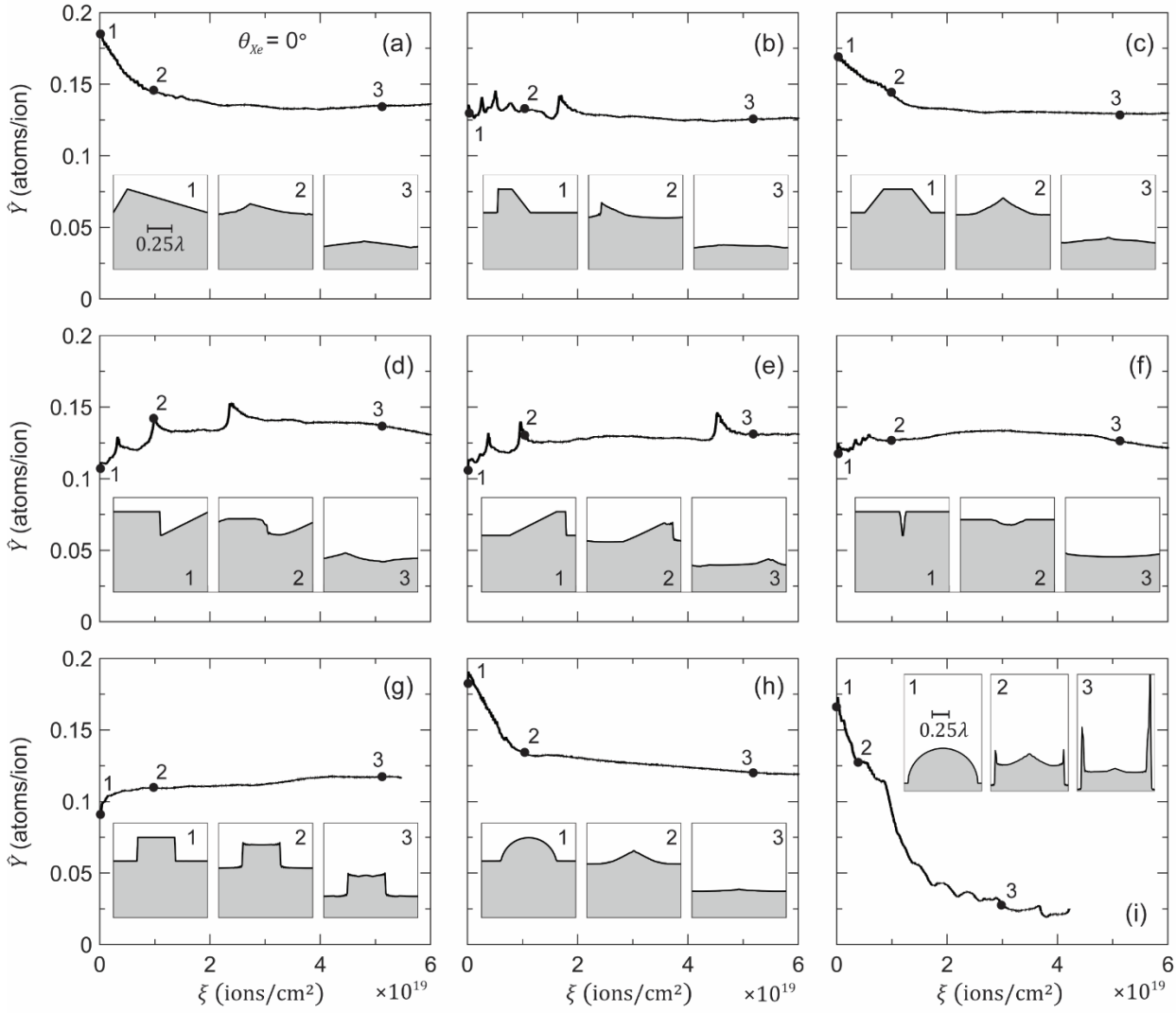

Figure 4.16: Transient to steady-state evolution of the sputtering yield ($\hat{Y}$) under $E_{Xe} = 500$ eV, $\theta_{Xe} = 0°$ for different initial surface morphologies, with snapshots of the evolving morphologies.

We next examine the surface morphology and sputtering yield evolutions for a large variety of initial surface structures, beyond the idealized sinusoidal morphologies above. Under a normal ion incidence in Fig. 4.16, all but two of these structures evolve towards a morphologically flat surface and attain the same steady-state sputtering yield (Fig. 4.16a-f and Fig. 4.16h). The transient response of these structures, however, are sensitive to the initial morphology and can have transient sputtering yields that are higher or lower than the steady-state values. The sputtering yield of a surface represented by a periodic step function (Fig. 4.16g) is one interesting exception: both the morphology and the sputtering yield are almost unchanged across the range of ion fluence because the majority of the ions are already impacting the surface in the normal direction ($\theta_{Xe} \approx \bar{\theta}_{Xe} \approx 0°$) from the start. Another exception is the semi-circular morphology (Fig. 4.16i), where the close periodicity leads to significant redeposition of the sputterants and drives the growth of thin vertical walls. These walls in turn are able to trap even more sputterants, significantly reducing the sputtering yield. Such vertical thin wall morphologies have been observed in the post-sputtered scanning microscopy images of exposed semi-circular carbon fibers subjected to similar ion conditions of $E_{Xe} = 500$ eV and $\theta_{Xe} = 0°$ [7], as well as $E_{Xe} = 1$ keV and $\theta_{Xe} = 0°$ [8]. Our simulations show that such wall structures cannot form when the periodic semi-circular morphologies are sufficiently far apart to reduce the contributions of sputter deposition (Fig. 9h).

Under an oblique ion incidence angle ($\theta_{Xe} = 60°$), all the surfaces evolve to form surface steps at steady-state, with similar steady-state sputtering yields of 0.2 ± 0.05 atoms/ion (Fig. 4.17). The details of the initial morphology, however, are observed to control the eventual size-scales of these surface steps at steady-state. Interestingly, steep wall initial configurations in Fig. 4.17d,e,g results in the development of unique tapering thin wall structures (configuration 2) which momentarily reduces the sputtering yield to 0.1 atoms/ion because of increased redeposition of the

sputterants; the eventual erosion of these wall structures increases the sputtering yield back to its steady-state value.

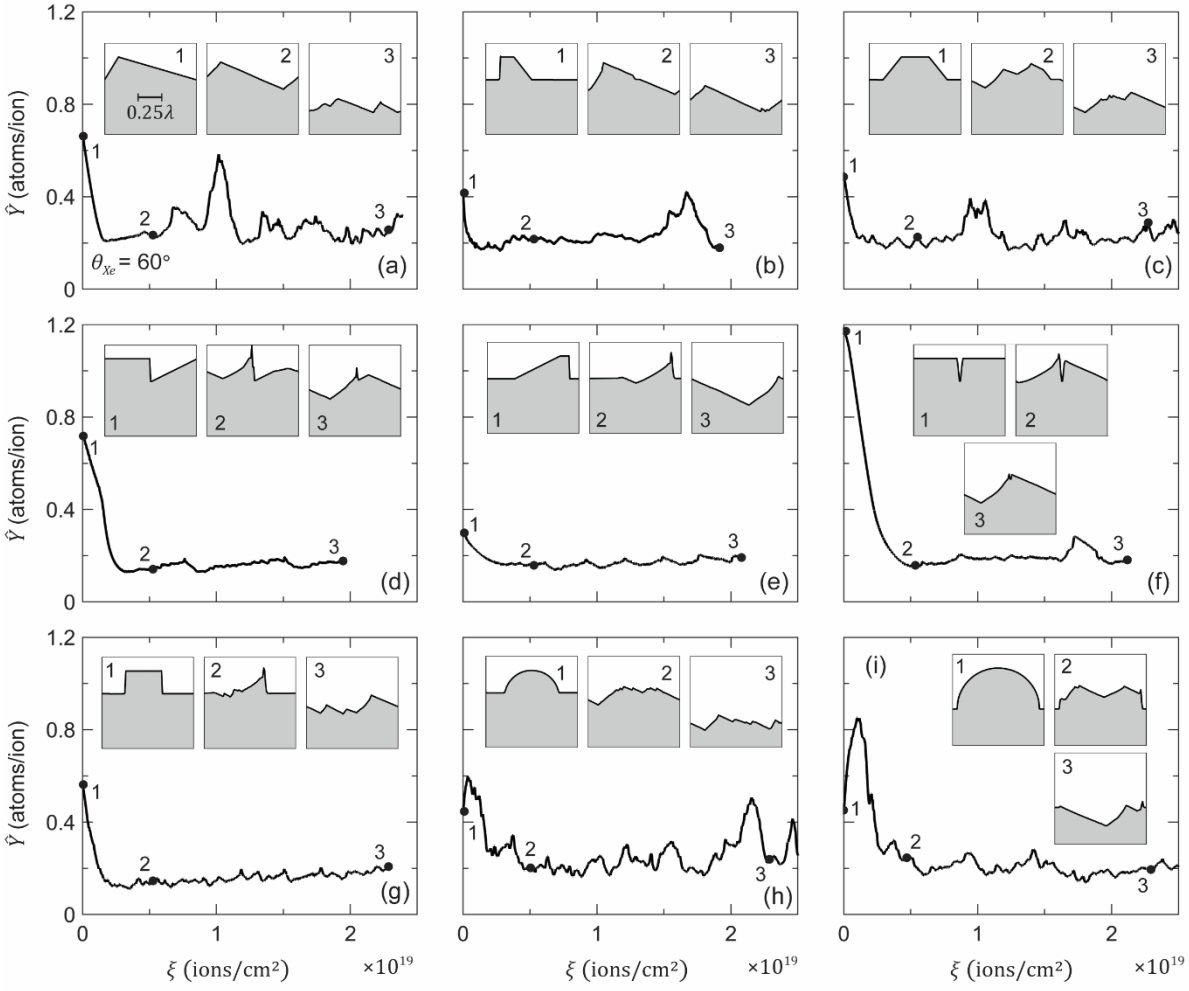

Figure 4.17: Transient to steady-state evolution of the sputtering yield ($\hat{Y}$) under $E_{Xe} = 500$ eV, $\theta_{Xe} = 60°$ for different initial surface morphologies, with snapshots of the evolving morphologies.

Motivated by the above observations, we test the efficacy of two possible thin wall configurations, comprising of periodic near-vertical walls, with a thickness to height aspect ratio of 1:5, in mitigating sputtering under $E_{Xe} = 500$ eV and $\theta_{Xe} = 0°, 60°$ (Fig. 4.18). Under $\theta_{Xe} = 0°$, the steady-state sputtering yield of this wall configuration (solid line in Fig. 4.18a) is remarkably reduced by nearly 40% compared to an initial sinusoidal morphology (black dash line)

and is consistently lower than all of the prior experimental sputtering yield measurements ($\hat{Y}$ values denoted by arrows). More importantly, the wall morphology is not significantly degraded by the sputtering process (Fig. 4.18a-1 to 4.18a-3) even at a very high fluence. Under $\theta_{Xe} = 60°$, we have reduced the interval between each vertical wall to allow the ions to only impact the sacrificial wall structures, while allowing for significant re-deposition of the sputtered species (Fig. 4.18b). The bombardment process morphs the top-exposed surface of the vertical walls into sharp vertical peaks (akin to Fig. 4.17d,e,g) and dramatically reduces the sputtering yield to 0.1 atoms/ion. Surprisingly, this low sputtering yield can be sustained for a considerable amount of fluence, until the wall is completely degraded, giving way to formation of the usual characteristic surface steps at steady-state. We remark that this transient period of low sputtering yield can be extended with taller wall structures, since they act as sacrificial elements to delay the formation of surface steps.

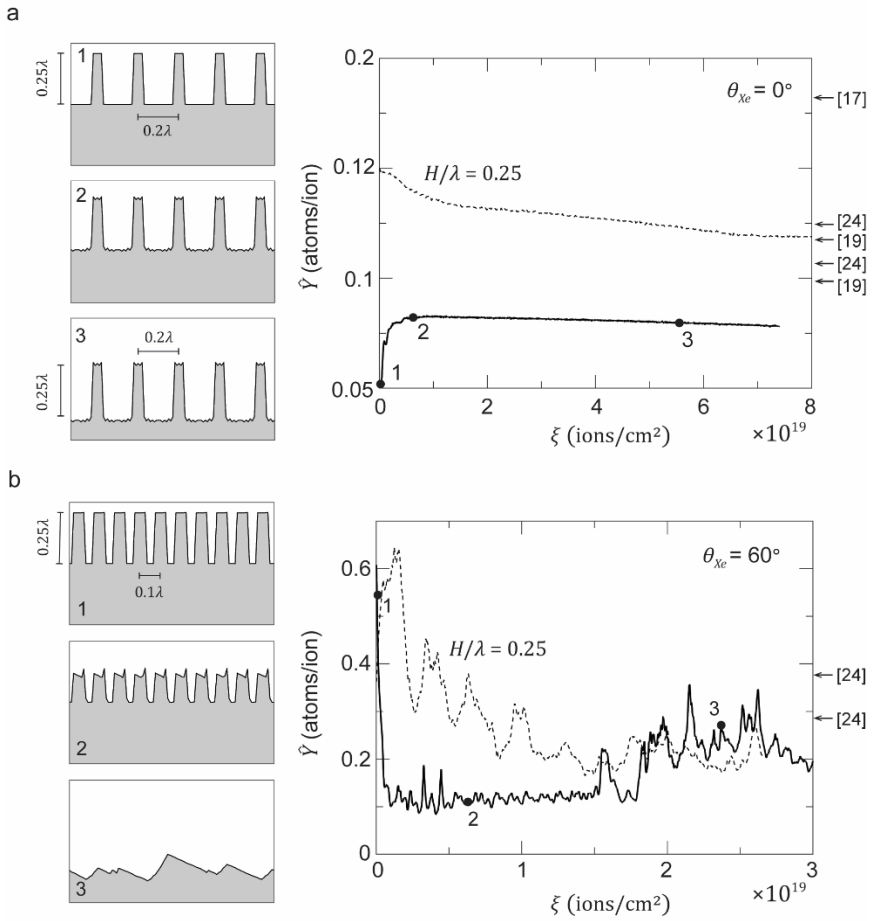

Figure 4.18: Periodic, near-vertical, wall structures to forestall sputtering under $E_{Xe} = 500$ eV with $\theta_{Xe} = 0°$ (**a**) and $\theta_{Xe} = 60°$ (**b**). Left: snapshots of the evolving surface morphologies. Right: evolution of the sputtering yield ($\hat{Y}$) for the proposed wall structures (solid curves) versus an initial sinusoidal morphology with $\frac{H}{\lambda} = 0.25$ (dashed curves); black arrows: experimental sputtering yield measurements from literature.

4.5 Discussions and Implications

Experiments have reported a large spread in the sputtering yield measurements of carbon under the bombardment of heavy ions, but could not establish a correlation between the sputtering yield and the various polymorphs and allotropes of carbon (e.g. graphite, diamond, carbon-carbon composites) [8,15]. MD results in Chapter 3 show that repeated bombardment of ions creates an amorphous carbon subsurface structure, that is largely independent of the ion energy and incidence angle, as well as the initial carbon structure. Thus, the sputtering yield at steady-state is expected to be independent of both the initial carbon structure as well as the prior sputtering history. From the standpoint of MD simulations, steady-state sputtering yield is attained once the initial carbon structure (graphite, diamond) has been transformed to this unique amorphous carbon subsurface structure. While computationally costly to simulate, this microstructural transformation necessitates an ion fluence that is several-orders-of-magnitude smaller compared to that required to evolve a surface to attain its steady-state morphology in our Monte Carlo simulations. Starting from a multilayer graphitic structure, for example, MD simulations require an ion fluence of $\sim 10^{15}$ ions/cm^2 to attain its amorphous subsurface structure [1], versus 10^{19} to 10^{20} ions/cm^2 (assuming $\lambda = 2$ μm) for the range of initial morphologies considered here to transition to the steady-state characteristic structures. Thus, this surface morphological transition is the limiting time-scale to achieve true steady-state sputtering yield.

Regardless of the initial surface morphology (with exception of specifically designed stepped structures), we show that a normal ion flux always erodes a morphologically rough surface to become uniformly flat to achieve steady-state sputtering yield. Similarly, a high oblique ion flux is also not very sensitive to the initial morphology and tends to create characteristic surface steps at steady-state through a multiscale process involving: (a) the nucleation of local surface

(elemental) undulations, which grow to form microscale surface steps, (b) layer-by-layer slicing of these surface steps, and (c) concomitant alternating decay of neighboring steps. At moderate ion incidence angles, the steady-state sputtering configurations can transition to either of these two surface structures (flat versus surface steps), depending on the roughness of the starting morphologies. Consequently, while one cannot expect the initial surface morphologies of different carbon structures used in various sputtering experiments to be even remotely similar, our studies here suggest that the steady-state yield is not very sensitive to the initial surface morphology.

Our Monte Carlo simulation results are in good agreement with prior sputtering experiments. Under a normal xenon ion flux with energy of 1 keV, SEM images show distinct smoothening of the morphologies across all carbon surfaces (including polished and unpolished pyrolytic graphite, graphite with different grain sizes, and carbon-carbon composites) after 3 hours of sputtering, regardless of the initial surface roughness (Fig. 4.4b) and [8]. Under an oblique ion flux ($\theta_{Xe} = 60°$), SEM imaging shows the initial cylindrical morphologies of exposed carbon fibers (resembling the initial morphologies in Fig. 4.16h,i) erode to form micro-scale surface steps, with characteristic features which are virtually identical to those observed in our simulations [7], i.e. ion- and back-facing sides of each step are perpendicular and parallel to the beam direction (Fig. 4.6b). Similar behaviors have been reported in the noble gas bombardment of other covalently-bond structures [19–21]. Our Monte Carlo model is only able to capture the evolution of micro-scale surface features since any nanoscale mechanisms/features (from MD) are homogenized at the elemental level. Nevertheless, the micro-scale characteristic surface steps under an oblique ion incidence at steady state closely resemble the nanoscale terraced surfaces (each step spanning 10 to 100 nm) observed experimentally under relatively high angles of ion incidence and at high ion

fluences [22,23]. This self-similarity suggests that similar mechanistic processes could govern the formation and evolution of these nanoscale features.

While the initial surface morphology does not control the steady-state sputtering yield, the transition time (fluence) to attain these steady-state structures is dependent on the initial surface roughness, as well as the ion incidence energy and angle. We show that the fluence to attain steady-state sputtering yield under a normal ion incidence can be several-folds higher than under an oblique ion incidence, since any undulations have to be flattened completely under the former. Attaining steady-state sputtering yield under a normal ion fluence is therefore challenging, especially when the ion energies are low. While the exact xenon ion fluence used in obtaining the sputtering yield measurements in experiments are not widely reported, typical sputtering experiments are conducted over a period of 1 to 4 hours, and in certain cases, over a maximum of 8 hours [7]. This corresponds to an estimated fluence of $\sim 10^{18}$ to 10^{19} ions/cm^2 for a typical ion source, which is far short of the estimated $\sim 10^{19}$ to 10^{20} ions/cm^2 required to attain the characteristic steady-state structures in our simulations. Another source of uncertainty is the fluctuations in the steady-state sputtering yield under an oblique ion incidence, arising from the alternating decay of neighboring surface steps. The extent of such fluctuations depends on the density of these surface steps. Thus, the reported sputtering yield, even at steady-state, should be bounded by an uncertainty. The error bars in our Monte Carlo predictions, which account for the transient to steady-state variations in surface morphologies, are found to encompass a large portion of the experimental scatter across the range of ion energies and incidence angles (Fig. 4.10).

Our simulations also show that the sputtering yield can be controlled to a certain extent by surface patterning. For example, periodic wall structures are found to reduce the sputtering yield under a normal ion flux by nearly two-fold, and the general morphology is maintained even under

extended ion fluences. Under an oblique ion fluence, similar wall structures are found to lower the sputtering yield in the transient regime by several-folds. However, by acting as sacrificial elements, these wall structures eventually degrade to form surface steps, recovering back the steady-state sputtering yield. Such structural designs can enhance the sputter resistance capabilities of plasma facing materials, including the optics in ion thrusters, pole covers of Hall effect thrusters, or as well as carbon wall panels of ground-based testing facilities for electric space propulsion [24–27]. Nevertheless, the nodes in our current one-dimensional Monte Carlo simulations only account for vertical changes in height and cannot model possible horizontal (sideway or out-of-plane) growth, which potentially could be important for accurately simulating sputterant deposition on tall vertical wall structures. Such two- or even three-dimensional (out-of-plane) Monte Carlo modeling is a subject of future work.

4.6 Conclusions

In summary, we have extended the length- and time-scales of MD simulations through a Monte Carlo model to elucidate the effects of the evolving surface morphology on the ion-surface sputtering mechanisms and resulting sputtering yield of carbon substrates. Regardless of the initial surface morphologies, our results show that the sputtered surfaces generally transition to a flat morphology under normal ion flux or form multiple surface steps under an oblique ion flux; the ensuing steady-state sputtering yield is insensitive to the initial surface morphology. We updated the fitting parameters for Eckstein & Preuss [28] sputter yield formula (Equation 3.4 – 3.9) and the fitting parameters for the sputterants angle distribution (Equation 3.1) described in Chapter 3 for the morphology steady-state in Table 4.3 and 4.4. Finally, the initial surface structures can be engineered to reduce the sputtering yield under extended ion fluences in the transient regime, applicable for use in plasma facing applications.

Table 4.3: Fitting parameters for the Eckstein & Preuss [28] sputter yield formula based on the steady state sputtering yield MC-simulation data (Fig. 4.10).

Parameters	Uncertainty bounds		
	Min	**Mean**	**Max**
Q (eV^{-1})	1.89	2.18	2.53
λ	3.82	4.05	2.83
μ	2.17	1.97	1.84
E_{th} (eV)	11.71	10.92	12.09
f	2.47	2.29	3.01
c	0.84	0.44	0.62
b	0.61	0.71	0.61

Table 4.4: Fitting parameters for two Gaussian distribution functions (equation (3.1)) used to describe the PDF of the sputtered carbon direction (α) as a function of E_{Xe} and θ_{Xe}.

$E_{Xe} = 200$ eV						$E_{Xe} = 350$ eV					
θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A	θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A
0°	44.42	29.72	-31.65	36.03	0.43	0°	32.88	36.12	-40.80	32.01	0.56
30°	35.64	30.89	-57.90	22.06	0.30	30°	21.03	37.89	-41.58	23.46	0.55
45°	24.04	34.59	-39.47	22.46	0.58	45°	39.36	28.84	-29.68	35.34	0.31
60°	16.28	42.40	-59.41	14.55	0.73	60°	2.64	46.48	-65.92	13.29	0.69
75°	25.35	40.22	-57.75	17.84	0.21	75°	22.61	39.22	-59.23	14.84	0.36
$E_{Xe} = 500$ eV						$E_{Xe} = 700$ eV					
θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A	θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A
0°	37.38	31.97	-36.23	31.87	0.48	0°	55.37	21.77	-55.28	22.38	0.49
30°	23.34	37.84	-46.79	23.51	0.53	30°	21.75	38.44	-54.86	24.22	0.43
45°	18.85	38.59	-46.48	23.40	0.52	45°	26.22	35.97	-37.49	28.39	0.49
60°	17.70	40.82	-62.81	16.76	0.70	60°	10.80	40.79	-54.75	18.65	0.65
75°	34.30	24.11	-48.25	38.92	0.35	75°	40.83	36.21	-51.82	19.60	0.19

Table 4.4 (cont.)

$E_{Xe} = 1000$ eV						$E_{Xe} = 1500$ eV					
θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A	θ_{Xe}	μ_1	σ_1	μ_2	σ_2	A
0°	43.73	27.07	-44.68	26.04	0.51	0°	45.19	30.51	-43.29	31.59	0.49
30°	39.66	29.80	-42.72	25.57	0.32	30°	34.17	31.14	-43.14	23.74	0.33
45°	35.37	32.59	-29.74	33.81	0.32	45°	20.60	32.89	-20.87	37.78	0.36
60°	11.82	38.61	-60.31	16.18	0.69	60°	10.28	40.55	-60.96	15.78	0.60
75°	36.27	29.81	-48.51	19.96	0.30	75°	22.63	36.77	-51.52	17.40	0.39

4.7 References

[1] H. Tran, H.B. Chew, Surface morphology and carbon structure effects on sputtering: Bridging scales between molecular dynamics simulations and experiments, Carbon 205 (2023) 180–193. https://doi.org/10.1016/j.carbon.2023.01.015.

[2] R. Kolasinski, J. Polk, D. Goebel, L. Johnson, Carbon Sputtering Yield Measurements at Grazing Incidence, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, Sacramento, California, 2006. https://doi.org/10.2514/6.2006-4337.

[3] O. Bobes, K. Zhang, H. Hofsäss, Ion beam induced surface patterns due to mass redistribution and curvature-dependent sputtering, Phys. Rev. B 86 (2012) 235414. https://doi.org/10.1103/PhysRevB.86.235414.

[4] H. Hofsäss, O. Bobes, Prediction of ion-induced nanopattern formation using Monte Carlo simulations and comparison to experiments, Applied Physics Reviews 6 (2019) 021307. https://doi.org/10.1063/1.5043188.

[5] C.C. Umbach, R.L. Headrick, K.-C. Chang, Spontaneous Nanoscale Corrugation of Ion-Eroded SiO 2 : The Role of Ion-Irradiation-Enhanced Viscous Flow, Phys. Rev. Lett. 87 (2001) 246104. https://doi.org/10.1103/PhysRevLett.87.246104.

[6] M. Moseler, P. Gumbsch, C. Casiraghi, A.C. Ferrari, J. Robertson, The Ultrasmoothness of Diamond-like Carbon Surfaces, Science 309 (2005) 1545–1548. https://doi.org/10.1126/science.1114577.

[7] R. Deltschew, M. Tartz, V. Plicht, E. Hartmann, H. Neumann, Sputter characteristics of carbon-carbon compound material, in: 27th International Electric Propulsion Conference, 2001.

[8] M. Tartz, Pyrolytic Graphite and Carbon-carbon Sputter Behaviour Under Xenon Ion Incidence, in: 29th International Electric Propulsion Conference, 2005: p. 10.

[9] P. Sigmund, Sputtering by ion bombardment theoretical concepts, in: R. Behrisch (Ed.), Sputtering by Particle Bombardment I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981: pp. 9–71. https://doi.org/10.1007/3540105212_7.

[10] M.W. Thompson, II. The energy spectrum of ejected atoms during the high energy sputtering of gold, The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics 18 (1968) 377–414. https://doi.org/10.1080/14786436808227358.

[11] J.J. Blandino, D.G. Goodwin, C.E. Garner, Low energy sputter yields for diamond, carbon–carbon composite, and molybdenum subject to xenon ion bombardment, Diamond and Related Materials 9 (2000) 1992–2001. https://doi.org/10.1016/S0925-9635(00)00350-2.
[12] R.P. Doerner, D.G. Whyte, D.M. Goebel, Sputtering yield measurements during low energy xenon plasma bombardment, Journal of Applied Physics 93 (2003) 5816–5823. https://doi.org/10.1063/1.1566474.
[13] J. Gruber, Low-Energy Sputter Erosion of Various Materials in a T 5, in: 27th International Electric Propulsion Conference, 2001.
[14] D. Rosenberg, G.K. Wehner, Sputtering Yields for Low Energy He^+-, Kr^+-, and Xe^+-Ion Bombardment, Journal of Applied Physics 33 (1962) 1842–1845. https://doi.org/10.1063/1.1728843.
[15] J. Williams, M. Johnson, D. Williams, Differential Sputtering Behavior of Pyrolytic Graphite and Carbon-Carbon Composite Under Xenon Bombardment, in: 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, American Institute of Aeronautics and Astronautics, Fort Lauderdale, Florida, 2004. https://doi.org/10.2514/6.2004-3788.
[16] R.D. Kolasinski, J.E. Polk, D. Goebel, L.K. Johnson, Sputtering yield measurements at glancing incidence using a quartz crystal microbalance, Journal of Vacuum Science & Technology A 25 (2007) 236–245. https://doi.org/10.1116/1.2435375.
[17] A.P. Yalin, J.D. Williams, V. Surla, K.A. Zoerb, Differential sputter yield profiles of molybdenum due to bombardment by low energy xenon ions at normal and oblique incidence, J. Phys. D: Appl. Phys. 40 (2007) 3194. https://doi.org/10.1088/0022-3727/40/10/025.
[18] E. Oyarzabal, R.P. Doerner, M. Shimada, G.R. Tynan, Carbon atom and cluster sputtering under low-energy noble gas plasma bombardment, Journal of Applied Physics 104 (2008) 043305. https://doi.org/10.1063/1.2968549.
[19] I. Bizyukov, A. Mutzke, R. Schneider, A.M. Gigler, K. Krieger, Morphology and changes of elemental surface composition of tungsten bombarded with carbon ions, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 266 (2008) 1979–1986. https://doi.org/10.1016/j.nimb.2008.03.211.
[20] A.S. Shumilov, I.I. Amirov, Morphology simulation of the surface subjected to low-energy ion sputtering, Tech. Phys. 60 (2015) 1056–1062. https://doi.org/10.1134/S1063784215070245.
[21] A.S. Shumilov, I.I. Amirov, Profile Evolution of Silicon Nanostructures in Argon-Plasma Sputtering, J. Synch. Investig. 14 (2020) 935–943. https://doi.org/10.1134/S1027451020050195.
[22] M.P. Harrison, D.A. Pearson, R.M. Bradley, Emergence and detailed structure of terraced surfaces produced by oblique-incidence ion sputtering, Phys. Rev. E 96 (2017) 032804. https://doi.org/10.1103/PhysRevE.96.032804.
[23] D.A. Pearson, R.M. Bradley, Theory of terraced topographies produced by oblique-incidence ion bombardment of solid surfaces, J. Phys.: Condens. Matter 27 (2014) 015010. https://doi.org/10.1088/0953-8984/27/1/015010.
[24] D.M. Goebel, B. Jorns, R.R. Hofer, I.G. Mikellides, I. Katz, Pole-piece Interactions with the Plasma in a Magnetically Shielded Hall Thruster, in: 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, American Institute of Aeronautics and Astronautics, 2014. https://doi.org/10.2514/6.2014-3899.

[25] P.M. Sforza, Chapter 14 - Space Propulsion, in: P.M. Sforza (Ed.), Theory of Aerospace Propulsion, Butterworth-Heinemann, Boston, 2012: pp. 541–565. https://doi.org/10.1016/B978-1-85617-912-6.00014-1.
[26] G.C. Soulas, The Impact of Back-Sputtered Carbon on the Accelerator Grid Wear Rates of the NEXT and NSTAR Ion Thrusters, in: Washington, D.C., 2013. https://ntrs.nasa.gov/citations/20150021367 (accessed October 31, 2022).
[27] G. Williams, T. Haag, J. Foster, J.V. Noord, S. Malone, T. Hickman, M. Patterson, Analysis of the Pyrolytic Graphite Ion Optics Following the 2000-hour Wear Test of the HiPEP Ion Thruster, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, 2006. https://doi.org/10.2514/6.2006-5005.
[28] W. Eckstein, R. Preuss, New fit formulae for the sputtering yield, Journal of Nuclear Materials 320 (2003) 209–213. https://doi.org/10.1016/S0022-3115(03)00192-2.

CHAPTER 5: THRESHOLD ENERGY FOR SPUTTERING OF MONATOMIC SURFACES WITH NOBLE GAS IONS [1]

[1] *Extracts of this chapter are currently under reviewed in a journal*

The scale-bridging MD-MC framework presented in Chapter 3 and 4 can be extended for any noble gas ion-target material systems. However, the model accuracy relies heavily on the provided elemental sputtering yield properties from MD simulation, expressed via fitting coefficients in the Eckstein and Preuss [1] sputtering model. For sputtering of other targets materials under noble gas ion bombardment, this semi-empirical model requires further recalibration of the fitting parameters, limiting the usage and predictability of such models. While many has treated the fitting parameters as phenomenological [1–4], one of the parameters, E_{th}, has a physical meaning, and is known as the sputtering threshold energy which is defined as the minimum (normal) ion incidence energy to initiate sputtering. While E_{th} is often extrapolated from the measured energy dependence of the sputtering yields for a specific ion-target system, the large scatter in the already minuscule sputter yield measurements at low ion energies leads to order-of-magnitude uncertainties in the extrapolated E_{th} [5–8]. For instance, Polk has shown that a correction in the double-charge ion, which makes the ion impact energy twice the intended energy in the experiment, can alter the extrapolated threshold energy of xenon bombardment on carbon from 5 eV to 38 eV [5]. These problems are exacerbated when using noble gas as the ion beam, resulting in experiment estimate of threshold energy that are independent of the mass ratio between the ion and the target's atom [7]. In this chapter, we quantify the sputtering threshold energies of monoatomic surfaces under the bombardment of noble gas ions across a broad range of ion-target combinations using molecular dynamics (MD) simulations. Section 5.1 described the modeling

approach to obtain sputtering threshold energy, which results, shown in Section 5.2, uncovers the major limitations of prior semi-empirical formulations for threshold energy, E_{th}. Section 5.3 describes a data-driven approach, based on symbolic regression through evolutionary algorithm to relate the MD threshold energies to the ion-target properties. Our new expression for threshold energy shows a strong functional dependence on the nucleus charge governing ion-target repulsion and the target density, in addition to the heat of sublimation and ion-target mass ratio in prior semi-empirical models. We discuss the physical implications of our new functions in Section 5.4 and conclude in Section 5.5 with a summary.

5.1 Sputtering Threshold Energy Modeling

Sputtering threshold energy, E_{th}, is a theoretical concept strictly defined as the critical ion incidence energy where the sputter yield goes to zero. However, the stochastic nature of the atomistic systems such as thermal vibration will lead to a range of possible E_{th} for given ion-target systems. Therefore, it is reasonable set an upper prediction bounds for E_{th} by assuming finite cut-off value for sputter yield. As a result, we operationally define the threshold energy as the smallest ion incidence energy, E_i, where at least one sputtered target is observed within 10,800 ion impact events. We use LAMMPS [9] to perform molecular dynamics simulations to perform the ion bombardment process.

5.1.1 Actual ion-targets

We first generate the ground truth sputtering threshold energy values for 175 ion-target combinations by performing molecular dynamics (MD) simulations on the bombardment of actual and realistic crystalline or amorphous targets with noble gas ions. Our face-centered cubic (FCC) or body-centered-cubic (BCC) crystalline targets have lattice spacing, a_{lat} (Å), and are fully periodic in the (x-y) directions (Fig. 5.1a). The (100), (110), or (111) free surface of these

structures are oriented in the vertical (z) direction, with corresponding ($18a_{lat} \times 18a_{lat} \times 12a_{lat}$), ($15a_{lat} \times 15\sqrt{2}a_{lat} \times 12a_{lat}$), or ($13\sqrt{2}a_{lat} \times 13\sqrt{2}a_{lat} \times 12a_{lat}$) crystalline lattice dimensions in the ($x \times y \times z$) directions, respectively. A vacuum gap of $6a_{lat}$ is introduced above the free surface of the simulation box to facilitate the bombardment of energetic noble gas ions. We model amorphous metals representing metallic glass by subjecting fully-periodic bulk crystalline metal lattices, each of dimensions ($18a_{lat} \times 18a_{lat} \times 12a_{lat}$), to a target temperature set close to the boiling point of the respective metals at a pressure of 1 bar, under an NPT ensemble for 10 ps with a time step of 1 fs. Subsequently, we switch to an NVT ensemble, and hold the temperature fixed for an additional 10 ps to ensure complete melting of the structure. We then reduce the time step to 0.5 fs and rapidly quench the respective systems under an NPT ensemble to room temperature (300 K) and pressure (1 bar) at a rate of 10^{14} K/s [10] (see Fig. 5.1b for amorphous a-Ta). This quenching rate matches the experimental condition to form metallic glass [3]. Studies have shown the quenching rate between $10^{12} - 10^{14}$ K/s does not significantly change the physical characteristics of the final amorphous structure, including the atomic density, porosity, and coordination number [4]. Finally, we introduce a $6a_{lat}$ vacuum gap along the z-direction of the simulation box to create a free surface to be exposed to energetic ions. The interatomic interactions of the metal atoms within these crystalline or amorphous targets are governed by corresponding Embedded Atom Method (EAM) potentials [5] listed in Table 5.1. Our carbon target has an amorphous subsurface which is created by bombarding multilayer graphene of dimensions ($48.5 \times 50.4 \times 50$ Å^3) repeatedly with 300 Xe ions under a normal ion incidence with ion energy of 200 eV to attain the amorphous carbon structure associated with steady-state sputtering detailed in Chapter 3 (Fig. 5.1c). The carbon-carbon interactions are governed by an Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) potential [14].

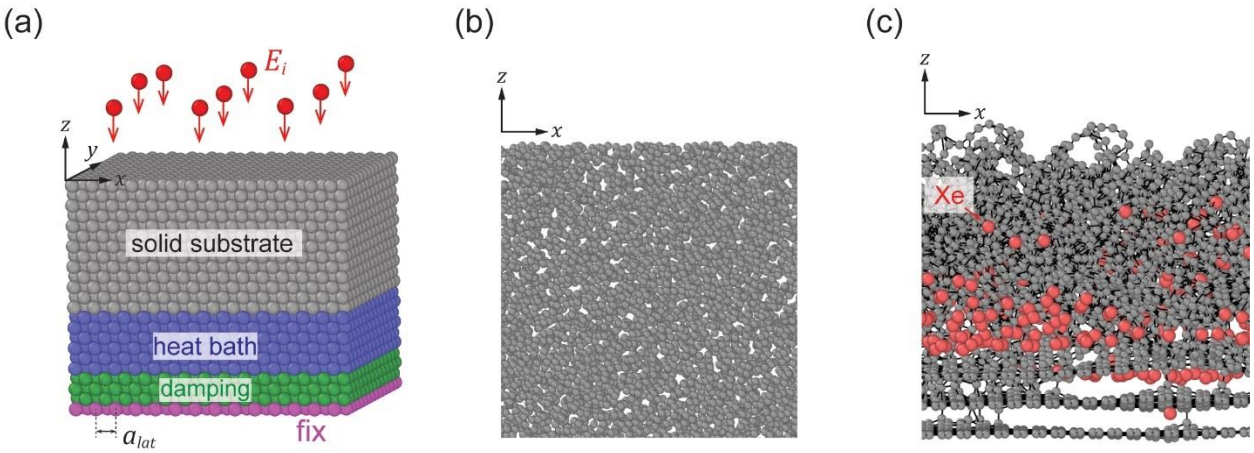

Figure 5.1: (a) Atomic configuration of a crystalline monoatomic target under the simultaneous bombardment of nine energetic ions (red), with designated fixed (purple), damping (green), heat bath (blue), and ion-bombarded (grey) layers. **(b,c)** Side views of the atomistic configurations of amorphous Ta (a-Ta) **(b)**, and amorphous carbon (a-C) associated with steady-state sputtering by Xe ion bombardment at 200 eV **(c)**.

For all our MD model structures, we rigidly fix the bottom $z \leq a_{lat}$ ($a_{lat} = 3.4$ Å for carbon) atoms throughout our simulations and designate the next $2a_{lat}$ layer above as the damping layer to absorb the stress waves caused by ion impact. This is followed by a $3a_{lat}$-thick heat bath layer (Fig. 5.1a). Prior to the bombardment sequence, we adopt a time step of 1 fs and subject both the heat bath layer and the remaining (ion-impacted) region above to an NVT ensemble maintained at 400 K by a Berendsen thermostat for 20 ps, which is representative of the surface temperature for plasma-facing materials [8]. A time-accelerated scheme to generate the ion impact events then follows. We divide the in-plane dimensions of the MD simulations box into 3×3 grids. For each bombardment sequence, we turn off the thermostat and simultaneously deposit 9 noble gas ions (one within each grid) at a $5a_{lat}$ vertical distance above the target surface. Each deposited ion is also at least $5a_{lat}$ away from each other to ensure 9 independent ion impact events. Each ion has an initial (vertical) ion velocity corresponding to the ion energy, E_i, and the ion mass, M_i. We resolve the initial impact dynamics for 1.2 ps with a time step of 0.1 fs, before switching on the

thermostat in the heat bath region, which is set to the target temperature of 400 K for the next 13 ps. Adopting a time step of 0.5 fs, we then quench both the heat bath and ion-impacted layer above to 400 K for the subsequent 5 ps. We repeated this process 1,200 times to achieve the desired 10,800 impact events.

We summarize the obtained sputtering threshold energy from our MD process, E_{th}^{MD}, in table 5.1, along with the reported lattice parameters, a_{lat}, and the sublimation energy or cohesion energy U_s. Specifically, U_s is a parameter used to represent the chemical binding energy of the target material and widely utilize to describe the sputtering process [ref]. For crystalline systems, U_s can be computed from the per atom potential energy of a fully-periodic simulation box representing the bulk material. For amorphous materials, the coordination number varies resulting in a distribution of U_s along with a distribution of atomic densities ρ_v. The former is computed from the difference in total potential energy between a (fully-periodic) bulk target and one with a randomly removed atom, and repeating this process multiple times to obtain a statistical distribution. The latter is the ratio between number of atoms within a moving window of ($10 \times 10 \times 10$ Å^3) just below the surface, and the associated volume constructed using alpha shape function [16].

Table 5.1: Summary of the sputtering threshold energy obtained from MD, E_{th}^{MD}, under the noble gas ion (He, Ne, Ar, Kr, Xe) bombardment of actual target materials with corresponding EAM potential shown in the [reference].

Target Surface		**Impact Ions**				
Species	**Atomic Structure**	**He** (eV)	**Ne** (eV)	**Ar** (eV)	**Kr** (eV)	**Xe** (eV)
Aluminum (Al) $U_s = 3.36$ eV $a_{lat} = 4.03$ Å [17]	FCC (100)	11.25	13.75	16.25	18.75	21.25
	FCC (110)	11.25	11.25	13.75	16.25	16.25
	FCC (111)	11.25	11.25	13.75	16.25	16.25
Vanadium (V) $U_s = 5.31$ eV $a_{lat} = 3.03$ Å	BCC (100)	21.25	26.25	33.75	41.25	46.25
	BCC (110)	18.75	28.75	36.25	38.75	43.75

Table 5.1 (cont.)

[18]	BCC (111)	21.25	28.75	31.25	33.75	38.75
	Amorphous	23.75	28.75	28.75	33.75	43.75
Chromium (Cr) $U_s = 4.20$ eV $a_{lat} = 3.53$ Å [19]	FCC (100)	18.75	26.25	33.75	41.25	43.75
Iron (Fe) $U_s = 4.28$ eV $a_{lat} = 2.88$ Å [18]	BCC (100)	18.75	26.25	36.25	41.25	43.75
	BCC (110)	18.75	28.75	31.25	33.75	38.75
	BCC (111)	18.75	31.25	33.75	36.25	36.25
Cobalt (Co) $U_s = 4.41$ eV $a_{lat} = 3.54$ Å [19]	FCC (100)	18.75	31.25	36.25	41.25	46.25
Nickle (Ni) $U_s = 4.45$ eV $a_{lat} = 3.52$ Å [19]	FCC (100)	23.75	31.25	36.25	43.75	48.75
	FCC (110)	21.25	28.75	31.25	38.75	38.75
	FCC (111)	21.25	28.75	33.75	36.25	41.25
Copper (Cu) $U_s = 3.54$ eV $a_{lat} = 3.62$ Å [20]	FCC (100)	16.25	23.75	26.25	33.75	38.75
	FCC (110)	21.25	23.75	26.25	31.25	33.75
	FCC (111)	16.25	21.25	23.75	31.25	33.75
Zirconium (Zr) $U_s = 6.22$ eV $a_{lat} = 3.62$ Å [21]	BCC (100)	26.25	23.75	23.75	31.25	31.25
Molybdenum (Mo) $U_s = 6.80$ eV $a_{lat} = 3.15$ Å [22]	BCC (100)	33.75	41.25	48.75	51.25	61.25
	BCC (110)	28.75	38.75	46.25	53.75	56.25
	BCC (111)	28.75	38.75	43.75	48.75	53.25
Palladium (Pd) $U_s = 3.91$ eV $a_{lat} = 3.88$ Å [20]	FCC (100)	21.25	26.25	28.75	36.25	38.75
Silver (Ag) $U_s = 2.85$ eV $a_{lat} = 4.09$ Å [20]	FCC (100)	13.75	18.75	21.25	23.75	26.25
	FCC (110)	13.75	16.25	16.25	18.75	21.25
	FCC (111)	11.25	18.75	18.75	21.25	21.25
Tantalum (Ta) $U_s = 8.01$ eV $a_{lat} = 3.30$ Å [23]	BCC (100)	46.25	43.75	48.75	58.75	61.25
	BCC (110)	43.75	41.25	43.75	48.75	53.75
	BCC (111)	46.25	43.75	48.75	51.25	53.75
	Amorphous	38.75	41.25	46.25	48.75	53.75

Table 5.1 (cont.)

Tungsten (W) $U_s = 8.90$ eV $a_{lat} = 3.16$ Å [18]	BCC (100)	71.25	56.25	58.75	71.25	73.75
	BCC (110)	83.75	46.25	53.75	58.75	66.25
	BCC (111)	73.75	48.75	51.25	58.75	66.25
	Amorphous	48.75	46.25	48.75	58.75	58.75
Platinum (Pt) $U_s = 5.70$ eV $a_{lat} = 3.92$ Å [20]	FCC (100)	41.25	38.75	43.75	53.75	53.75
	FCC (110)	43.75	41.25	43.75	48.75	48.75
	FCC (111)	43.75	41.25	43.75	46.25	46.25
Gold (Au) $U_s = 3.81$ eV $a_{lat} = 4.07$ Å [18]	FCC (100)	21.25	28.75	31.25	33.75	36.25
Carbon (C) U_s, a_{lat} (Fig. 4) [14]	Amorphous (Fig. S4)	-	-	-	-	33.75

5.1.2 Pseudo ion-targets

For the actual ion-targets above, the atomic interactions within the target materials are governed by realistic interatomic potentials (EAM potentials in the case of metals), while realistic ion and target masses (M_i, M_t) and consequently atomic numbers (Z_i, Z_t), are used in the Ziegler-Biersack-Littmark (ZBL) [24] potential governing ion-target interactions. To uncouple the parametric effects of M_i, M_t and other related material properties that will be discussed in Section 5.2, we also create a series of pseudo ion-targets in MD. The interatomic interactions between the target atoms are governed by a Lennard-Jones (LJ) pair potential

$$\begin{aligned} U &= 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) && r \le r_c \\ U &= 0 && r > r_c \end{aligned} \tag{5.1}$$

where r is the interatomic distance between pair atoms, and r_c is the cut-off radius which we set to 4 Å to resemble r_c for the EAM potentials governing our actual ion-targets. For a specified (ε, σ) pair, we start with an assumed a_{lat}, and create $(8a_{lat} \times 8a_{lat} \times 8a_{lat})$ FCC or BCC atomic crystal structures in a fully periodic simulation box. We minimize the total potential energy of the system through a conjugated gradient method while allowing the simulation box size to change. We compute a_{lat} and U_s for the energy-minimized structure, and introduce a $6a_{lat}$ vacuum gap along the z-direction of the simulation box to create a free surface to be exposed to the energetic ions. In total, we create 54 unique combinations (groups) of cohesion energy U_s, lattice parameter a_{lat}, crystal packing, and surface orientation through this process, as summarized in Table 5.2. Within each of these 54 groups, we introduce four pseudo targets of masses M_t $(= 11.2, 33.5, 89.4, 112$ amu), each subjected to the bombardment of four pseudo noble gas ions of masses M_i $(= 11.2, 33.5, 89.4, 112$ amu) to obtain 16 pseudo ion-targets per group, for a total of 864 ion-target combinations.

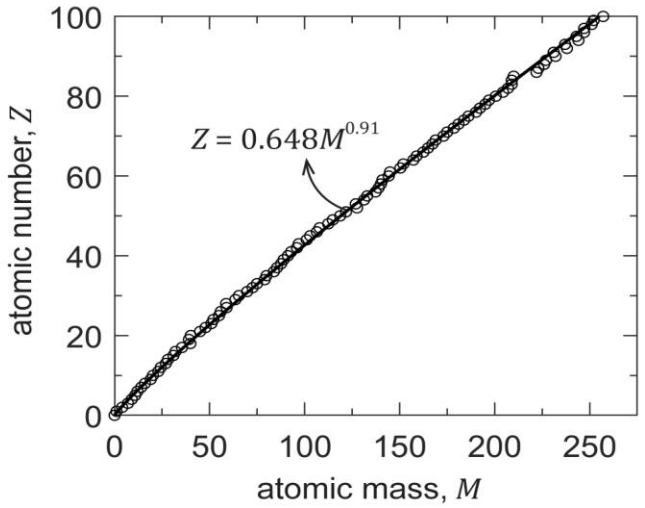

Figure 5.2: Relationship between the atomic number, Z, and the atomic mass, M, along with a best fit curve.

Each ion-target interaction is still governed by the same repulsive ZBL potential but we now expand the simulated values of M_i for these pseudo noble gas-like ions. Neglecting isotopes, each monoatomic element in the periodic table has a one-to-one relationship between M and Z

(circle symbols in Fig. S3). We assume $M = aZ^b$ and obtain the fitted parameters of $a = 1.61$ and $b = 1.1$ using maximum likelihood estimation (solid line in Fig. S3), based on all monoatomic species in the periodic table ($1 \leq Z \leq 118$).

Table 5.2: Summary of the (ε, σ) parameters of (S2) used to construct 864 pseudo ion-targets, comprising of 54 groups, each group with the same heat of sublimation (U_s), lattice parameter (a_{lat}), crystal packing (FCC or BCC), and surface orientation. Black and red 'x' denotes training and testing datasets.

BCC:

LJ parameter	$\boldsymbol{\varepsilon}$ **(eV)**	1.60	1.45	1.31	1.11	0.98	0.87	0.74	0.71	0.71	0.71	0.71	0.71	0.67
	$\boldsymbol{\sigma}$ **(Å)**	2.50	2.50	2.68	2.49	2.68	2.39	2.93	2.27	2.20	2.10	2.04	1.96	2.28
Property	$\boldsymbol{U_s}$ **(eV)**	9.18	8.47	8.01	6.80	6.01	5.33	4.82	4.81	4.81	4.81	4.81	4.81	4.12
	$\boldsymbol{a_{lat}}$ **(Å)**	3.28	3.29	3.51	3.27	3.52	3.15	3.83	2.99	2.88	2.75	2.67	2.58	3.01
Orientation	**100**	x	x	x	x	x	x	x	x	x	x	x	x	x
	110		x		x	x	x		x	x	x	x	x	x
	111		x			x	x			x	x	x	x	x

FCC:

LJ parameter	$\boldsymbol{\varepsilon}$ **(eV)**	0.86	0.66	0.65	0.63	0.52	0.51	0.41	0.40	0.34
	$\boldsymbol{\sigma}$ **(Å)**	2.54	2.24	2.64	2.46	2.30	2.57	2.52	2.32	2.53
Property	$\boldsymbol{U_s}$ **(eV)**	5.03	4.04	3.92	3.90	3.49	3.43	3.55	2.99	2.77
	$\boldsymbol{a_{lat}}$ **(Å)**	3.99	3.52	4.19	3.87	3.61	4.08	3.96	3.65	3.99
Orientation	**100**	x	x	x	x	x	x	x	x	x
	110	x	x	x	x	x	x		x	
	111	x	x	x	x	x	x		x	

5.2 Limitations of Prior Semi-empirical Formulations

Studies have rationalized E_{th} as a physical parameter representing the minimum ion impact energy for the target atom to overcome the chemical binding energy of the surface, leading to models of the form [17]

$$E_{th} = U_s f\left(\frac{M_t}{M_i}\right) \tag{5.2}$$

where (M_t, M_i) is the atomic mass of the (target, ion), and U_s is the heat of sublimation or cohesion energy of the target material. Many variants of $f\left(\frac{M_t}{M_i}\right)$ have been proposed. For instance, early theoretical inceptions of f were based on energy and momentum conservation [18]

$$f = \frac{1}{4}\left(1 + \frac{M_t}{M_i}\right)^2 \left(\frac{M_t}{M_i}\right)^{-1} \tag{5.3a}$$

but underestimate heavy ion sputtering, since the momentum of the ion at a normal ion incidence must reverse for the target atoms to escape [25]. A semi-empirical function, fitted to the extrapolated E_{th} from sputtering yield data across a vast ion-target dataset, was subsequently proposed [19]

$$f = 1.9 + 3.8\left(\frac{M_t}{M_i}\right)^{-1} + 0.134\left(\frac{M_t}{M_i}\right)^{1.24} \tag{5.3b}$$

Later theoretical considerations for inelastic collisions with further refinement from experiment data led to a clear delineation between the sputtering response for light and heavy ions [20]

$$\begin{aligned} f &= 3.35 + 1.675\left(\frac{M_t}{M_i}\right)^{-1} + 1.675\frac{M_t}{M_i}, && \frac{M_t}{M_i} \geq 1 \\ f &= 1.925 + 3.1\left(\frac{M_t}{M_i}\right)^{-1} + 1.425\left(\frac{M_t}{M_i}\right)^{-2} + 0.25\frac{M_t}{M_i}, && \frac{M_t}{M_i} < 1 \end{aligned} \tag{5.3c}$$

We first ascertain the predictive accuracies of (5.2) by comparing the various functional forms of f in (5.2a-c) with the ground truth sputtering threshold energy values for 175 ion-target combinations described in 5.1.1, computed from molecular dynamics (MD) simulations (Fig. 5.3a-

c, black). Based on the E_{th}^{MD} ground-truth predictions (Table 5.1), we find that (5.3a) drastically underestimates heavy ion sputtering (Fig. 5.3a) as previously reported [25]; (5.3b) and (5.3c) provide largely similar E_{th} predictions (Fig. 5.3b,c), though slight improvements are observed with (5.3c) due to the separate considerations for $M_t \geq M_i$ and $M_t < M_i$. Nevertheless, significant number of data points still reside outside the ± 20 eV error margin, particularly for the bombardment of heavy (Xe, Kr) and very light (He) ions.

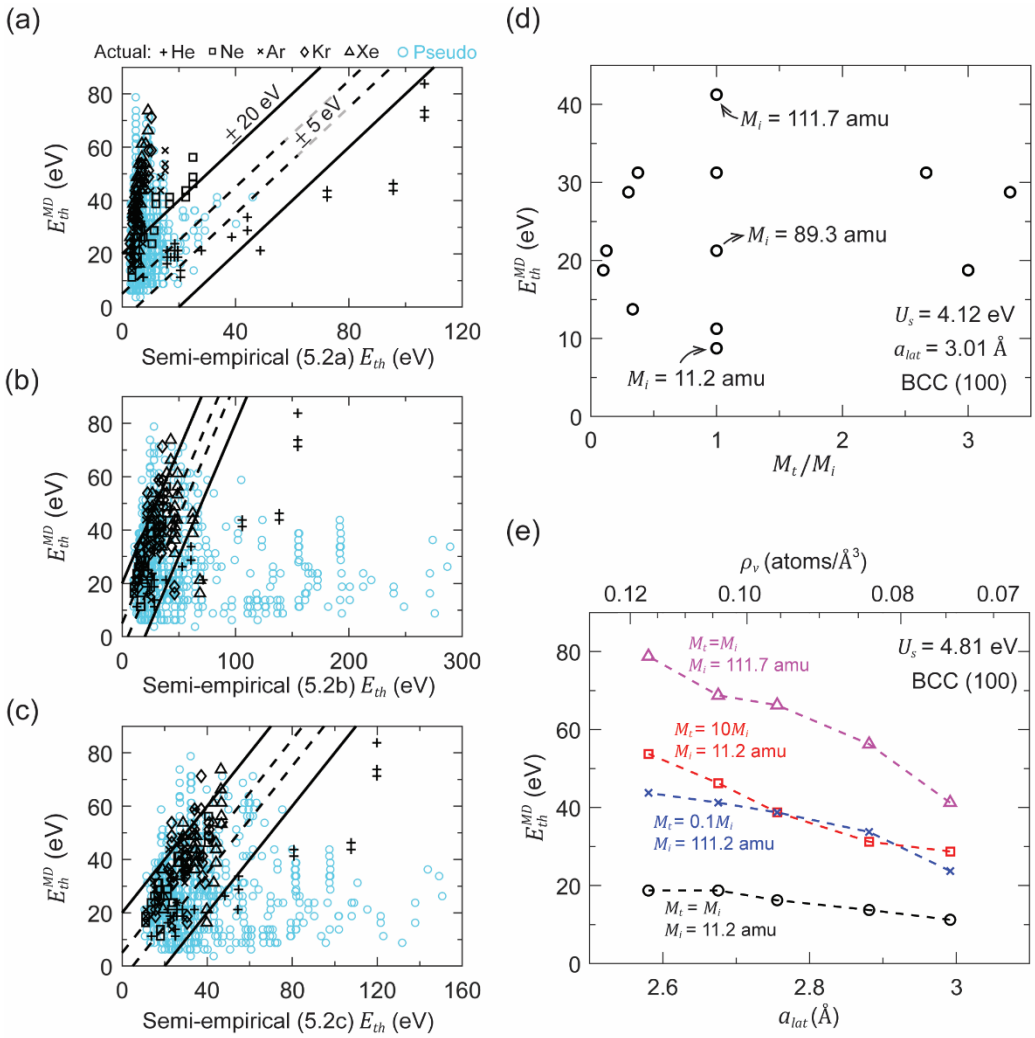

Figure 5.3. Sputtering threshold energy from MD (E_{th}^{MD}) versus semi-empirical predictions for actual (black) and pseudo (colored) ion-targets (**a-c**), with parametric dependence of E_{th}^{MD} on the mass ratio $\left(\frac{M_t}{M_i}\right)$ and ion mass (M_i) (**d**), as well as lattice parameter (a_{lat}) and atomic density (ρ_v) **(e)**, for pseudo ion-targets.

Some have argued that the energy to remove an atom from a free surface is higher than in the bulk and suggested the replacement of U_s – a bulk property – with the surface binding energy [10], $E_s = U_s + E_v^f$, where E_v^f is the vacancy formation energy. In our computations of E_s for crystalline metals, we calculate E_v^f from the difference between the total potential energy of the target crystalline substrate with an exposed surface, and one with a randomly-removed atom on the surface. Our results in Fig. 5.4a show a near linear scaling of $E_s = \sim 1.5 U_s$, with minimal differences among crystalline substrates with different surface orientations. As a result, the choice between E_s and U_s as the normalizing term for the sputtering threshold energy (E_{th}) cannot resolve the observed discrepancies between existing semi-empirical predictions and E_{th}^{MD} (Fig. 5.4b-c). Moreover, the sole dependence of E_{th} on $\left(\frac{M_t}{M_i}, U_s\right)$ in (5.2) suggests that details of the interatomic interactions between the ion and target atoms, as well as the physical properties of the target material, may be neglected – attributes we find from our data-driven formulation in this Chapter to significantly influence E_{th}.

As described in Section 5.1.2, we create a series of pseudo ion-target systems in MD simulations to allow us to uncouple the parametric effects of M_i, M_t, U_s, as well as the target crystal packing (FCC, BCC), surface orientation (n), and lattice spacing (a_{lat}), on E_{th}^{MD}. The pseudo threshold energy data E_{th}^{MD} consist of 864 unique pseudo ion-targets, which we subdivide into 54 groups, each group comprising of 16 ion-target combinations with the same U_s, a_{lat}, crystal packing, and surface orientation, but with varying pairs of (M_t, M_i). We overlay the computed E_{th}^{MD} versus the corresponding semi-empirical predictions of these pseudo ion-targets in Fig. 5.3a-c (colored), and again observe the lack of a clear correlation between the existing models (5.3a-c) and MD. In fact, the more extensive dataset in Fig. 5.3b-c reveals many instances where the same

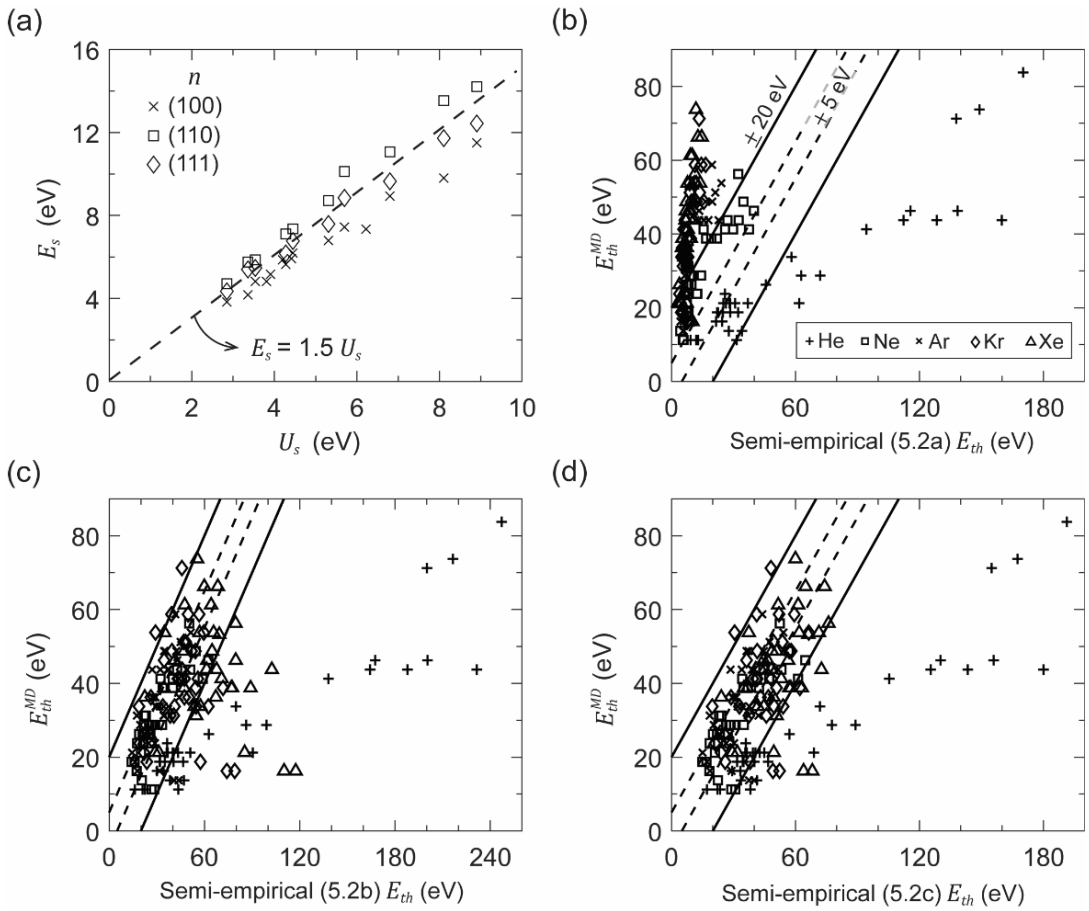

Figure 5.4: (a) Relationship between the surface binding energy, E_s, and heat of sublimation, U_s for various (100), (110), or (111) surface orientations. **(b-d)** Sputtering threshold energy from MD (E_{th}^{MD}) versus semi-empirical predictions ((5.2a-c) in the main paper) for actual ion-targets under noble gas ion bombardment.

E_{th} predictions – indicative of ion-targets with the same $\frac{M_t}{M_i}$ and U_s – can have substantially different E_{th}^{MD}. For the specific case where we vary $\frac{M_t}{M_i}$ but fix the remaining parameters in Fig. 5.3d, the vertical scatter in E_{th}^{MD}, corresponding to different M_i at $\frac{M_t}{M_i} = 1$, shows that the mass ratio alone is insufficient to predict E_{th}. Since the atomic mass has a one-to-one relationship with the atomic number, this dependence of E_{th} on both M_t and M_i can be associated with the ion-target interatomic interaction terms within the ZBL potential: $Z_t Z_i$, and $(Z_t^{0.23} + Z_i^{0.23})^{-1}$ representing the screened nuclear repulsion and ion-target interaction radius respectively. Additionally, we

observe a clear dependence of E_{th}^{MD} on a_{lat}, and in turn, the atomic surface density, ρ_v, for the various combinations of (M_t, M_i) (Fig. 5.3e).

5.3 A New Data-Driven Formulation for Sputtering Threshold Energy

Motivate by the observations in Section 5.2, we reformulate (5.2) to include the additional functional dependence on the atomic number, $Z_{i,t}$ and atomic surface density, ρ_v

$$E_{th} = U_s\, g(\rho_v)\, f\left(\frac{M_t}{M_i}, Z_t Z_i, (Z_t^{0.23} + Z_i^{0.23})^{-1}\right) \tag{5.4}$$

We remark that the validity of this expression for all noble gas ion bombardments on monoatomic targets would imply that the complicated ion-target interactions and physical target properties in determining E_{th} are solely dependent on (M_i, M_t, g, U_s); (5.4) is therefore not constrained to any specific interatomic potential for the target material in MD. As such, we obtain the closed-form expressions for f and g in (5.4) using a data-driven approach based on the input (M_i, M_t, ρ_v, U_s) versus output (E_{th}^{MD}) datasets of our 864 "pseudo" ion-targets. We then validate this formulation against ground-truth MD data from "actual" ion-targets, represented by the more realistic EAM potential for monoatomic metallic targets.

For ion-targets with the same (U_s, ρ_v), the multiplicative decomposition in (5.4) implies that E_{th} scales linearly with f with a slope of $U_s g$. It follows that within each of our j = 1 to 54 groups of pseudo ion-targets, each group with its unique (U_s, ρ_v), the computed E_{th}^{MD} should scale linearly with f. This linearity relationship is quantified through the correlation coefficient, r_j, corresponding to a best fit line through the origin obtained from linear regression; $r_j^2 = 1$ represents a perfect linear fit to the data, while $r_j^2 = 0$ denotes the complete lack of correlation to the data. Cumulatively, we optimize for the mathematical expression of f by minimizing the training fitness $K = 1 - \frac{1}{N}\sum_{j=1}^{N} r_j^2$ for $N = 37$ out of the 54 independent groups, while reserving

the remaining randomly selected 17 groups for testing (Red and black 'x' in Table 5.2). We represent each possible mathematical expression of f as a linear combination of different symbolic tree structures (maximum set at 4; see final optimized tree structure in Fig. 5.6a), scaled by a set of weights optimized through symbolic regression [30]. The terminal nodes on each tree structure are either constants or the input variables $\frac{M_t}{M_i}, Z_i Z_t, (Z_i^{0.23} + Z_t^{0.23})^{-1}$, and are connected to functional nodes selected from ($\times$, $-$, $+$, $\div$, $\sqrt{}$, exp, ln, power) operations. The total expressional complexity ε_c is the sum of the complexity of each tree, which we define as the summation of the node count itself and all its possible full sub-trees [25].

5.3.1 Symbolic regression through evolutionary algorithm

We utilize an evolutionary algorithm to sample a large range of candidate expressions for f in Eq. (5.4) using GPTIPS [19], *with the modified fitness K, provided in Appendix A.2*. As demonstrated schematically in Fig. 5.5a, for each independent run, an initial population of N genes is randomly created, with each gene representing a multi-tree mathematical expression [19]. For subsequent generations thereafter, Pareto tournaments are conducted [20,21] to rank N genes performance based on minimizing a predetermined fitness function K and expressional complexity, ε_c. Per example shown in Fig. 5.5b, ε_c for a single tree is defined as the summation of the node count itself and all its possible full sub-trees (defined as the connected path from any node, sans the root node, to the terminal node) [20]. For two expressions with the same number of nodes, this definition of expressional complexity will promote a "simpler" expression as a "flatter" tree. For instance, both trees t_1 and t_2 in Fig. 5.5b have three nodes each, with the top node as the root node. The first full sub-tree of both t_1 and t_2 has one node each (terminal node), contributing to an expressional complexity of 1. The second full sub-tree of t_2 has also one node (terminal node) with an additional expressional complexity of 1, but the second full sub-tree of t_1 has 2 nodes with

an additional expressional complexity of 2. Thus, the "flatter" t_2 expression with 2 levels has a smaller total expressional complexity of 5 $(= 3 + 1 + 1)$ versus 6 $(= 3 + 1 + 2)$ for the t_1 expression with 3 levels.

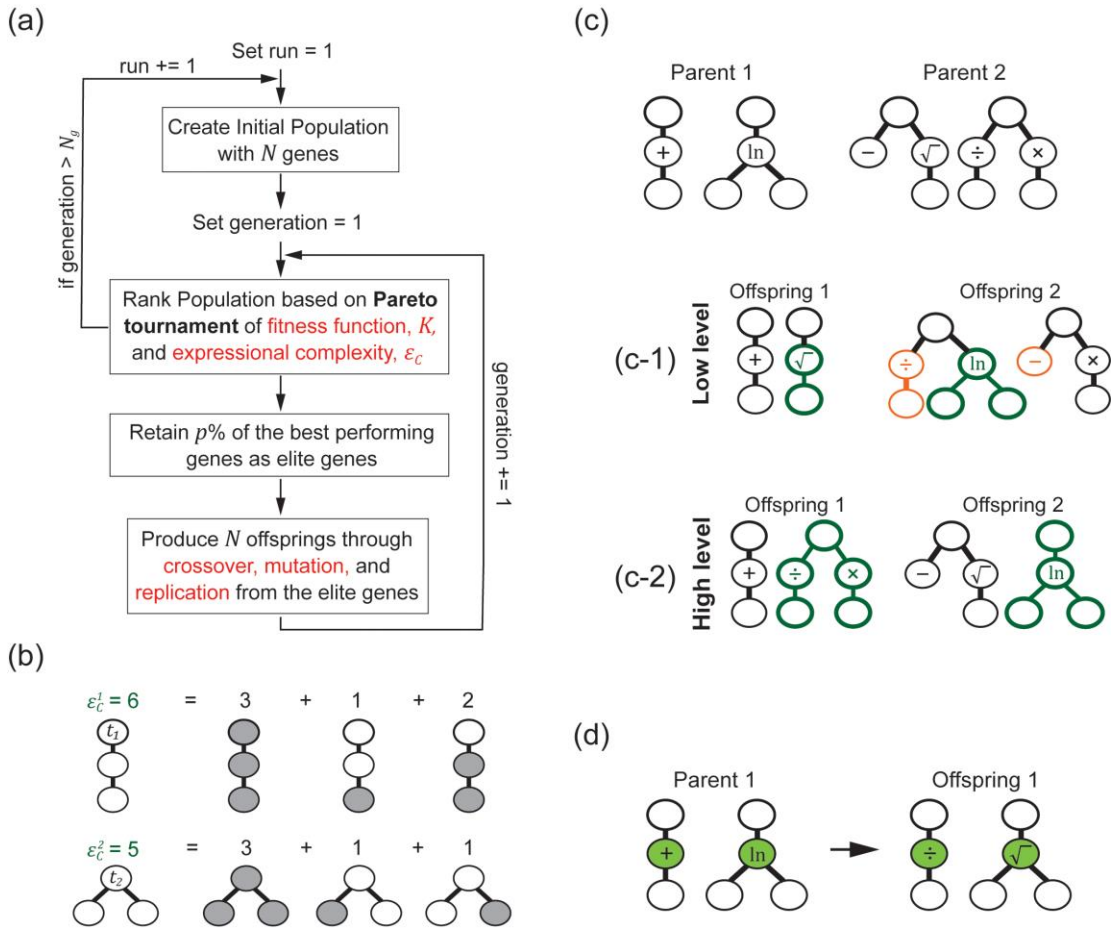

Figure 5.5: (a) Flowchart of the evolutionary algorithm for symbolic regression. **(b)** Illustration of expressional complexity for two different trees with equal nodes. **(c)** Illustration of crossover between parent 1 and 2 to produce offsprings 1 and 2, either through a low-level process of switching of the sub-tree branches from the same parent (orange) or from different parents (green) **(c-1)**, or through a high-level process where the entire tree is swapped between two parents **(c-2)**. **(d)** Illustration of mutation.

A $p\%$ of the best performing genes are designated as elite genes, where N offsprings are generated as the new population for the next generations from these elite genes through crossover, mutation, and/or replication [19,22]. As illustrated in Fig. 5.5c, a crossover can be at low level (c-

1), where the genes undergo sub-tree crossover between Parent 1 and 2 (green) and within Parent 2 (orange). In addition, crossover can occur at a high level (c-2), where the entire single tree components are swap between Parent 1 and 2 (green) to produce two offsprings. Note that to ensure N offsprings are generated from $\frac{Np}{100}$ parents, there can be multiple non-unique combination of parent pairs. In addition, the elite genes themselves can undergo mutation to produce offsprings, where the functional nodes are randomly changed to other possible functions (green in Fig. 5.5d). Furthermore, replication can occur where the offspring is simply the elite gene. This process is then repeated for N_g generations per run. In this work, $N = 250, p = 30$ and $N_g = 160$ generations to ensure $\min(K)$ has converged. To minimize the statistical effects of initial population selection from the first generation, we conduct 200 independent runs to arrive at a total of $200N$ possible expressions. In the end, we represent the optimal choices for K versus ε_c for all these $200N$ expressions, via a Pareto front. We repeat this procedure for different combinations of input variables for f to construct the different Pareto fronts in Fig. 5.6.

5.3.2 Results

We use an evolutionary algorithm to sample a large spatial range (~50,000) of possible symbolic expressions for f [24,26], and show the Pareto front [27] of K versus ε_c in Fig. 5.6b for each of the 7 possible combinations of the input variables for f (inset). In all the Pareto fronts, K is rapidly minimized within $\varepsilon_c \leq 10$. The transition point in the Pareto front, where further increase in ε_c does not significantly reduce K, represents the local optimal function for each possible combination of input variables. The global optimal function (Fig. 5.6b, filled red circle) has a training fitness $K = 0.126$ with individual r_j^2 ranging from 0.63-0.97, and is expressed as

$$f = 0.6\frac{M_t}{M_i} + 7.3(Z_i Z_t)^{0.25} \tag{5.5}$$

The expression has similar testing fitness ($K = 0.137$) and r_j^2 range (0.78-0.94) for the remaining 17 testing groups, which suggests no overfitting. In comparison, $K \sim 1$ for the prior semi-empirical

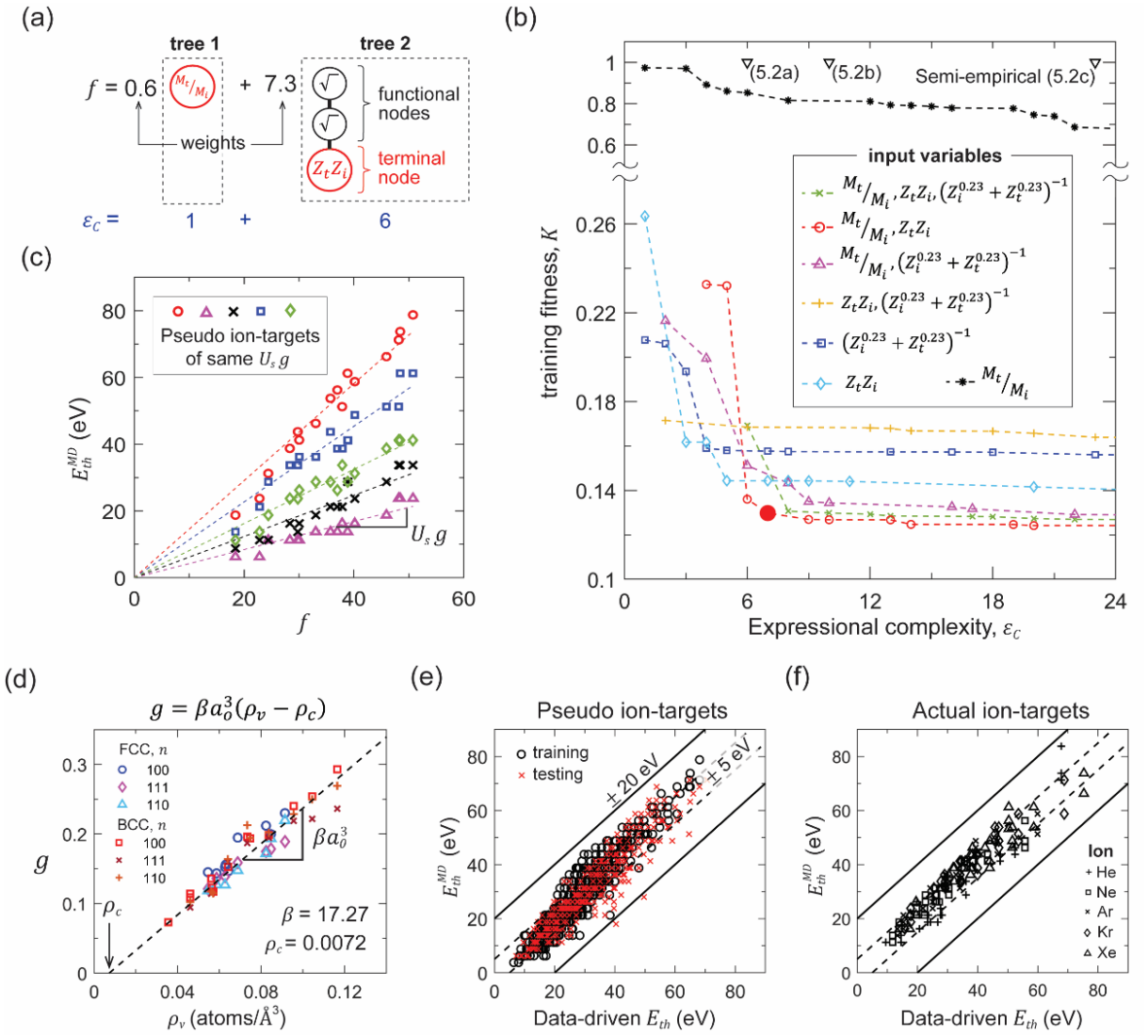

Figure 5.6: Symbolic expression of sputtering threshold energy ($E_{th} = U_s g f$) from evolutionary algorithm: (**a**) Two-tree symbolic representation of the global optimal f. (**b**) Pareto front of training fitness (K) and expressional complexity (ε_c) for f trained on pseudo ion-targets, with different combinations of input variables, along with global optimal f (solid red symbol). (**c**) Linear scaling of the MD threshold energy (E_{th}^{MD}) versus global optimal f for representative pseudo ion-target groups, each with constant $U_s g$. (**d**) Linear scaling between g versus the atomic density (ρ_v) independent of crystal packing and surface orientation, n. (**e,f**) E_{th}^{MD} versus data-driven predictions for pseudo (**e**) and actual (**f**) ion-targets, representing the training/testing (**e**) and validation (**f**) datasets.

formulations in (5.2a-c), confirming the failure of these formulations to capture the linearity dependence of E_{th} versus f, even when $f\left(\frac{M_t}{M_i}\right)$ is further optimized (Fig. 5.6b, black dashed lines).

The strong linearity fit between E_{th} versus f in (5.5) for all 54 individual groups of pseudo ion-target data (representative groups in Fig. 5.6c) highlights the validity of (5.4). The gradient of each linear relation ($U_s g$) provides us with a single data point for $g(\rho_v)$. Cumulatively, we observe a linear relationship between g and ρ_v, with negligible effects of crystal packing (FCC versus BCC) and surface orientation (n) (Fig. 5.6d). Therefore, we propose the functional dependence

$$g = \beta a_o^3 (\rho_v - \rho_c) \tag{5.6}$$

where the normalization constant $a_o = 0.529$ Å is the Bohr radius, and $\beta (= 17.27)$ and $\rho_c (= 0.0072$ atoms/Å$^3)$ are two parameters fitted to the $g(\rho_v)$ data in Fig. 5.6d using maximum likelihood estimation. The fitted value of $\rho_c = 0.0072$ atoms/Å^3 coincides with having ~2 atoms within an atomic sphere of influence of radius $r_c = \sim 4$ Å, which is the cut-off radius for both the LJ and EAM potentials. When $\rho_v = \rho_c$ in (5), $g = E_{th} = 0$. Therefore, ρ_c physically represents the smallest possible atomic density for an energetic ion to be necessary for sputtering of a monoatomic surface, below which the target would simply sublimate on its own. The fitted ρ_c from our pseudo data supports this implication, since there would be no interatomic interaction within the target material if ρ_c comprises of fewer than 2 atoms within r_c.

Together, the expressions in (5.4)-(5.6) represent our new data-driven formulation for E_{th}. While (5.5) is solely derived from the training data of pseudo ion-targets, the predictions from our final expression for E_{th} compare equally well with E_{th}^{MD} across the training and testing pseudo ion-target datasets (Fig. 5.6e). More importantly, when (5.4)-(5.6) is applied to the actual ion-target dataset based on more realistic EAM potentials for the target materials (Fig. 5.6f), our data-driven

predictions are in near perfect agreement with the corresponding E_{th}^{MD} across the spectrum of light to heavy ions with most errors falling well within ± 5 eV (Fig. 5.7).

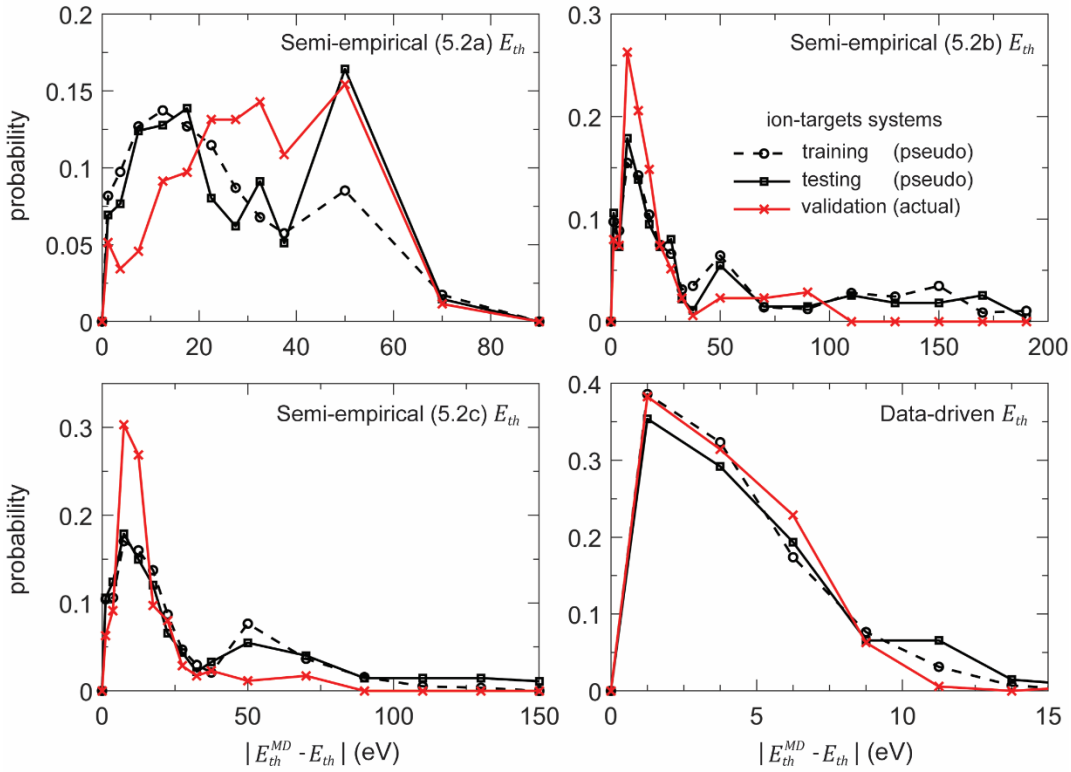

Figure 5.7: Probability of the absolute error differences in 2.5 eV intervals, comparing the sputtering threshold energy from MD (E_{th}^{MD}) versus E_{th} from prior semi-empirical equations and from our data-driven formulation.

5.4 Discussions and Implications

Threshold for sputtering represents the upper energy limit of the energetic incidence ion where there are no to very low amount of ejected target's material. It serves as a crucial component to accurately predict sputtering yield at very low ion incidence energy [1–4]. While prior semi-empirical models assume the sole dependence of f on $\frac{M_t}{M_i}$, which is associated with momentum exchange, our data-driven formulation for f in (4) also shows that $Z_i Z_t$, governing the screened nuclear repulsion between the ion and target atoms, is important. Because Z scales directly with

M, f is effectively a function of both $\frac{M_t}{M_i}$ and M_i (Fig. 5.8a). For heavy ions (Xe, Kr), the momentum exchange term is negligible indicating that E_{th} is governed primarily by screened nuclear repulsion. Conversely, for very light ions (Ne, He), the momentum exchange term now plays a primary role in the sputtering process, explaining the prior empirical delineation of E_{th} in (2c) into two separate $\frac{M_t}{M_i}$ regimes. This transition also corresponds to distinct changes in the physical

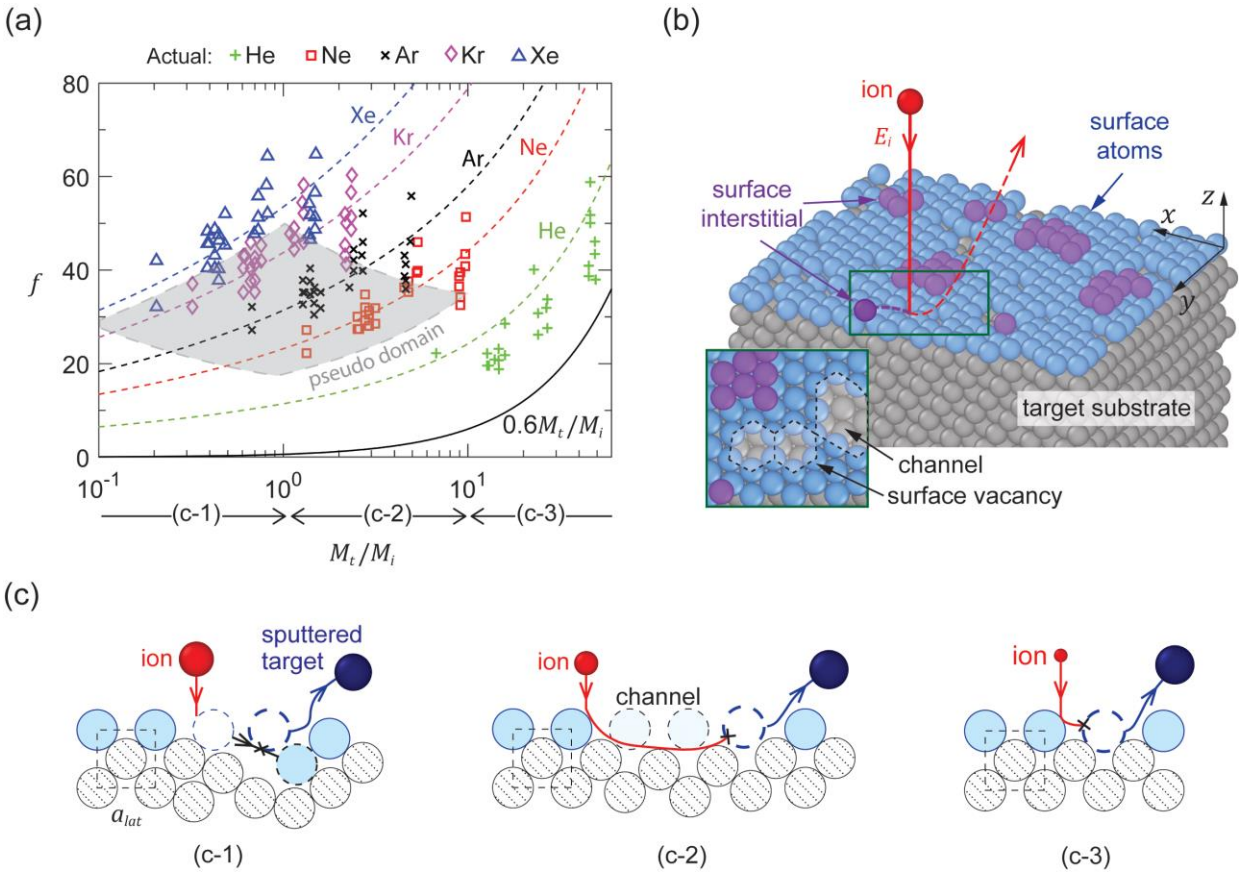

Figure 5.8: (**a**) Variation of f versus mass ratio $\left(\frac{M_t}{M_i}\right)$ for different noble gas ions. Symbols: actual ion-targets, shaded region: pseudo ion-target domain, dashed lines: expressional predictions combining momentum exchange and screened nuclear repulsion, solid line: momentum exchange contribution. (**b**) Atomistic snapshot for ion bombardment near the threshold energy depicting the formation of vacancy-interstitial pairs, and subsequent reconfiguration to form contiguous channels. (**c**) Transitions in the sputtering mechanisms with increasing $\frac{M_t}{M_i}$: surface collision cascade (c-1), channel sputtering (c-2), and primary knock-on sputtering (c-3).

sputtering mechanism as revealed by our MD simulations. At ion energies close to E_{th}^{MD}, the impact of the incoming ion can displace the surface atom of the target to create surface vacancy-interstitial pairs, with the latter located at the absorption sites (Fig. 3b). Collectively, the surface vacancies can rearrange themselves to form contiguous channels. When $1 \leq \frac{M_t}{M_i} \leq \sim 10$, both momentum exchange and nuclear repulsion affect sputtering (Fig. 5.7a), and the incoming ion or impacted surface atom can glide along these channels to sputter a target atom several lattice spacings away from the initial ion impact site (Fig. 5.7c-2). When M_i is very small ($M_t \geq \sim 10 M_i$), channel sputtering is no longer necessary since the ions are sufficiently small to penetrate the interstitial sites to cause sputtering of primary knock-on atoms by momentum exchange (Fig. 5.7c-3). Conversely, for very large ions ($M_t \leq M_i$), the highly repulsive ion-target interactions (large $Z_i Z_t$) drive the sputtering of secondary and at most tertiary knock-on atoms from surface collision cascades near the impact site (Fig. 5.7c-1). We further remark that the domain of our pseudo ion-target dataset (Fig. 5.8a, grey regime) used in the construction of f in (5.5) is much smaller (but denser) than the domain encompassed by the EAM-based verification dataset (Fig. 5.8a, symbols).

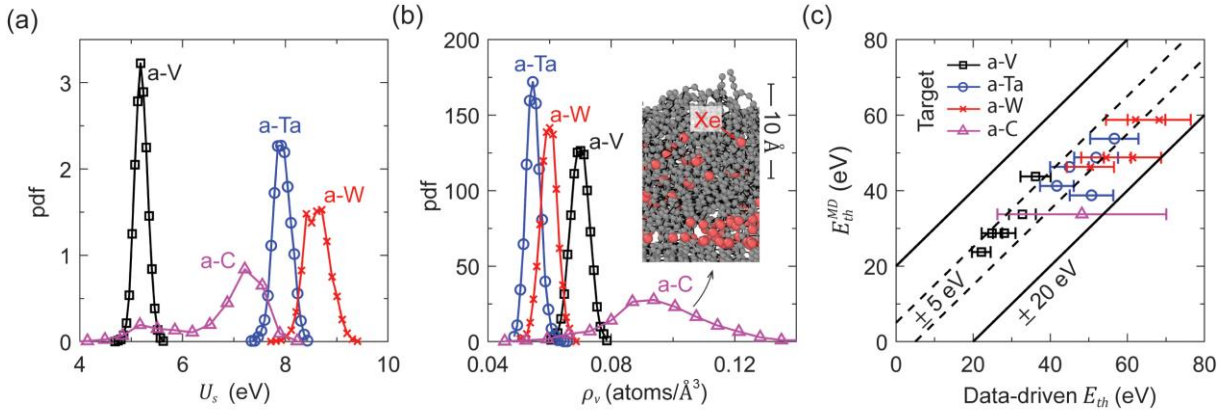

Figure 5.9: (**a,b**) Probability distribution function (pdf) of sublimation energy (U_s) (**a**) and atomic density (ρ_v) (**b**) for amorphous V, Ta, W, and C. Inset in (**b**): atomistic configuration of amorphous C from Xe ion bombardment [32]. (**c**) Sputtering threshold energy from MD (E_{th}^{MD}) versus data-driven predictions for noble gas bombardment of amorphous systems. Error bars: 2-standard deviation uncertainties in predictions propagated from pdfs in (**a,b**).

Nevertheless, our data-driven predictions of f show good extrapolation capabilities towards the actual EAM-based dataset even far beyond this training envelop.

Our data-driven formulation for E_{th} is based on monoatomic crystalline metals. However, the formulation is insensitive to the crystal packing or surface orientation. In addition, the details of the interatomic interaction within the target material appear to be unimportant, outside of ρ_v and U_s, as evidenced by similarly small error distributions when the formulation is applied to EAM (validation) and LJ (training-testing) data (Fig. 5.7). These suggests the generalizability of this formulation to non-metals as well as amorphous materials. To demonstrate this, we model a series of amorphous metallic glass (a-V, a-Ta, a-W) in MD [11,12,10] to quantify E_{th}^{MD}, described in section 5.1.1. Unlike crystalline targets, U_s and ρ_v for amorphous metals have a distribution (Fig. 5.9a,b), resulting in a corresponding uncertainty [34] in the predicted E_{th} from (5.4)-(5.6) (Fig. 5.10). Even with the 2-standard-deviation uncertainties, the range of predicted E_{th} is still in great agreement with the simulated E_{th}^{MD} (Fig. 5.9c). Finally, we test the applicability of our data-driven formulation to carbon (a non-metal) amorphized by Xe ion bombardment [35] (inset in Fig. 5.9b).

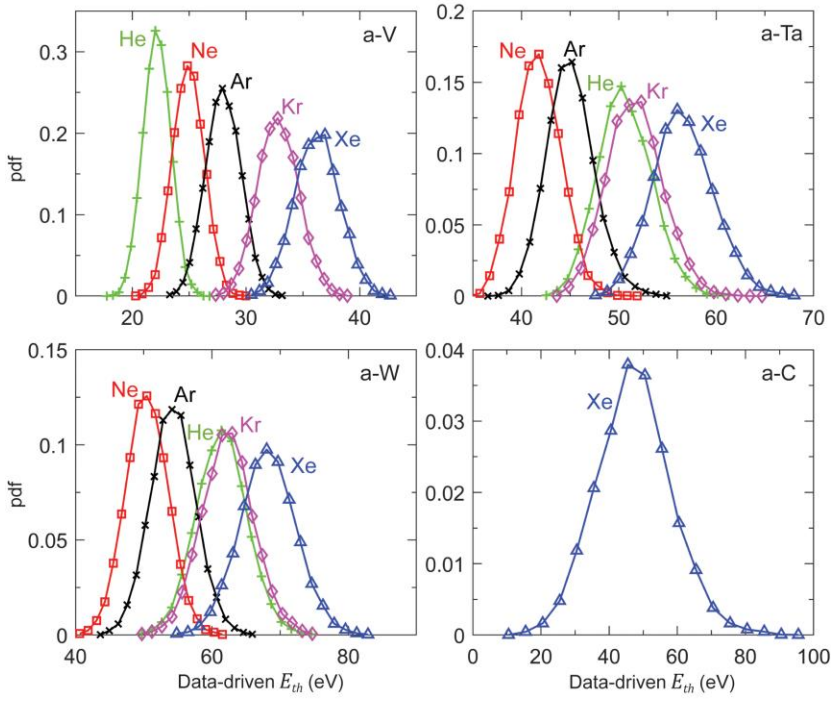

Figure 5.10: Probability distribution function (pdf) of the uncertainty in the predicted E_{th} from our data-driven formulation for amorphous W, Ta, V (a-W, a-Ta, a-V) under the bombardment of noble gas ions (He, Ne, Ar, Kr, Xe), and amorphous carbon (a-C) under Xe ion bombardment. Each pdf is constructed from 10,000 randomly sampled values for U_s and ρ_v from the respective pdfs in Fig. 5.9

Based on the measured distributions of U_s and ρ_v, we obtain a 2-standard-deviation uncertainty of 47.4 ± 26 eV in the predicted E_{th} (Fig. 5.9c), where the lower range falls mostly within the ± 5 eV error band (dash-line). We remark that for these statistical amorphous systems, the threshold energy for sputtering should reside at the lower range of the predicted E_{th} that corresponds to weaker bonds and interactions (low cohesion and atomic density) between target atoms.

5.5 Conclusions

In conclusion, a new data-driven expression for the sputtering threshold energy, applicable to the noble gas ion bombardment of crystalline and amorphous targets, has been formulated based on a massive ion-target dataset generated by MD simulations. The expression has strong underlying physics, and integrates the combined contributions of mass ratio, screened nuclear repulsion, and target atomic density to achieve significantly improved predictions over prior semi-empirical models. The formulation has fundamental implications in plasma-surface physics with broad applicability to plasma manufacturing and engineering.

5.6 References

[1] W. Eckstein, R. Preuss, New fit formulae for the sputtering yield, Journal of Nuclear Materials 320 (2003) 209–213. https://doi.org/10.1016/S0022-3115(03)00192-2.

[2] P. Sigmund, Sputtering by ion bombardment theoretical concepts, in: R. Behrisch (Ed.), Sputtering by Particle Bombardment I, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981: pp. 9–71. https://doi.org/10.1007/3540105212_7.

[3] Y. Yamamura, S. Shindo, An empirical formula for angular dependence of sputtering yields, Radiation Effects 80 (1984) 57–72. https://doi.org/10.1080/00337578408222489.

[4] C. García-Rosales, W. Eckstein, J. Roth, Revised formulae for sputtering data, Journal of Nuclear Materials 218 (1995) 8–17. https://doi.org/10.1016/0022-3115(94)00376-9.

[5] J.E. Polk, A critical review and meta-analysis of xenon-on-carbon sputter yield data, Journal of Applied Physics 135 (2024) 040701. https://doi.org/10.1063/5.0180720.

[6] R.V. Stuart, G.K. Wehner, Sputtering Yields at Very Low Bombarding Ion Energies, Journal of Applied Physics 33 (1962) 2345–2352. https://doi.org/10.1063/1.1728959.

[7] D. Rosenberg, G.K. Wehner, Sputtering Yields for Low Energy He^+-, Kr^+-, and Xe^+-Ion Bombardment, Journal of Applied Physics 33 (1962) 1842–1845. https://doi.org/10.1063/1.1728843.

[8] G.H. Al-Malkawi, A.-M.B.A. Al-Ajlony, K.F. AL-Shboul, Computer simulation of the sputtering energy thresholds for some plasma-facing component materials irradiated with helium, deuterium, and tritium ions, Nuclear Materials and Energy 35 (2023) 101442. https://doi.org/10.1016/j.nme.2023.101442.
[9] A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in 't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, R. Shan, M.J. Stevens, J. Tranchida, C. Trott, S.J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Computer Physics Communications 271 (2022) 108171. https://doi.org/10.1016/j.cpc.2021.108171.
[10] A. Khmich, K. Sbiaai, A. Hasnaoui, Structural behavior of Tantalum monatomic metallic glass, Journal of Non-Crystalline Solids 510 (2019) 81–92. https://doi.org/10.1016/j.jnoncrysol.2019.01.024.
[11] L. Zhong, J. Wang, H. Sheng, Z. Zhang, S.X. Mao, Formation of monatomic metallic glasses through ultrafast liquid quenching, Nature 512 (2014) 177–180. https://doi.org/10.1038/nature13617.
[12] S. Trady, M. Mazroui, A. Hasnaoui, K. Saadouni, Molecular dynamics study of atomic-level structure in monatomic metallic glass, Journal of Non-Crystalline Solids 443 (2016) 136–142. https://doi.org/10.1016/j.jnoncrysol.2016.04.004.
[13] M.S. Daw, M.I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Phys. Rev. B 29 (1984) 6443–6453. https://doi.org/10.1103/PhysRevB.29.6443.
[14] S.J. Stuart, A.B. Tutein, J.A. Harrison, A reactive potential for hydrocarbons with intermolecular interactions, J. Chem. Phys. 112 (2000) 6472–6486. https://doi.org/10.1063/1.481208.
[15] R.B. Lobbia, J.E. Polk, R.R. Hofer, V.H. Chaplin, B. Jorns, Accelerating 23,000 hours of Ground Test Backsputtered Carbon on a Magnetically Shielded Hall Thruster, in: AIAA Propulsion and Energy 2019 Forum, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-3898.
[16] H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–72. https://doi.org/10.1145/174462.156635.
[17] X.-Y. Liu, F. Ercolessi, J.B. Adams, Aluminium interatomic potential from density functional theory calculations with improved stacking fault energy, Modelling Simul. Mater. Sci. Eng. 12 (2004) 665. https://doi.org/10.1088/0965-0393/12/4/007.
[18] P.A.T. Olsson, Semi-empirical atomistic study of point defect properties in BCC transition metals, Computational Materials Science 47 (2009) 135–145. https://doi.org/10.1016/j.commatsci.2009.06.025.
[19] D. Farkas, A. Caro, Model interatomic potentials for Fe–Ni–Cr–Co–Al high-entropy alloys, Journal of Materials Research 35 (2020) 3031–3040. https://doi.org/10.1557/jmr.2020.294.
[20] X.W. Zhou, R.A. Johnson, H.N.G. Wadley, Misfit-energy-increasing dislocations in vapor-deposited CoFe/NiFe multilayers, Phys. Rev. B 69 (2004) 144113. https://doi.org/10.1103/PhysRevB.69.144113.
[21] M. Zhou, B. Fu, Q. Hou, L. Wu, R. Pan, Determining the diffusion behavior of point defects in zirconium by a multiscale modelling approach, Journal of Nuclear Materials 566 (2022) 153772. https://doi.org/10.1016/j.jnucmat.2022.153772.
[22] M.W. Finnis, J.E. Sinclair, A simple empirical N-body potential for transition metals, Philosophical Magazine A 50 (1984) 45–55. https://doi.org/10.1080/01418618408244210.

[23] R. Ravelo, T.C. Germann, O. Guerrero, Q. An, B.L. Holian, Shock-induced plasticity in tantalum single crystals: Interatomic potentials and large-scale molecular-dynamics simulations, Phys. Rev. B 88 (2013) 134101. https://doi.org/10.1103/PhysRevB.88.134101.
[24] J.F. Ziegler, J.P. Biersack, The Stopping and Range of Ions in Matter, in: D.A. Bromley (Ed.), Treatise on Heavy-Ion Science: Volume 6: Astrophysics, Chemistry, and Condensed Matter, Springer US, Boston, MA, 1985: pp. 93–129. https://doi.org/10.1007/978-1-4615-8103-1_3.
[25] W. Eckstein, C. Garciá-Rosales, J. Roth, J. László, Threshold energy for sputtering and its dependence on angle of incidence, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 83 (1993) 95–109. https://doi.org/10.1016/0168-583X(93)95913-P.
[26] E. Hotston, Threshold energies for sputtering, Nuclear Fusion 15 (1975) 544–547.
[27] N. Matsunami, Y. Yamamura, Y. Itikawa, N. Itoh, Y. Kazumata, S. Miyagawa, K. Morita, R. Shimizu, H. Tawara, Energy Dependence of Ion-Induced Sputtering Yields of Monatomic Solids, Atomic Data and Nuclear Data Tables 31 (1984) 1. https://doi.org/10.1016/0092-640X(84)90016-0.
[28] Y. Yamamura, H. Tawara, Energy Dependence of Ion-Induced Sputtering Yields from Monoatomic Solids at normal incidence, Atomic Data and Nuclear Data Tables 62 (1996) 149–253. https://doi.org/10.1006/adnd.1996.0005.
[29] X. Li, X. Zhang, Y. Xu, G. Lei, S. Liu, H. Li, Z. Cui, Y. Zhu, J. Hu, S. Geng, X. Chen, H. Liu, X. Wang, J. Huang, H. Liu, J. Cheng, J. Shen, H. Lan, C. Tang, Molecular dynamics study of surface binding energy and sputtering in W-V alloys, Fusion Engineering and Design 195 (2023) 113971. https://doi.org/10.1016/j.fusengdes.2023.113971.
[30] D.P. Searson, GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining, in: A.H. Gandomi, A.H. Alavi, C. Ryan (Eds.), Handbook of Genetic Programming Applications, Springer International Publishing, Cham, 2015: pp. 551–573. https://doi.org/10.1007/978-3-319-20883-1_22.
[31] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997) 341–359. https://doi.org/10.1023/A:1008202821328.
[32] G.F. Smits, M. Kotanchek, Pareto-Front Exploitation in Symbolic Regression, in: U.-M. O'Reilly, T. Yu, R. Riolo, B. Worzel (Eds.), Genetic Programming Theory and Practice II, Springer US, Boston, MA, 2005: pp. 283–299. https://doi.org/10.1007/0-387-23254-0_17.
[33] J.M. Chaves-González, M.A. Pérez-Toledano, A. Navasa, Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection, Engineering Applications of Artificial Intelligence 43 (2015) 89–101. https://doi.org/10.1016/j.engappai.2015.04.002.
[34] R.C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, Society for Industrial and Applied Mathematics, USA, 2013.
[35] H. Tran, H.B. Chew, Surface morphology and carbon structure effects on sputtering: Bridging scales between molecular dynamics simulations and experiments, Carbon 205 (2023) 180–193. https://doi.org/10.1016/j.carbon.2023.01.015.

CHAPTER 6: FINAL REMARKS AND FUTURE RESEARCH DIRECTIONS

6.1 Key Findings

The primary goal of this thesis was to understand the fundamental physics of the sputtering process at low ion incidence energy and in turn, develop a sputtering constitutive model to advance nano-micro surface manufacturing and space electric propulsion technology. In pursuit of this goal, atomistic simulations upscaled to Monte Carlo models were used to uncover the multiscale mechanisms of carbon structure evolution at the atomistic-scale, and surface morphology evolution at the meso-scale, under xenon ion bombardment. In addition, the multiscale simulations were used to quantify the sputter yield as well as the differential yield (ejection angle) and energy distribution of the sputterants relevant to carbon transport modeling. Moreover, with symbolic regression through evolutionary algorithm, we arrived at a general sputtering threshold energy for monoatomic solids under noble gas ion bombardments. The key findings are summarized below.

Despite ongoing efforts over the past several decades, studies have reported an order-of-magnitude variation in the measured sputter yield of carbon at ion energies relevant to EP devices. In addition, the influence of different allotropes of carbon, and surface morphology effects on the sputtering yield characteristics were not well quantified either experimentally or computationally. In Chapter 3, large-scale, massively parallel MD simulations on the energetic bombardment of xenon ions on different carbon substrates were performed. The results revealed that the continuous bombardment process results in rapid amorphization of the carbon subsurface. More interestingly, my MD results showed virtually indistinguishable amorphous sub-surface structures regardless of the ion incidence energies, angles, and initial carbon structures – a finding that has been reported experimentally but was demonstrated computationally for the first time.

Additionally, surface morphology effects are well-known to influence the sputtering yield during ion bombardment but cannot be directly accounted for with MD simulations because of the inherent length- and time-scale limitations. In Chapter 4, the MD simulation results were upscaled to a Monte Carlo (MC) model to account for the evolving surface morphology and sputtering yield with ion fluence. This scale-bridging approach uncovered the mechanisms underpinning several experimentally observed phenomena, including, the smoothening and flattening of initially rough surface morphologies by a normal ion flux, and the formation of characteristic surface steps at the microscale under an oblique ion flux. Quantitatively, my MC results showed that the transient-to-steady-state surface morphology evolution during the noble gas bombardment of carbon surfaces is responsible for significant fluctuations in the sputter yield – the bounded uncertainties well encompass a large portion of the experimental scatter across the range of ion energies and incidence angles. The predictions of this MD-MC approach were further used to design surface patterns with vertical thin sacrificial walls to enhance the sputter resistance capabilities of EP thruster and wall chamber surfaces. Finally, using a Bayesian-based Markov Chain Monte Carlo approach, the parameters of a semi-empirical carbon sputtering model based on MD-MC simulation results were calibrated. This engineering model is currently being used for carbon transport modeling in the NASA center JANUS.

The semi-empirical model used in Chapter 3 and 4 is based on Eckstein & Preuss [1], with multiple fitted parameters to be calibrated with sputter yield data at low sputtering energies for a specific ion-target system. Chapter 5 takes a radically different approach by arriving at a general expression for the threshold energy for sputtering of monoatomic surfaces with noble gas ions using symbolic regression through evolutionary algorithms, trained on massive amounts of data generated by MD. While prior semi-empirical models assume the sole dependence of threshold

energy on target-ion mass ratio and heat of sublimations, my new proposed data-driven formulation also shows that the screened nuclear repulsion between the ion and target atoms, along with the atomic density, is equivalently important. This new expression has strong underlying physics, with order-of-magnitude improved threshold energy predictions over prior semi-empirical models and is applicable to both crystalline and amorphous monoatomic targets.

6.2 Towards a Sputtering Theory of "Everything"

Eckstein & Preuss [1] sputtering model, combining equation (3.4) and (3.6) in Chapter 3, is expressed as

$$Y_{total}(E_i,\theta) = Qs_n \frac{\left(\frac{E_i}{E_{th}}-1\right)^{\mu}}{\frac{\lambda}{w}+\left(\frac{E_i}{E_{th}}-1\right)^{\mu}} (\cos(\theta_i^c))^{-f} \exp(\boldsymbol{b}(1-(\cos(\theta_i^c))^{-1})) \quad (6.1)$$

The Eckstein formulation has seven fitting parameters $Q, \lambda, \mu, E_{th}, f, c, b$, to correct for low energy ion sputtering. As aforementioned in Chapter 1, experiment data for the sputter properties of other noble gas ions on other target materials of interest is extremely limited at low ion incidence energy because of very low sputter yield [2], which leads to large uncertainties in the fitted values in Eckstein formulation [3]. Sputtering data is also scarce under an oblique ion incidence angle, and there is a marked absence of critical data such as the differential yield, and energy distribution of the sputterants. Studies have attempted to supplement the existing sparse experimental dataset with computational data, notably using sputtering simulation codes, such as Transport of Ions in Matter (TRIM) or the Stopping Range of Ions in Matter (SRIM), which are based on binary collision approximation (BCA) [4,5]. However, BCA is known to breakdown at these low (< 2 keV) ion energies of relevance to EP [4]. Even in the case of abundant and high-resolution experiment data, the fitted, non-physical parameters in Eckstein formulations are still specific to a given ion-target material system, hence, limiting the generalizability to other ion-target combinations. Using a data-

driven approach, a physics-based functional form of one of the parameters E_{th} (and incidentally the parameter which has the largest uncertainty) has been constructed in Chapter 5.

A natural next step in future research would be to construct the functional form of the remaining fitting parameters in (6.1), i.e., identifying closed-form, data-driven, analytical expressions for each of these individual fitting parameters as a function of the noble gas ion incidence energy and the physical/chemical/mechanical properties of both the ion and the target. Ultimately, this would render (6.1) to become fully predictive. To do so, we will need to generate the ground truth data, consisting of sputter yield, differential yield profile, and sputterant energy distribution for 18 different monoatomic solids (Al, Si, Ti, V, Cr, Fe, Co, Ni, Cu, Zr, Mo, Pd, Ag, Ta, W, Pt, Au, Pb) under 5 noble gas ion bombardment (He, Ne, Ar, Kr, Xe) under a wide range of ion incidence energies and angles from MD simulations. These generated MD data will be used to fit the respective functional expressions for Q, λ, μ, f, c, b in (6.1) using Symbolic Regression, following a similar approach detailed in Chapter 5. In the event that the search domain for the possible expressions are multivariate and complex, one can potentially utilize various machine learning techniques proposed in several literature to aid in the search process for symbolic regression, including deep learning [6], graph neural networks [7], Bayesian neural networks [8] or neural-guided genetic programing [9,10].

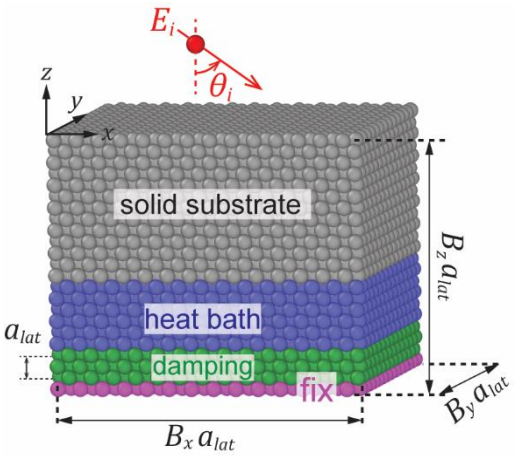

Figure 6.1: Schematic of a crystalline monoatomic target under bombardment of an energetic ion at incidence energy E_i and incidence angle θ_i. The (100) free surface of this substrate are oriented in the vertical (z) direction, with $(B_x a_{lat} \times B_y a_{lat} \times B_z a_{lat})$ crystalline lattice dimensions in the $(x \times y \times z)$ directions

Preliminary results: To achieve the objectives detailed above, a foremost step is the generation of sputter data for an extensive range of ion-target combinations. To this end, I have conducted MD simulations (Fig. 6.1) for the bombardment of 4 different noble gas ions (Ne, Ar, Kr, Xe) on 16 different monoatomic solids (Al, Ti, V, Cr, Fe, Co, Ni, Cu, Zr, Mo, Pd, Ag, Ta, W, Pt, Au) under a normal ion incidence angle, $\theta_i = 0°$, for 8 different ion incidence energies ($E_i = 50, 100, 200, 500, 750, 1000, 1500, 2000$ eV). This constitutes a total of 512 $(= 4 \times 16 \times 8)$ cases. As a preliminary run, I performed only 100 ion-surface impacts per case, and report the sputtering yield in Fig. 6.2. Minor fluctuations in the steady-state sputter yield as a function of the ion incidence energy for $E_i > 500$ eV can be observed due to the small number of ion

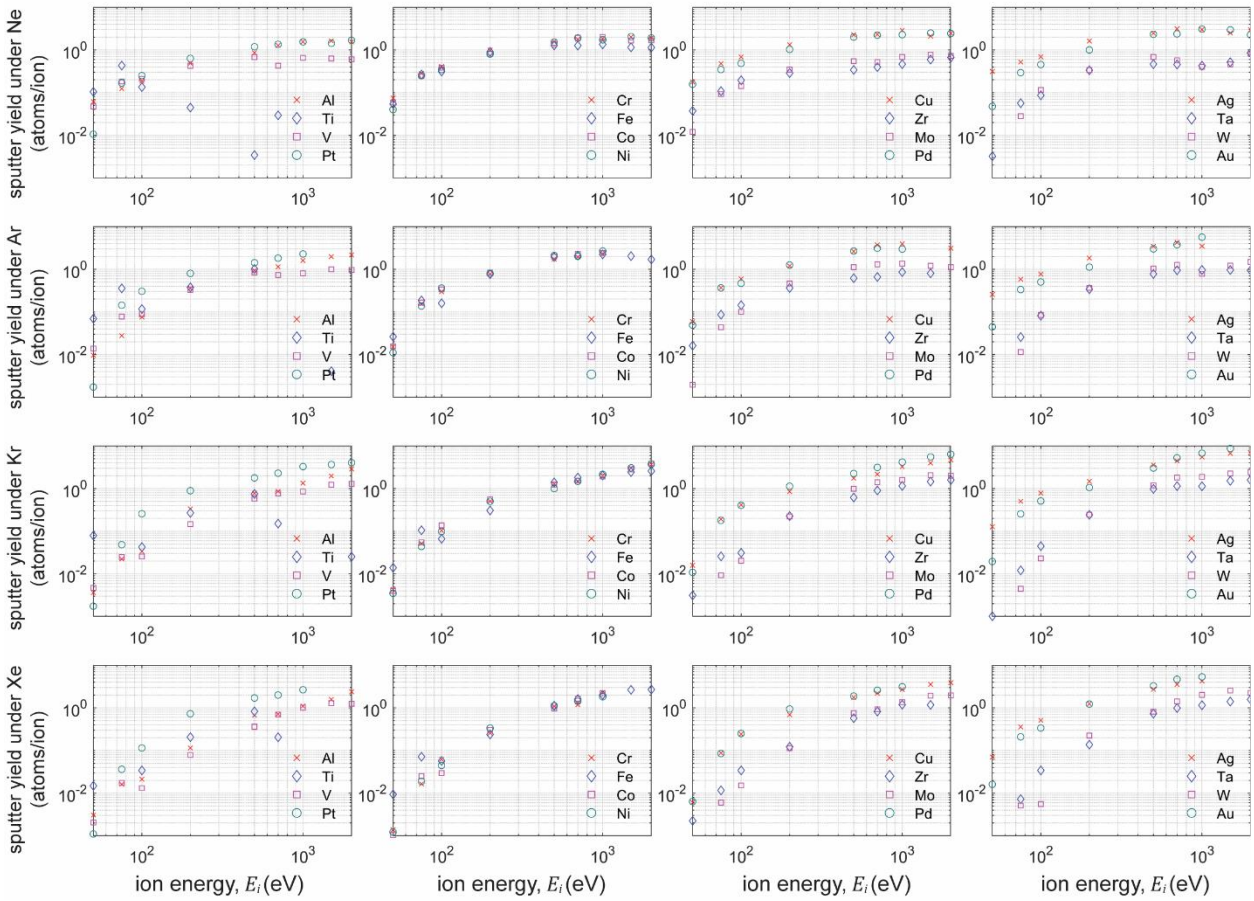

Figure 6.2: Sputter yield as a function of ion incidence energy, $\boldsymbol{E_i}$ at normal ion incidence angle ($\boldsymbol{\theta_i}$) across various realistic ion-target systems. The sputter yield is obtained from 100 ion bombardment event per case.

bombardments, suggesting that a higher fluence is required to arrive at a statistically significant sputter yield.

***Challenges and Future Directions*:** Ideally, the sputter yield, differential yield, and sputterant energy distribution should converge to the correct with increasing number of ion impact events, increasing size of the simulated target surface, and increasing simulation time to resolve the impact dynamics. However, to construct a sufficiently diverse dataset for symbolic regression, one would require the sputter properties for the bombardment of 5 noble gas ions (He, Ne, Ar, Kr, Xe) on 18 targets (Al, Si, Ti, V, Cr, Fe, Co, Ni, Cu, Zr, Mo, Pd, Ag, Ta, W, Pt, Au, Pb), at 8 different ion incidence energies ($E_i = 0.05 \rightarrow 2$ keV), and at 6 ion incidence angles ($\theta_i = 0°, 15°, 30°, 45°, 60°, 75°$), constituting a total of 4320 ($= 5 \times 18 \times 8 \times 6$) cases. The major challenge here is that these simulations combined are extremely computational intensive, even with current state-of-the-art high-performing computer. Thus, we would first need to identify the optimal simulation parameters to obtain the correct sputter properties within acceptable accuracy. To do so would necessitate detailed understanding of the spread and depth of the collision cascades within the target (Fig. 6.3), which could vary with impact energy, angle, as well as ion-target properties. For instance, BCA simulations such as TRIM or SRIM suggest that the damage zone

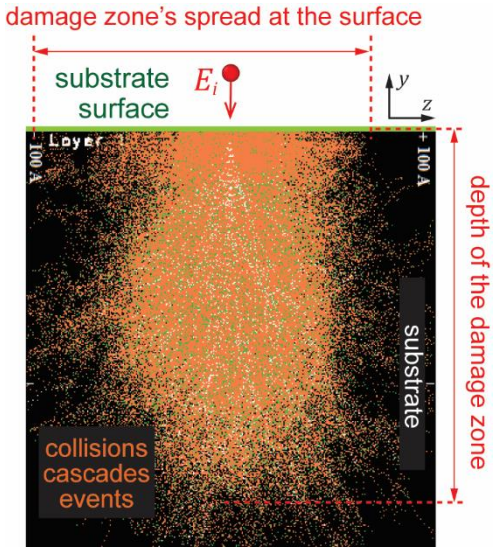

Figure 6.3: SRIM [4] simulations for Kr bombardment on Al at normal ion incidence angle with energy $E_i = 2$ keV, showing the spread and the depth of the damage zone caused by a series of collisions cascades events (orange). Collision cascades events at the surface, hence the spread of the damage zone, cause the ejections of the surface atoms, i.e. sputtering.

is deep but narrow for incoming ions with small mass [4]. More importantly, BCA results [11] have shown that the sputter yield is independent of the damage depth beyond 30Å, while most of the sputtered atoms arise from the spread of the collision cascades (Fig. 6.3). Clearly, the damage mechanism, cascading size, and effects on sputtering properties, in relation to the ion incidence energy, angle, and ion-target combination, must be established prior to simulating the 4320 cases.

6.3 Sputtering in the Presence of Carbon Contaminations

In many plasma engineering devices, the sputtered atoms from the bombardment of energetic ions can subsequently deposit onto other (often different) material surfaces. In the case of EP, a significant amount of sputtered carbon atoms from the accelerator grid, pole cover, and facility walls can be transported back and get redeposited onto critical thruster component surfaces, including the carbon pole covers, center-mounted cathode, anode, molybdenum's ion engine grid, and boron nitride channels [3,12,13]. Carbon contamination of the surfaces of these critical thruster components can severely affect the measured erosion rates, leading to significant uncertainties in EP thrusters' lifetime and performance assessments [12,14]. A possible future work is the extension of the MD-MC framework presented in Chapters 3 and 4 to include the effects of carbon contamination, as outlined in Fig. 6.4. The sputter element properties (yield, angle and energy distribution) for each element in the MC models will not only be a function of the ion incidence energy and angle, but also the extent of carbon contamination, φ, on the target surface. This new parameter φ can be defined as the ratio between thickness of the carbon contaminated layer, Δd, and the ion penetration depth of the carbon substrate, d_p. For $\varphi \geq 1$, the elemental sputter properties are those of carbon (Chapter 3). For $\varphi = 0$, there is no carbon contamination, and the sputter properties are those of the elemental material (proposed future research in Section 6.1).

The objective of this future research will be to obtain the sputtering properties in between these limits ($0 < \varphi < 1$).

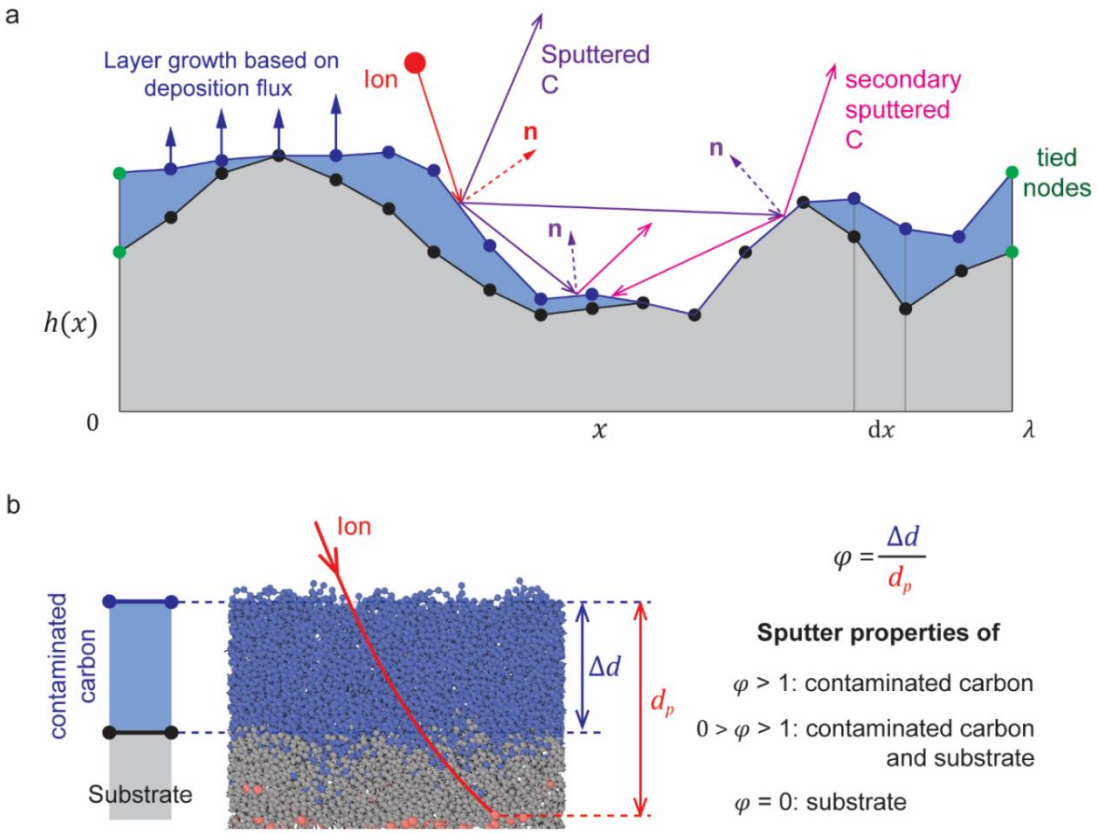

Figure 6.4: (a) Monte Carlo modeling of sputtering-deposition caused by impingement of energetic ions (red) on an undulating surface (grey) with various thickness of carbon contamination layer (blue), knocking-off primary (purple) and secondary (pink) sputterants. (b) Schematic demonstrated the calculation of **φ** for a surface element in the MC model in (a). **φ** is the ratio between the contaminated carbon's depth, **Δd** (blue) and the ion average penetration depth ($\mathbf{d_n}$).

To quantify the evolving φ from its initial pristine (carbon free) state, one would need to compute the absorptivity (sticking coefficient) of carbon, c_s, on different substrate material surfaces. As a preliminary result, I first quantify c_s for a carbon substrate in Section 6.3.1 below as a function of carbon incidence energy and angle, representing the deposition of carbon atoms on carbon substrates. Quantifying c_s for other substrate materials is more challenging, and two possible approaches are discussed in Section 6.3.2.

6.3.1 Carbon deposition on carbon

Carbon deposition simulations in MD are performed by slowly depositing carbon onto a substrate following previous approaches [15–17]. Fig. 6.5 shows the atomistic snapshot of the growing deposited carbon layer onto a diamond substrate, as well as onto an amorphous carbon (a-C) substrate created by xenon ion bombardment. The carbon deposition energy is set to $E_C = 5\ (eV)$ under a normal incidence angle $\theta_C = 0°$. The evolving atomistic structures are then partitioned into vertical bins of 0.35 nm (representing the diamond lattice parameter), and the average proportion of sp, sp^2 and sp^3 bonds, along with the atomic density, ρ, are computed within each bin. The bottom two rows in Fig. 6.5 shows the spatial distribution of sp, sp^2 and sp^3 bond proportions and $\rho\left(\frac{atom}{cm^3}\right)$ corresponding to four different deposition stages. The initial structure of a-C is somewhat porous, and the deposited carbon atoms help densify the a-C subsurface, increasing both the percentage of sp^2 bonds and the atomic density of the subsurface layer. Since carbon energy of 5 eV is below the threshold energy for sputtering of diamond or a-C, the newly deposited amorphous carbon layer is formed without causing sputtering of the underlying diamond or a-C substrate. As such, the spatial distribution of bond ratios and atomic density, ρ, of the as-deposited amorphous carbon layers are nearly identical regardless of the underlying substrate (diamond or a-C). At $E_C = 5\ (eV)$ and $\theta_C = 0°$, the deposited carbon structure consists of 89% sp^2, 5% sp^3 and $\rho = 1.095 \times 10^{23}\left(\frac{atom}{ion}\right)$.

However, changing the incidence angle of the deposited carbon atom θ_C does have a pronounced effect on the structural characteristics of the as-deposited layer, especially at high θ_C. Figure 6.6 shows snapshots of the atomistic configuration after 8000 carbon atoms are deposited at $E_C = 5\ (eV)$ and at $\theta_C = 0°, 30°, 60°$ (top row). Also included are the spatial distributions of

porosities within each deposited layer, generated using alpha-shape method [18]. While the porosity distribution appears very similar between $\theta_C = 0°, 30°$, the shadowing effect and the small vertical damage energy causes the deposition at $\theta_C = 60°$ to have higher overall porosities

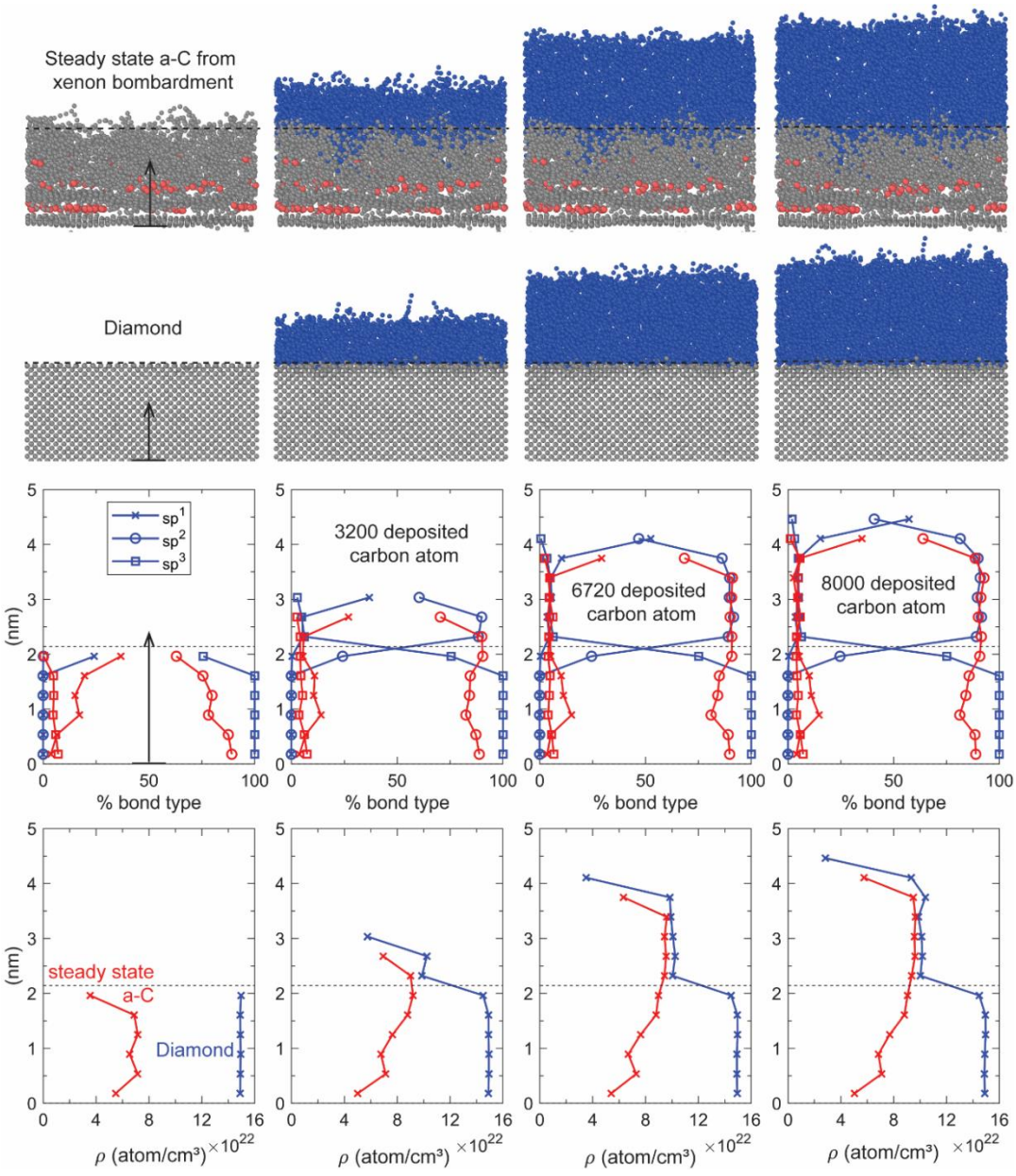

Figure 6.5: Atomic configurations of the growing carbon deposited layer on to diamond and steady state a-C from xenon bombardment (top two rows) at the carbon energy of $E_C = 5\ (eV)$ and angle $\theta_C = 0°$.The bottom two rows shows the spatial distribution of sp, sp^2 and sp^3 bond proportions and $\rho\left(\frac{atom}{cm^3}\right)$ corresponding to four different deposition stages, comparing steady state a-C substrate (red) with diamond substrate (blue).

with the much larger porosity clusters (black region in Fig. 6.6a). The deposited surface also is notably rougher with larger surface height undulations. The increased porosity at $\theta_C = 60°$ also results in an amorphized layer with lower (higher) quantity of sp^2 (sp) bonds as well as a lower atomic density. The proportion of sp, sp^2, and sp^3 bonds, along with the atomic density, ρ, at various incidence carbon energy and angle are summarized in Table 6.1. Results show that changing the carbon incidence angle and energy has a pronounced effect on the structural characteristics of the as-deposited layer, hence the carbon absorptivity c_s (Fig. 6.6b), especially at high θ_C.

Table 6.1: The proportion of sp, sp^2 and sp^3 bonds, along with the atomic density, ρ at various carbon incidence contamination energy and angle.

E_C	θ_C	%sp	%sp^2	%sp^3	ρ ($\times 10^{23}$ $atom/cm^3$)
1 eV	**0°**	12.8	85.3	1.9	0.986
	30°	15.6	83	1.4	0.945
	60°	23.7	75.4	0.9	0.77
5 eV	**0°**	6	89	5	1.095
	30°	7.7	88	4.3	1.058
	60°	13	83	4	0.899
20 eV	**0°**	2.8	87	10.2	1.199
	30°	2.3	88	9.7	1.171
	60°	6.9	86.5	6.6	1.07
Steady state a-C created by xenon bombardment		24	72	4	0.72

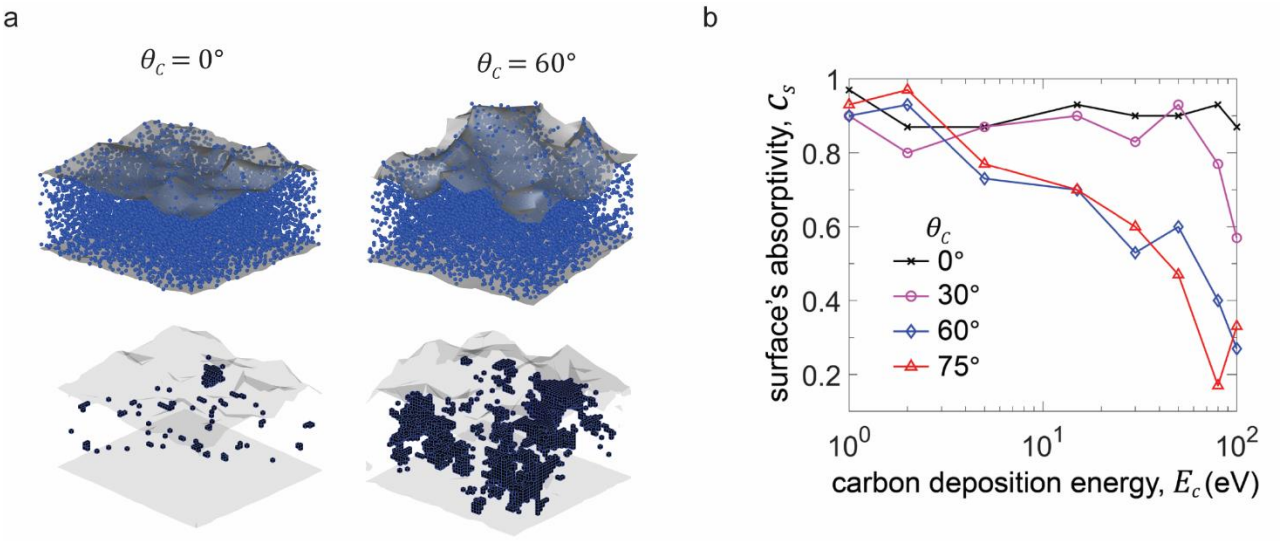

Figure 6.6: (a) snapshots of the atomic configuration after 8000 carbon atoms are deposited at $E_C = 5\ (eV)$ and at $\theta_C = 0°, 60°$ (top row). Also included are the spatial distributions of porosities within each deposited layer (black region in the bottom row). (b) Carbon surface's absorptivity, c_s as a function of the contaminated carbon energy, E_C (eV) for various contamination angle θ_C

6.3.2 Challenges and future directions: carbon deposition on general substrates

The accuracy of the MD simulations closely depends on the interatomic potential (IAP) governing the interaction between atoms in the system. While potentials governing the interaction between carbon atoms are well-established, those that involve the complex interactions between C atoms and other species (representing the contaminated surface compositions) are often non-existent. In the final section of this thesis, I propose two possible approaches to overcome this challenge and provide some preliminary results.

The first approach is to develop IAPs that are suitable for carbon deposition simulations using machine learning via artificial neural networks [19,20]. For instance, the potentials for C-B-N and C-Mo necessary for modeling the sputtering of carbon contaminated boron nitride and molybdenum surfaces are non-existent in the current literature. In constructing these potentials, one can adopt a machine learning-based interatomic potential (MLIAP), detailed in Fig. 6.7. The

input for the neural network is characterized by an environment descriptor, G_i, which ensures that the input of the neural network is independent of the coordination of the atom. In addition, the descriptor needs to be invariant with respect to translation and rotational operations and the exchanges of any equivalent atom. Currently, there are four widely used descriptors: (a) Atom Centered Symmetry Function (ACSF), (b) Embedded Atom Density (EAD), (c) SO4 Bispectrum Components, and (d) Smooth SO3 Power Spectrum. The dataset to train this neural network will encompass the possible C-B-N or C-Mo configurations representative of actual atomistic configurations of the respective carbon contaminated surfaces under ion bombardment and will be obtained from Density Functional Theory (DFT) calculations. This MLIAP is envisioned to have an accuracy close to DFT but with the computational speed of MD simulations.

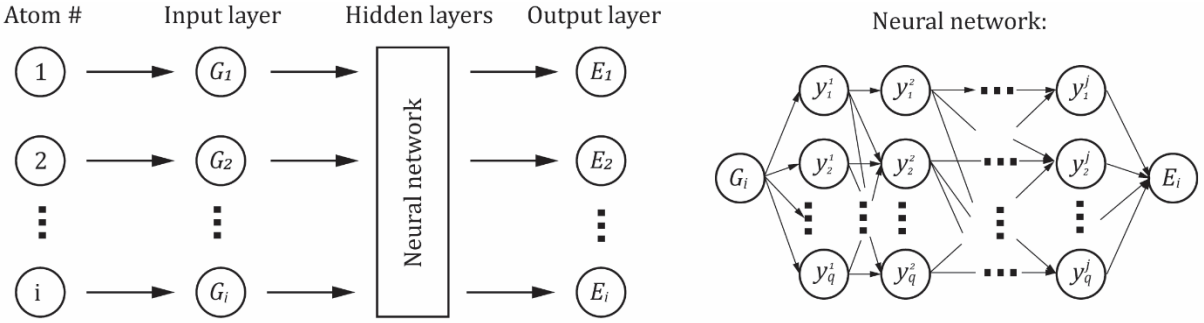

Figure 6.7: Schematic representation of a neural network potential for a given configuration. Coordinates of the atoms are used to compute the environment descriptors, which serve as an input for the neural network, with output as the energy of the atoms.

I evaluate the feasibility of using Neural Network as the basis for the MLIAP by training on a carbon-carbon systems. Here for ease of implementation in LAMMPS, a Smooth SO3 Power Spectrum is chosen as a start to train the carbon-carbon systems [21]. This descriptor is derived by taking the auto correlation of the atomic neighbor density function through expansions on the 2-sphere and a radial bases on a smoothened atomic neighbor density function. In addition, I augment

this descriptor to account for longer-range van der Waals forces. In particular, the energy of an atom α can be decomposed to a long-range and a short-range interaction.

$$E_\alpha = E_\alpha^{long} + E_\alpha^{short} \tag{6.2}$$

where E_α^{long} is a theoretically motivated r^{-6} term as in the Leonard-Jones potential.

$$E_\alpha^{long} = -8.3427 \sum r^{-6} S(r) \tag{6.3}$$

Here $S(r)$ is a switching function to ensure the function monotonically decreases from one to zero over the switch off range. The framework for this approach is proposed by [21].

The training dataset is based on the MD-generated atomistic structure of graphite obtained under Xe ion bombardment (Chapter 3). For one data structure, I subdivide each atomic structure at the end of the Xe bombardment process into smaller simulation cells for up to 60 atoms per simulation cell. The minimized energy of each of these simulation cells, along with the coordinate information of the atoms are used for the training of the neural network. Figure 6.8 shows the training process by comparing of the MD and neural network (NN) energies of the structures in the training set (2781 structures) with increasing number of epoch. This neural network has three layers, each with 50 nodes. We choose the commonly used hyperbolic tangent function as the nonlinear activation function. To test the accuracy of our MLIAP, we used the newly trained MLIAP to simulate a xenon bombardment on a multilayer graphene heating at 600K. Even though the multilayer graphene can retain the expected structure, the bombardment process causing the instability in the sputtering process, leads to inaccurate sputtering yield. *Further work is required to resolve this issue, including a more dedicated training dataset for surface absorption, defects, and growth specifically tailored for the carbon deposition simulations.*

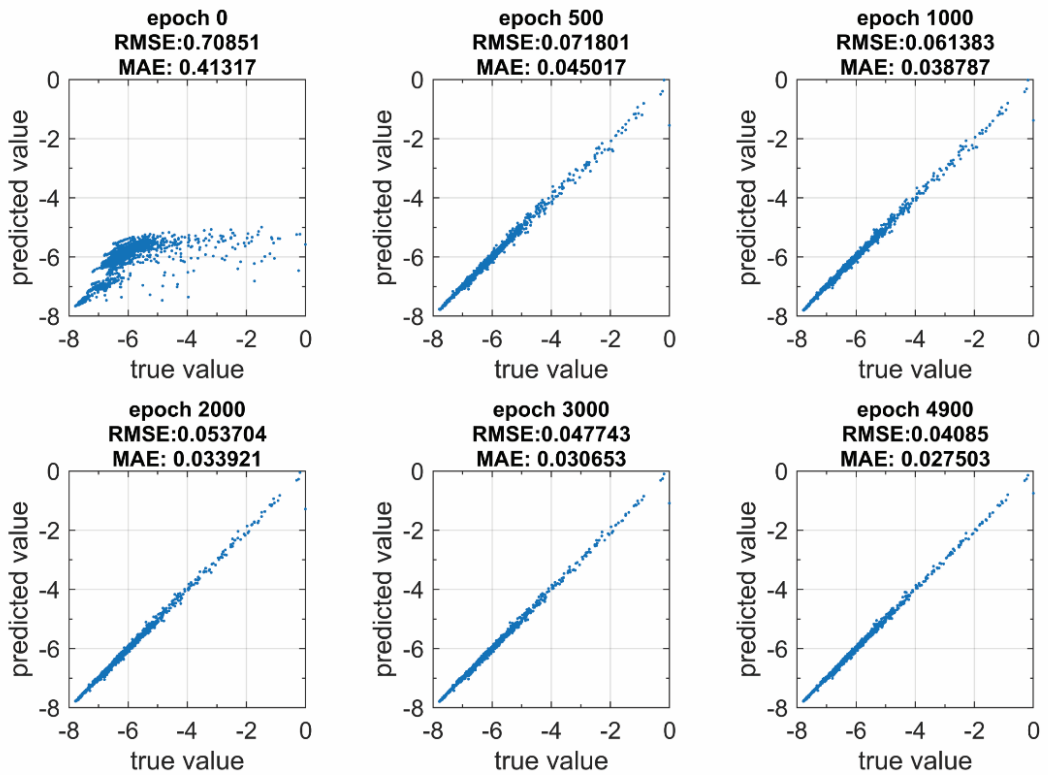

Figure 6.8: Comparing of the MD (true) and neural network (predicted) energies of the structures in the training set with increasing number of epoch

The second possible approach is to bypass IAP altogether and to instead construct the Potential Energy Surface (PES) governing the interaction between the deposited carbon atom and the target surface atoms of interest, and in turn utilize the PES to run Classical Dynamics. PES describes the energy landscape of the system. Specific to carbon contamination modeling, PES would represent the potential energy, $U_{PES}(x,y,z)$, of the carbon adatom, located at (x,y,z), interacting with the substrate [22–24]

$$U_{PES}(x,y,z) = U_t(x,y,z) - U_s - U_a \tag{6.4}$$

where U_t is the total potential energy of the carbon-substrate system, and U_s, U_a represent the potential energy of the substrate, and the incoming carbon atom respectively (Fig. 6.9a). PES can be constructed via Density Function Theory (DFT), without the need of any IAP in MD simulations. However, DFT is highly computational expensive so only a sparse number of 3-

dimensional U_{PES} can reasonably be obtained. Interpolating between these data points is possible through Machine Learning, utilizing either a Gaussian Process or Neural Network approach. Subsequently, the force field, F_{C-s}, representing the force acting on the deposited carbon atom by the substrate can be calculated as

$$F_{C-s} = -\frac{\partial U}{\partial X_i}, X = [x, y, z] \tag{6.5}$$

Classical dynamics [25,26] can then be performed to track the trajectory, $X(t)$, of a single deposited carbon atom over time t. This process is repeated to obtain a full MD simulation of the carbon deposition process, and in turn to construct the carbon absorptivity, c_s of the targeted substrate once a sufficient amount of statistics is obtained. To demonstrate this, I have generated

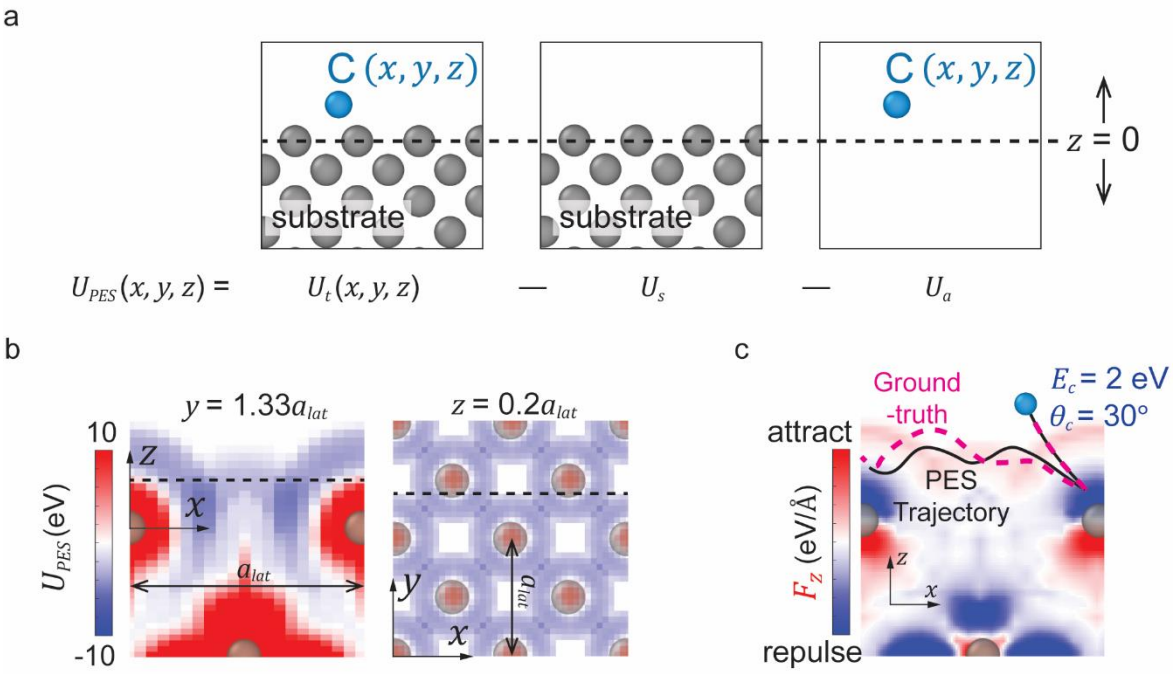

Figure 6.9: (a) Schematic demonstrating the calculation of the potential energy surface, U_{PES} between the substrate atoms (grey) the adatom-C (blue) located at (x, y, z) where $z = 0$ is the substrate's surface. **(b)** Two slices of a 3D PES between C and Si substrate with the (100) crystal planes oriented in the surface normal direction. **(c)** Comparing the trajectory of a deposited carbon with $E_c = 2$ eV and $\theta_C = 30°$ between direct MD calculations (pink) and from classical dynamics (black)using the PES in (b). Background shows the slice of the force field, F_z, computed from the PES in (b) using equation (6.5).

U_{PES} with C on a frozen Silicon substrate using MD simulations with Tersoff Si-C potential [27] with two representative PES cross section shown in Fig. 6.9b. I then deposited a carbon atom is incidence energy $E_C = 2$ eV and $\theta_C = 30°$ and computed its trajectory, particularly the position $X(t)$, and velocity $\dot{X}(t)$, using the velocity-Verlet method [28] from Newton's second law

$$m_C \ddot{X}(t) = F_{C-s}(X(t)) \tag{6.6}$$

where $m_C = 12$ (amu) as mass of carbon atom. Fig. 6.9c shows that the trajectory computed using the PES, overlay on top of the force field F_{C-s}, is almost the same with the ground-truth trajectory generated from MD simulations.

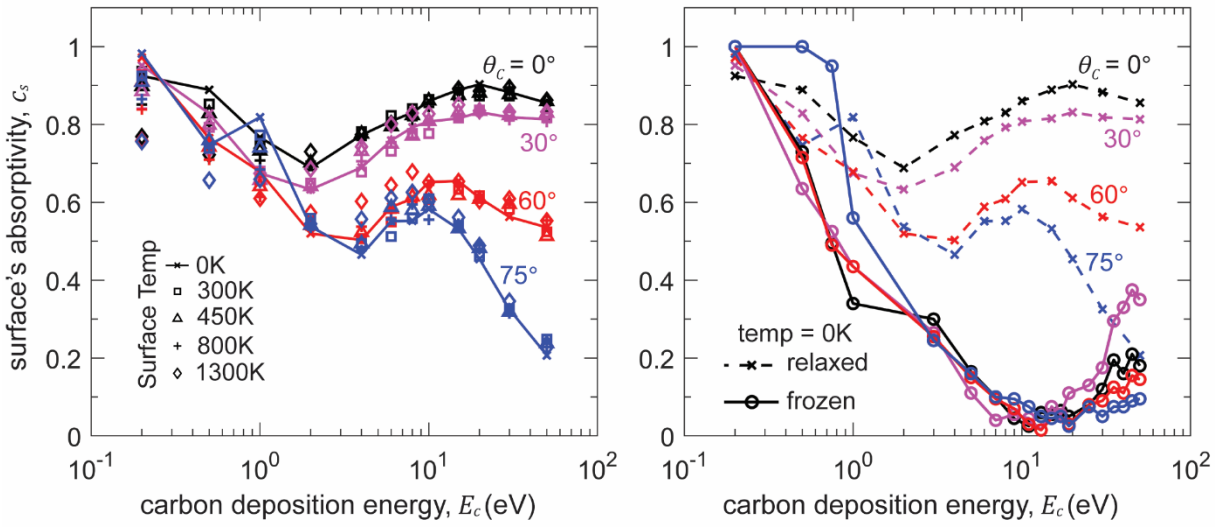

Figure 6.10: Surface absorptivity, c_s as a function of carbon deposition energy. E_C for C on Si substrate at various incidence angle θ_C (lines) (a) for different surface temperature (symbols) and (b) between relaxed (bond reordering – dash line) and frozen (fix bonds – solid line) Si substrate during the deposition process. The results are computed based on MD simulations following a similar approach discussed in Section 6.3.1.

However, one limitation here is that the substrate's surface was frozen in place throughout the minimization process when the PES was computed. As a result, this approach neglects the effects of substrate temperature and possible bond reordering of the substrates (relaxed substrate). Through MD simulations, the substrate's temperature appears to have minimal effect on the carbon

absorptivity, c_s, across a range of carbon deposition energies, E_C and angles, θ_C (Fig. 6.10a). However, there are significant differences in c_s between a frozen and relaxed substrate, especially at high E_C (Fig. 6.10b). In fact, c_s appears to be independent from θ_C for a frozen substrate. A possible approach to overcome this problem is to train a Convolution Neural Network (CNN) that takes 3D PES, represented as an 3D image, along with E_C, θ_C, and the substrate's properties such as bond strength or atomic density as inputs to predict the substrate's absorptivity c_s. The training and testing dataset can be generated from LJ potential, following a similar procedure in Chapter 5, with the validation dataset from other potentials. The input to this CNN will then be the PES from DFT, while the output will be the c_s of the corresponding substrate material.

6.4 References

[1] W. Eckstein, R. Preuss, New fit formulae for the sputtering yield, Journal of Nuclear Materials 320 (2003) 209–213. https://doi.org/10.1016/S0022-3115(03)00192-2.

[2] J.E. Polk, A critical review and meta-analysis of xenon-on-carbon sputter yield data, Journal of Applied Physics 135 (2024) 040701. https://doi.org/10.1063/5.0180720.

[3] J.T. Yim, Yim, J.T. (2017). A Survey of Xenon Ion Sputter Yield Data and Fits Relevant to Electric Propulsion Spacecraft Integration., in: 35th International Electric Propulsion Conference, 2017.

[4] J.F. Ziegler, M.D. Ziegler, J.P. Biersack, SRIM – The stopping and range of ions in matter (2010), Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 268 (2010) 1818–1823. https://doi.org/10.1016/j.nimb.2010.02.091.

[5] J.F. Ziegler, J.P. Biersack, The Stopping and Range of Ions in Matter, in: D.A. Bromley (Ed.), Treatise on Heavy-Ion Science: Volume 6: Astrophysics, Chemistry, and Condensed Matter, Springer US, Boston, MA, 1985: pp. 93–129. https://doi.org/10.1007/978-1-4615-8103-1_3.

[6] M. Zhang, S. Kim, P.Y. Lu, M. Soljačić, Deep Learning and Symbolic Regression for Discovering Parametric Equations, (2023). http://arxiv.org/abs/2207.00529 (accessed August 23, 2024).

[7] T. Mundhenk, M. Landajuela, R. Glatt, C.P. Santiago, D. faissol, B.K. Petersen, Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2021: pp. 24912–24923. https://papers.nips.cc/paper_files/paper/2021/hash/d073bb8d0c47f317dd39de9c9f004e9d-Abstract.html (accessed August 23, 2024).

[8] M. Cranmer, D. Tamayo, H. Rein, P. Battaglia, S. Hadden, P.J. Armitage, S. Ho, D.N. Spergel, A Bayesian neural network predicts the dissolution of compact planetary systems,

Proceedings of the National Academy of Sciences 118 (2021) e2026053118. https://doi.org/10.1073/pnas.2026053118.
[9] D. Angelis, F. Sofos, T.E. Karakasidis, Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives, Arch Computat Methods Eng 30 (2023) 3845–3865. https://doi.org/10.1007/s11831-023-09922-z.
[10] S. Udrescu, M. Tegmark, AI Feynman: A physics-inspired method for symbolic regression, (2020). https://doi.org/10.1126/sciadv.aay2631.
[11] G. Hobler, D. Kovač, Dynamic binary collision simulation of focused ion beam milling of deep trenches, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 269 (2011) 1609–1613. https://doi.org/10.1016/j.nimb.2010.12.076.
[12] R.B. Lobbia, J.E. Polk, R.R. Hofer, V.H. Chaplin, B. Jorns, Accelerating 23,000 hours of Ground Test Backsputtered Carbon on a Magnetically Shielded Hall Thruster, in: AIAA Propulsion and Energy 2019 Forum, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-3898.
[13] M. Tartz, E. Hartmann, R. Deltschew, H. Neumann, Experimental validation of a grid erosion simulation, in: 35th Joint Propulsion Conference and Exhibit, American Institute of Aeronautics and Astronautics, 1999. https://doi.org/10.2514/6.1999-2860.
[14] G. Williams, T. Haag, J. Foster, J.V. Noord, S. Malone, T. Hickman, M. Patterson, Analysis of the Pyrolytic Graphite Ion Optics Following the 2000-hour Wear Test of the HiPEP Ion Thruster, in: 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, American Institute of Aeronautics and Astronautics, 2006. https://doi.org/10.2514/6.2006-5005.
[15] N. Kametani, M. Nakamura, K. Yashiro, T. Takaki, Investigating residual stress evolution in the deposition process of diamond-like carbon film through molecular dynamics, Computational Materials Science 209 (2022) 111420. https://doi.org/10.1016/j.commatsci.2022.111420.
[16] H.U. Jäger, K. Albe, Molecular-dynamics simulations of steady-state growth of ion-deposited tetrahedral amorphous carbon films, Journal of Applied Physics 88 (2000) 1129–1135. https://doi.org/10.1063/1.373787.
[17] T. Muramoto, T. Hyakutake, M. Nishida, T. Kenmotsu, Molecular dynamics study on carbon film deposition, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 269 (2011) 1752–1754. https://doi.org/10.1016/j.nimb.2010.11.015.
[18] H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–72. https://doi.org/10.1145/174462.156635.
[19] V.L. Deringer, G. Csányi, Machine learning based interatomic potential for amorphous carbon, Phys. Rev. B 95 (2017) 094203. https://doi.org/10.1103/PhysRevB.95.094203.
[20] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A.V. Shapeev, A.P. Thompson, M.A. Wood, S.P. Ong, Performance and Cost Assessment of Machine Learning Interatomic Potentials, J. Phys. Chem. A 124 (2020) 731–745. https://doi.org/10.1021/acs.jpca.9b08723.
[21] M. Wen, E.B. Tadmor, Hybrid neural network potential for multilayer graphene, Phys. Rev. B 100 (2019) 195419. https://doi.org/10.1103/PhysRevB.100.195419.
[22] G. Fratesi, Potential energy surface of alkali atoms adsorbed on Cu(001), Phys. Rev. B 80 (2009) 045422. https://doi.org/10.1103/PhysRevB.80.045422.

[23] M.A. Di Césare, H.F. Busnengo, W. Dong, A. Salin, Role of dynamic trapping in H2 dissociation and reflection on Pd surfaces, The Journal of Chemical Physics 118 (2003) 11226–11234. https://doi.org/10.1063/1.1575208.
[24] R.A. Olsen, H.F. Busnengo, A. Salin, M.F. Somers, G.J. Kroes, E.J. Baerends, Constructing accurate potential energy surfaces for a diatomic molecule interacting with a solid surface: H2+Pt(111) and H2+Cu(100), The Journal of Chemical Physics 116 (2002) 3841–3855. https://doi.org/10.1063/1.1446852.
[25] V. Morón, L. Martin-Gondre, C. Crespos, P. Larregaray, P. Gamallo, R. Sayós, Classical dynamics study of atomic oxygen over graphite (0001) with new interpolated and analytical potential energy surfaces, Computational and Theoretical Chemistry 990 (2012) 132–143. https://doi.org/10.1016/j.comptc.2012.01.030.
[26] A. Groß, Reactions at surfaces studied by ab initio dynamics calculations, Surface Science Reports 32 (1998) 291–340. https://doi.org/10.1016/S0167-5729(98)00008-9.
[27] J.H. Los, J.M.H. Kroes, K. Albe, R.M. Gordillo, M.I. Katsnelson, A. Fasolino, Extended Tersoff potential for boron nitride: Energetics and elastic properties of pristine and defective h-BN, Phys. Rev. B 96 (2017) 184108. https://doi.org/10.1103/PhysRevB.96.184108.
[28] M. Alducin, R. Díez Muiño, H.F. Busnengo, A. Salin, Low sticking probability in the nonactivated dissociation of N2 molecules on W(110), The Journal of Chemical Physics 125 (2006) 144705. https://doi.org/10.1063/1.2355672.

APPENDIX A: CODE

A.1 MATLAB Code for 1D Monte Carlo for Surface Morphology Evolution under Ion Bombardment

Main Input and Run Code

```
%% General Note and Assumptions
% yield (atoms/ion) is treated as probability
% yield angle distribution is also treated as probability but independently
% from the yield
% Xenon absorption rate is a constant (also treated as probability)
% -> does not account for pocket of Xenon gas got escape
% for calculation purpose : angle wise: COUNTER CLOCKWISE IS POSITIVE

F = findall(0,'type','figure','tag','TMWWaitbar');
delete(F)
close all

clear;
clc;
%% Input
=========================================================================================

% simulation type input
yield_mode = 'interpolate'; % interpolate, eckstein
distribution_mode = 'MD'; % MD, gaussian
pertube = false;
pertube_variation = 10; %nm
redeposition = true; % only applicable for version 13_cstep_parallel
secondary_sputtering = true; % only applicable for version 12+
second_sput_distribution = 'cosine';

% structure input
wave_num = 1;
model = 'wavecomb';
%plotheight = 600; %nm
if strcmp(model,'semicircle')
    lambda = 2000; %nm %simulation box size
    surface_span = 1600; %nm
    curve_shift = false;
    if curve_shift == true
        dis_slope = 40;  %deg, has to be less than 90 degree
    end
elseif strcmp(model,'wave')
    lambda = 2000; %nm
    H = 1500; %nm
elseif strcmp(model,'wavecomb')
    surface_input = @(x) 1/2*(0.75*sin(x) + 0.45*cos(2*x) + 0.35*cos(0.5*x));
    min_H = 1000;
    based_period = 2000;
    lambda = 2*based_period; %nm - make sure the lambda represents the largest (a*x),
lambda = a*lambda
elseif strcmp(model,'trap/tri')
    %       |<--c1-->/--c2--\            ^
    %       |      /         \           |
    %       |    /            \          h
    %       |  /               \         |
    % --a--|/                  \---b--   v
    % total length lambda
    % a + b + c1 + c2 < lambda
    lambda = 2000; %nm
```

```
    a = 200; %nm
    b = 800; %nm
    c1 = 600; %nm
    c2 = 0; %nm
    H = 400; %nm
    if (a+b+c1+c2) > lambda
        error('parameters error for trap/tri model')
    end
    % Added to allow multiple repeat
    count_repeat = 2;
elseif strcmp(model,'user')
    direc = 'D:\xxxxxx';
    fileid = 'wave_series';
    step_bombard = 0;
elseif strcmp(model,'restart')
    re_count = 1;
    step_bombard = -1; %choose the most recente step
    nilist = [198:200:2999999]*10^4; % search for file with this frequency
end
shift_test = 0; %CAUTION, this is more for debugging, make sure BCs still satisfied
dx = 20; %nm
buffer_depth = 2000; %nm
buffer_width = 0; %wave_num
periodic = true;
restart_cond = false;

width = 5; %nm

% simulation condition input
sputter_save_count = 1000000; % save sputtered value after xx bombardment
wait_update_freq = 500000; % change this number accordingly to the num_bombardment variable
below
plot_update_freq = 20;
restart_file_name = 'wave_combination_series';
plot_path = false;

% bombard ion information and controls parameters
phi = 60; %deg :: always moving toward +x and -y direction
phi_variation = 0; % ratio of variation in phi
tt_eV = 4; %2- 200 eV, 3- 350 eV, 4- 500 eV, 5- 1000 eV, 6 - 1500 eV, 7 - 2000 eV, 8 - 700
eV
eVlist = [100,200,350,500,1000,1500,2000,700]; %eV, need to match above
num_bombardment = 1*10^10;
atom_p_v = 0.08; %atom per Angstrom^3
current_density = 3.85*10^(19); %ion/m^2-s

% add since version 7, this in a way is to force termination
height_limit = 10000; %nm

% input for parallel settings
num_consecutive = 10000; % number of deposit before a surface height update occurs

%%
================================================================================================
=====

if sputter_save_count < num_consecutive
    warning('sputter_save_count cannot be less than num_consecutive, sputter_save_count is
now reset to num_consecutive')
    sputter_save_count = num_consecutive;
end

if phi == 0
    dhi = 5;
else
    dhi = dx/(2*tand(phi+phi_variation*phi));
end
```

```
% create the folder for the simulation
Folder = append(restart_file_name);
if not(isfolder(Folder))
    mkdir(Folder)
else
    warning('folder already exist')
end
curFolder = pwd;
input_file = append(curFolder,'/',restart_file_name,'/input');
output_file = append(curFolder,'/',restart_file_name,'/output');
restart_file_name = append(curFolder,'/',restart_file_name,'/',restart_file_name);

% get yield and distribution information
num = 1;
[yield_spline, md_y, vel_dis, angle_asso, en_dis, resput_prob, resput_dis] = ...
    get_per_element_sputter_info(distribution_mode,yield_mode,tt_eV,secondary_sputtering);

% Check for input error -variation is added since version 14th
if phi-phi_variation*phi < 0 || phi+phi_variation*phi > 89
    error('phi out of range');
end
if dhi*tand(phi+phi_variation*phi) > dx/2
    error('ion moves more than half a bin at a time, please reduce dhi');
end
if dhi > dx
    error('dhi is too big, please reduce them')
end

phil.angle = phi;
phil.variation = phi_variation;
phi = phil;

% using porosity information to calculate height per atom -- needs to give
% some additional thoughts on this as of version 12
h_p_atom = 1/(atom_p_v*10^3*dx*width); %nm

%% Set up store data
dt_save = floor(num_bombardment/sputter_save_count);
sput_atom_record = zeros(dt_save,2); %time

% Set up the simulation domain and bin size
if strcmp(model,'user')
    fileload = append(direc,fileid,'/',fileid,'_',num2str(step_bombard),'.mat');
    h_bin = load(fileload).x;
    lambda = h_bin(end,2)-h_bin(1,2);
    dx = h_bin(2,2)-h_bin(1,2);
    x = h_bin(:,2)';
    wave_num_mod = wave_num + buffer_width*2;
    %h_bin(:,3) = flipud(h_bin(:,3));
elseif strcmp(model,'restart')
    if step_bombard == -1
        for iiii = 1:1:size(nilist,2)
            fileload = append(restart_file_name,'_',num2str(nilist(iiii)),'.mat');
            if ~isfile(fileload)
                break
            end
        end
        step_bombard = nilist(iiii-1);
    end
    fileload = append(restart_file_name,'_',num2str(step_bombard),'.mat');
    h_bin = load(fileload).h_bin;
    lambda = h_bin(end,2);
    dx = h_bin(2,2)-h_bin(1,2);
    x = h_bin(:,2)';
    wave_num_mod = wave_num + buffer_width*2;
    %h_bin(:,3) = flipud(h_bin(:,3));
```

```
    %h_bin(:,3) = flipud(h_bin(:,3));
else
    wave_num_mod = wave_num + buffer_width*2;
    bin_id = 1:1:ceil(lambda*wave_num_mod/dx)+1;
    step_bombard = 0;
    x = linspace(0,lambda*wave_num_mod,size(bin_id,2));
    if x(2) - x(1) ~= dx
        dx = x(2) - x(1);
        warning('cannot have even time step with the prescribed dx, dx has been adjusted to
%g', dx);
    end
    %x = 0:dx:lambda*wave_num_mod;
    bin_id = 1:1:size(x,2);
    h_bin = zeros(size(bin_id,2),3); %node id, node x-value, node height
    h_bin(:,1) = bin_id';
    h_bin(:,2) = x'-buffer_width*lambda;
    if strcmp(model,'wave')
        h_bin(:,3) = -H/2.*cos(2*pi*(x'/lambda))+H/2+buffer_depth;
        %h_bin(:,3) = H/2.*cos(2*pi*(x'/lambda))+H/2+buffer_depth;
        %h_bin(:,3) = -H/2.*sin(2*pi*(x'/lambda))+H/2+buffer_depth;
    elseif strcmp(model,'wavecomb')
        h_bin(:,3) = min_H*surface_input(2*pi*(x'/based_period));
        h_bin(:,3) = (h_bin(:,3) - min(h_bin(:,3))) + buffer_depth;
        if ismembertol(h_bin(1,3),h_bin(end,3))
            h_bin(1,3) = h_bin(end,3);
        else
            error('periodic conditions not satisfied')
        end
    elseif strcmp(model,'semicircle')
        ksizek = surface_span/lambda;
        if curve_shift == true
            if dis_slope > 90
                error('dis_slope has to be less than 90 degree');
            end
            downward_shift = (surface_span/2)*tand(90 - dis_slope);
            lambdan =  ((surface_span/2)/cosd(90 - dis_slope))*2;
        else
            lambdan = surface_span;
            downward_shift = 0;
        end
        for i = 1:size(h_bin,1)
            if x(i) < (1-ksizek)/2*lambda || x(i) > (ksizek+(1-ksizek)/2)*lambda
                h_bin(i,3) = buffer_depth;
            else
                h_bin(i,3) = (lambdan^2/4-(x(i)-lambda/2).^2).^(1/2)-
downward_shift+buffer_depth;
            end
        end
        %h_bin(:,3) = smoothdata(h_bin(:,3),'gaussian');
        %h_bin(:,3) = flipud(h_bin(:,3));
    elseif strcmp(model,'trap/tri')
        h_bin((x > a & x <= a+c1),3) = (x(x > a & x <=(a+c1))-a)*(H/c1);
        h_bin((x > a+c1 & x <= a+c1+c2),3) = H;
        h_bin((x > a+c1+c2 & x <= lambda-b),3) = (lambda - x(x > a+c1+c2 & x <= lambda-b) -
b)*(H/(lambda-b-a-c1-c2));
        if count_repeat > 1
            windowq = floor(lambda/count_repeat/dx);
            h_bin_1peak = h_bin(1:windowq,:);
            for ttt = 2:count_repeat
                h_bin(windowq*(ttt-1)+1:windowq*ttt,3) = h_bin_1peak(:,3);
            end
        end
        h_bin(:,3) = h_bin(:,3) + buffer_depth;
    end
end
if shift_test > 0
    [~, idd] = min(abs(h_bin(:,2)-shift_test));
```

```
    h_bin(:,3) = circshift(h_bin(:,3), idd);
    if h_bin(1,3) ~= h_bin(end,3)
        warning('BCs does not satisfied, the results might not be correct, applied
autofix')
        if strcmp(model,'wavecomb')
            error('BCs does not satisfied, check lambda to make sure it satisfied')
        end
        h_bin(1,3) = h_bin(end,3);
    end
end
if pertube == true
    height_change = unifrnd(-pertube_variation,pertube_variation,size(h_bin,1),1);
    h_bin(:,3) = h_bin(:,3) + height_change;
    h_bin(1,3) = h_bin(end,3);
end

plotheight = max(h_bin(:,3)) + 200;
set(figure(num),'color','w')
plot(h_bin(:,2),h_bin(:,3),'--b','linewidth',1.2)
xlabel('x (nm)')
ylabel('h(x) (nm)')
axis equal
ylim([0 plotheight])
xlim([0 inf])
hold on
num = num+1;

% Save the necessary variables as structure for input file
input_struc.vel_disc = vel_dis;
input_struc.height_limit = height_limit; input_struc.lambda = lambda; input_struc.x = x;
input_struc.wave_num = wave_num;
input_struc.wave_num_mod = wave_num_mod; input_struc.dhi = dhi; input_struc.phi = phi;
input_struc.dx = dx;
input_struc.yield_spline = yield_spline; input_struc.angle_asso = angle_asso;
input_struc.eVrun = eVlist(tt_eV);
input_struc.h_p_atom = h_p_atom; input_struc.yield_mode = yield_mode;
input_struc.redeposition = redeposition;
input_struc.en_dis = en_dis; input_struc.resput_prob = resput_prob; input_struc.resput_dis
= resput_dis;
save(input_file,'-struct','input_struc');

%% Run the main function
num_run = floor(num_bombardment/num_consecutive);
num_ion = 0;
atom_out = 0;
m=1;
for i = 1:num_run
    if mod(i,num_run/wait_update_freq) == 0
        toc;
        save([restart_file_name,'_',num2str(num_ion+step_bombard),'.mat'],'h_bin');
        if strcmp(model,'restart')

save([restart_file_name,'_','sputter_record',num2str(re_count),'.mat'],'sput_atom_record');
        else
            save([restart_file_name,'_','sputter_record.mat'],'sput_atom_record');
        end
    end
    [atom_out, h_bin] = run_MC_v14(input_file,num_consecutive,h_bin,atom_out);
    num_ion = i*num_consecutive;
    if mod(num_ion,num_bombardment/dt_save) == 0
        sput_atom_record(m,1) = num_ion;
        sput_atom_record(m,2) = atom_out; %store atom out and properties
        m=m+1;
    end
end
```

Main MC Algorithm Function

```
function [atom_out,h_bin] = run_MC_v14(filename,num_bombardment,h_bin,atom_out)

    data = load(filename);
    vel_dis = data.vel_disc; lambda = data.lambda; x = data.x; wave_num = data.wave_num;
wave_num_mod = data.wave_num_mod;
    dhi = data.dhi; phi_cond = data.phi; dx = data.dx; yield_spline = data.yield_spline;
angle_asso = data.angle_asso;h_p_atom = data.h_p_atom;
    height_limit = data.height_limit; eVrun = data.eVrun; en_dis = data.en_dis;
    yield_mode = data.yield_mode; resput_prob = data.resput_prob; resput_dis =
data.resput_dis; redeposition = data.redeposition;

    %% Run the Monte Carlo bombardment
    % initialized statistic
    %bin_deposit_statistic = zeros(num_bombardment,1);

    % start simulation
    hc_store = cell(num_bombardment,1);

    parfor i = 1:num_bombardment

        xmin = min(h_bin(:,2));
        xmax = max(h_bin(:,2));

        height_id_change = [];
        n_i_c = 1;
        % set up deposition and capture zone based on maximum surface height --
        max_h = max(h_bin(:,3));
        if max_h > height_limit
            error('surface peak are too high')
        end
        h_capture = max_h + dhi + dhi/100; %to ensure the first step is not the bad one
        h_deposit = h_capture;
        %bin_ion = randi([bin_0 bin_lw]);
        %x_ion = h_bin(bin_ion,2) + dx/10;
        x_ion = lambda*wave_num_mod*rand();
        %x_ion = 9.826829e+02;% debug

        x_ion_start = x_ion;
        x_ion_debug = x_ion_start;
        h_ion = h_deposit;
        h_ion_start = h_ion;
        %x_ion = 552;
        %xi_st(i) = x_ion;

        % addition since version 14, created a distribution of incidence
        % angle
        phi_angle = phi_cond.angle;
        phi_variation = phi_cond.variation;
        % uniform variation
        phi = (phi_variation*phi_angle*2).*rand() + (phi_angle-phi_variation*phi_angle);

        % New algorithm to identify contact -- change since version 9, this
        % would retire the need for identify bin number
        while true
            h_ion = h_ion - dhi;
            if h_ion < 0
                parsave('last_structure',h_bin);
                warning('h_ion can never be negative, intersection never occurs, check code
or add more height if needed');
                error('x_ion_debug of %d with phi of %d',x_ion_debug, phi);
            end
            if dhi*tand(phi) > dx
```

```
                error('ion moves more than one bin at a time, please reduce dhi');
            end
            x_ion = x_ion + dhi*tand(phi);

            % identify the region where x_ion belongs
            xreg = x;
            hreg = h_bin(:,3)';
            [~, iddd] = mink(abs(xreg - x_ion),2); %
            iddd = sort(iddd); % since x is always increasing
            surf_x = xreg(iddd);
            if surf_x(1) >= surf_x(2)
                error('needs to order correctly and not the same xpoint');
            end
            surf_height = hreg(iddd);
            if size(surf_x) > 2
                error('cannot have three points')
            end
            % find the intersection between the ion path and the slope of
            % this region
            [x_ion_impact, h_ion_impact] = line_intersection([x_ion_start, h_ion_start,
x_ion, h_ion],...
                [surf_x(1), surf_height(1), surf_x(2), surf_height(2)]);

            % check to see wheter the ion_impact is within the region
            if x_ion_impact > surf_x(1) && x_ion_impact <= surf_x(2)
                bin_slope_local = (surf_height(2)-surf_height(1))/(surf_x(2)-surf_x(1));
                %bin_slope_local = (h_ion_impact - h_bin(bin_ion,3))/(x_ion_impact -
h_bin(bin_ion,2));
                break;
            end
            if x_ion >= xmax
                x_ion = x_ion - xmax + xmin;
            end
            if x_ion < xmin || x_ion > xmax
                error('x_ion out of zone, check codes');
            end
            x_ion_start = x_ion;
            h_ion_start = h_ion;
        end

        bin_angle_local = -acotd(bin_slope_local);
        if bin_angle_local < 0 %equivalent to be on the right side of the norm vector
            bin_normal_angle_local = bin_angle_local + 90; %normal vector rotate counter
clockwise by 90 deg
        else %equivalent to be on the left side of the norm vector
            bin_normal_angle_local = bin_angle_local - 90; % normal vector rotate clockwise
by 90 degree
        end

        % step 2: identify the local ion bombardment
        if phi - bin_normal_angle_local > 0
            ion_rel = 1; %left hit
        else
            ion_rel = -1; %right hit
        end
        ion_angle_local = abs(phi - bin_normal_angle_local);
        if ion_angle_local < 0 || ion_angle_local > 90
            error('ion local angle exceeds the reasonable range with x_ion at %d and phi at
%d', x_ion_debug,phi)
        end
        % step 3: using MD results + probability to get # of atom out and direction
        if strcmp(yield_mode,'eckstein')
            atom_ion = yield_spline(eVrun, ion_angle_local);
        else
            atom_ion = ppval(yield_spline, ion_angle_local);;
        end
        %atom_ion = ppval(yield_spline, ion_angle_local);
```

```
    atom_num = floor(atom_ion);
    next_prob = atom_ion - atom_num; % probability for the next ion
    if next_prob < 0
        error('sthg wrong, probability cannot be less than 0')
    end
    if (next_prob >= rand())
        atom_num = atom_num + 1;
    end
    % for each sputtered atom, reduce the height of the bin, track where
    % the atom going
    %atom_num = 1; % COMMENT OUT IN ACTUAL SIMULATION !!!!!!!!!!!!!!

    for ii = 1:atom_num
%             check(ccc)=h_bin(bin_ion);
%             ccc=ccc+1;
        %[~, iddd] = min(abs(x - x_ion_impact));
        %bin_ion = h_bin(iddd,1);
        height_id_change(n_i_c,1) = x_ion_impact;
        height_id_change(n_i_c,2) = -1;
        %height_id_change(n_i_c,3) = x_ion_impact;
        n_i_c = n_i_c + 1;
        %h_bin(bin_ion,3) = h_bin(bin_ion,3) - h_p_atom;
        % from ion_angle_local, identify the sputtered distribution
        % citation: adam nieslony :: matlab central based on the
        % inverse of the cdf function
        for j = 1:size(angle_asso,2)
            if ion_angle_local < angle_asso(j)
                break
            end
        end
        dist_low = vel_dis(:,j);
        dist_high = vel_dis(:,j+1);
        ang_low = angle_asso(j-1);
        ang_high = angle_asso(j);
        yy = ang_high - ion_angle_local;
        xx = ion_angle_local - ang_low;
        dist_angle_sput = (dist_low*yy + dist_high*xx)/(xx+yy);
        % using this distribution to choose sputtered atom
        px = vel_dis(:,1);
        p = dist_angle_sput;
        %px = px(:);
        p=p./trapz(px,p);
        cdfp = cumtrapz(px,p);
        % finding the parts of cdf parallel to the X axis
        ind=[true; not(diff(cdfp)==0)];
        % and cut out the parts
        cdfp=cdfp(ind);
        p=p(ind);
        px=px(ind);
        % generating the uniform distributed random numbers
        uniformDistNum=rand(1);
        % and distributing the numbers using cdf from input pdf
        phi_atom_local=interp1(cdfp,px,uniformDistNum,'linear');
        if en_dis ~= -1000
            dist_low = en_dis(:,j);
            dist_high = en_dis(:,j+1);
            dist_energy_sput = (dist_low*yy + dist_high*xx)/(xx+yy);
            % using this distribution to choose sputtered atom
            px = en_dis(:,1);
            p = dist_energy_sput;
            %px = px(:);
            p=p./trapz(px,p);
            cdfp = cumtrapz(px,p);
            % finding the parts of cdf parallel to the X axis
            ind=[true; not(diff(cdfp)==0)];
            % and cut out the parts
            cdfp=cdfp(ind);
```

```
                p=p(ind);
                px=px(ind);
                % generating the uniform distributed random numbers
                uniformDistNum=rand(1);
                % and distributing the numbers using cdf from input pdf
                energy_atom=interp1(cdfp,px,uniformDistNum,'linear');
            end
            phi_atom_local = phi_atom_local*ion_rel*(-1); %adjust for left and right side
            % bring back angle to global scale
            phi_atom = phi_atom_local + bin_normal_angle_local;
            phi_atom_o = phi_atom;
            if phi_atom >= 90
                hu = -1; %identify condition for height increase or decrease
                phi_atom = 180 - phi_atom;
            elseif phi_atom <= -90
                hu = -1;
                phi_atom = -180 - phi_atom;
            else
                hu = 1;
            end
            x_atom = x_ion_impact;
            h_atom = h_ion_impact;
            % if goes left, the starting location is a lower id
            % if goes right, the starting location is a higher id
            % advance the atom until contact
            initiate = true;
            deposit_cond = false;
            %x_atom_debug = x_atom;
            if redeposition == true
                while h_atom < h_capture
                    % advance the atom
                    cstep = dx/2;
                    if abs(phi_atom - 90) < 10^(-7)
                        x_atom = x_atom - cstep;
                        motion = 'left';
                    elseif abs(phi_atom + 90) < 10^(-7)
                        x_atom = x_atom + cstep;
                        motion = 'right';
                    else
                        % based on phi_atom, compute reasonable step size
                        if abs(phi_atom) >= atand(dhi/100)
                            dha = abs(dhi/tand(abs(phi_atom)));
                        else
                            dha = dhi;
                        end
                        if dha*tand(phi_atom) > 0
                            motion = 'left';
                            x_atom = x_atom - cstep; %dha*tand(phi_atom);
                        else
                            motion = 'right';
                            x_atom = x_atom + cstep; %dha*tand(phi_atom);
                        end
                        h_atom = h_atom + hu*(cstep/tand(abs(phi_atom))); %height increase
or decrease
                    end
                    if h_atom < 0
                        parsave('last_structure',h_bin);
                        warning('h_atom can never be negative, intersection never occurs,
check code or add more height if needed');
                        error('x_ion_debug of %d with phi of %d and phi atom of
%d',x_ion_debug, phi, phi_atom_o);
                    end
                    if initiate == true
                        initiate = false;
                        if strcmp(motion,'right')
                            [x_atom_start, h_atom_start] = line_intersection(...
                                [x_ion_impact, h_ion_impact, x_atom, h_atom],...
```

```
                            [surf_x(2), surf_height(2), surf_x(2), surf_height(2)+1]);
                            if h_atom_start < surf_height(2)
                                error('wrong initialization of atom2 with x_ion %d, phi %d,
phi_atom %d',...
                                    x_ion_debug, phi, phi_atom_o)
                            end
                        elseif strcmp(motion,'left')
                            [x_atom_start, h_atom_start] = line_intersection(...
                                [x_ion_impact, h_ion_impact, x_atom, h_atom],...
                            [surf_x(1), surf_height(1), surf_x(1), surf_height(1)+1]);
                            if h_atom_start < surf_height(1)
                                error('wrong initialization of atom2 with x_ion %d, phi %d,
phi_atom %d',...
                                    x_ion_debug, phi, phi_atom_o)
                            end
                        end
                        if x_atom_start >= xmax
                            x_atom_start = x_atom_start - xmax + xmin;
                        elseif x_atom_start <= xmin
                            x_atom_start = xmax - (xmin - x_atom_start);
                        end
                        x_atom = x_atom_start;
                        h_atom = h_atom_start;
                    else
                        % identify the region where x_atom belongs
                        xreg = x;
                        hreg = h_bin(:,3)';
                        [~, iddd] = mink(abs(xreg - x_atom),2); %
                        iddd = sort(iddd); % since x is always increasing
                        surf_xa = xreg(iddd);
                        if surf_xa(1) >= surf_xa(2)
                            error('needs to order correctly and not the same xpoint');
                        end
                        surf_heighta = hreg(iddd);
                        if size(surf_xa) > 2
                            error('cannot have three points')
                        end
                        % find the intersection between the ion path and the slope of
                        % this region
                        [x_atom_impact, h_atom_impact] = line_intersection([x_atom_start,
h_atom_start, x_atom, h_atom],...
                            [surf_xa(1), surf_heighta(1), surf_xa(2), surf_heighta(2)]);

                        % check to see wheter the ion_impact is within the region
                        if x_atom_impact > surf_xa(1) && x_atom_impact <= surf_xa(2)
                            deposit_cond = true;
                            break;
                        end
                        if x_atom >= xmax
                            x_atom = x_atom - xmax + xmin;
                        elseif x_atom <= xmin
                            x_atom = xmax - (xmin - x_atom);
                        end
                        if x_atom < xmin || x_atom > xmax
                            error('x_atom out of zone, check codes');
                        end
                        if h_atom < surf_heighta(1) && h_atom < surf_heighta(2)
                            error('dips below with x_ion %d, phi %d, phi_atom %d',
x_ion_debug, phi, phi_atom_o)
                        end
                        x_atom_start = x_atom;
                        h_atom_start = h_atom;
                    end
                end
            end
            if deposit_cond == true
                height_id_change(n_i_c,1) = x_atom_impact;
```

```
                height_id_change(n_i_c,2) = 1;
                n_i_c = n_i_c + 1;
                if en_dis ~= -1000
                    %% add since version 12 -- this one is for secondary
                    % sputtering effect
                    % get the atom2 impact surface normal angle
                    bin_slope_local_atom = (surf_heighta(2)-surf_heighta(1))/(surf_xa(2)-
surf_xa(1));
                    bin_angle_local_atom = -acotd(bin_slope_local_atom);
                    if bin_angle_local_atom < 0 %equivalent to be on the right side of the
norm vector
                        bin_normal_angle_local_atom = bin_angle_local_atom + 90; %normal
vector rotate counter clockwise by 90 deg
                    else %equivalent to be on the left side of the norm vector
                        bin_normal_angle_local_atom = bin_angle_local_atom - 90; % normal
vector rotate clockwise by 90 degree
                    end
                    % calculate the atom impact angle and check the contact information and
make sure it is
                    % reasonable
                    if strcmp(motion,'left')
                        angle_atom2 = 180 - phi_atom_o + bin_normal_angle_local_atom;
                        if phi_atom_o < 0
                            error('left motion cannot have negative angle');
                        end
                        if 180 - phi_atom_o < 0
                            error('this shouldnt be the case');
                        end
                        if hu == 1 && bin_normal_angle_local_atom > 0
                            error('left motion, travel upward, bin normal angle negative')
                        elseif hu == -1
                            if angle_atom2 < 0
                                atom_rel = 'left_hit';
                            else
                                atom_rel = 'right_hit';
                            end
                        elseif hu == 1
                            atom_rel = 'right_hit';
                        else
                            error('has to be one of these cases')
                        end
                    elseif strcmp(motion,'right')
                        angle_atom2 = 180 + phi_atom_o - bin_normal_angle_local_atom;
                        if phi_atom_o > 0
                            error('right motion cannot have positive angle');
                        end
                        if 180 + phi_atom_o < 0
                            error('this shouldnt be the case');
                        end
                        if hu == 1 && bin_normal_angle_local_atom < 0
                            error('right motion, travel upward, bin normal angle positive')
                        elseif hu == -1
                            if angle_atom2 > 0
                                atom_rel = 'left_hit';
                            else
                                atom_rel = 'right_hit';
                            end
                        elseif hu == 1
                            atom_rel = 'left_hit';
                        else
                            error('has to be one of these cases')
                        end
                    end
                    % based on the atom energy (energy_atom) and the atom impact angle
(angle_atom2),
                    % get the probability of resputtering
                    if energy_atom > 150
```

```
                        energy_atom = 149.99;
                    end
                    angle_atom2 = abs(angle_atom2);
                    if angle_atom2 < 0 || angle_atom2 > 90
                        error('atom resputtered local angle exceeds the reasonable range')
                    end
                    [~, id_2] = min(abs(resput_prob(1,:) - energy_atom)); %
                    id_ee = [0;0];
                    if energy_atom > resput_prob(1,id_2)
                        id_ee(1) = id_2;
                        id_ee(2) = id_2+1;
                    else
                        id_ee(2) = id_2;
                        id_ee(1) = id_2-1;
                    end
                    id_theta = [0;0];
                    [~, id_2] = min(abs(resput_prob(:,1) - angle_atom2)); %
                    if angle_atom2 > resput_prob(id_2,1)
                        id_theta(1) = id_2;
                        id_theta(2) = id_2+1;
                    else
                        id_theta(2) = id_2;
                        id_theta(1) = id_2-1;
                    end
                    if id_theta(1) == 1 || id_ee(1) == 1
                        error('this cant happen')
                    end
                    e_Y = [resput_prob(id_theta(1),1),resput_prob(id_theta(2),1)];
                    e_X = [resput_prob(1,id_ee(1)),resput_prob(1,id_ee(2))];
                    e_V = [resput_prob(id_theta(1),id_ee(1))
resput_prob(id_theta(1),id_ee(2));
                        resput_prob(id_theta(2),id_ee(1)) resput_prob(id_theta(2),id_ee(2))
];
                    prob_out = interp2(e_X,e_Y,e_V,energy_atom,angle_atom2);
                    atom2_num = floor(prob_out);
                    next_prob2 = prob_out - atom2_num; % probability for the next ion
                    if next_prob2 < 0
                        error('sthg wrong, probability cannot be less than 0')
                    end
                    gen_num = rand();
                    if (next_prob2 >= gen_num)
                        atom2_num = atom2_num + 1;
                    end
                    % advance this resputered atom called atom2
                    for jjj = 1:atom2_num

                        height_id_change(n_i_c,1) = x_atom_impact;
                        height_id_change(n_i_c,2) = -1;
                        n_i_c = n_i_c + 1;

                        % choose the angle of re-sputtering
                        dist_angle2_sput = resput_dis(:,2);
                        % using this distribution to choose sputtered atom
                        px = resput_dis(:,1);
                        p = dist_angle2_sput;
                        %px = px(:);
                        p=p./trapz(px,p);
                        cdfp = cumtrapz(px,p);
                        % finding the parts of cdf parallel to the X axis
                        ind=[true; not(diff(cdfp)==0)];
                        % and cut out the parts
                        cdfp=cdfp(ind);
                        p=p(ind);
                        px=px(ind);
                        % generating the uniform distributed random numbers
                        uniformDistNum=rand(1);
                        % and distributing the numbers using cdf from input pdf
```

```
                        phi_atom2_local=interp1(cdfp,px,uniformDistNum,'linear');
                        % bring back to global angle along with up or down
                        if strcmp(atom_rel,'right_hit')
                            atom2_rel = 1;
                        elseif strcmp(atom_rel,'left_hit')
                            atom2_rel = -1;
                        else
                            error('has to be one of these cases')
                        end
                        phi_atom2_local = phi_atom2_local*atom2_rel; %adjust for left and
right side, this is relative to global
                        % bring back angle to global scale
                        phi_atom2 = phi_atom2_local + bin_normal_angle_local_atom;
                        phi_atom2_o = phi_atom;
                        if phi_atom2 >= 90
                            hu2 = -1; %identify condition for height increase or decrease
                            phi_atom2 = 180 - phi_atom2;
                        elseif phi_atom2 <= -90
                            hu2 = -1;
                            phi_atom2 = -180 - phi_atom2;
                        else
                            hu2 = 1;
                        end
                        x_atom2 = x_atom_impact;
                        h_atom2 = h_atom_impact;
                        initiate2 = true;
                        deposit_cond2 = false;
                        % advance this resputtered atom
                        while h_atom2 < h_capture
                            cstep2 = dx/2;
                            if abs(phi_atom2 - 90) < 10^(-7)
                                x_atom2 = x_atom2 - cstep2;
                                motion2 = 'left';
                            elseif (phi_atom2 + 90) < 10^(-7)
                                x_atom2 = x_atom2 + cstep2;
                                motion2 = 'right';
                            else
                                % based on phi_atom, compute reasonable step size
                                if abs(phi_atom2) >= atand(dhi/100)
                                    dha2 = abs(dhi/tand(abs(phi_atom2)));
                                else
                                    dha2 = dhi;
                                end
                                if dha2*tand(phi_atom2) > 0
                                    motion2 = 'left';
                                    x_atom2 = x_atom2 - cstep2;
                                else
                                    motion2 = 'right';
                                    x_atom2 = x_atom2 + cstep2;
                                end
                                h_atom2 = h_atom2 + hu2*(cstep2/tand(abs(phi_atom2)));
                                %x_atom2 = x_atom2 + cstep2;
                            end
                            if h_atom2 < 0
                                parsave('last_structure',h_bin);
                                warning('h_atom can never be negative, intersection never
occurs, check code or add more height if needed');
                                error('x_ion_debug of %d with phi of %d and phi atom of %d
and phi atom 2 of %d',...
                                    x_ion_debug, phi, phi_atom_o,phi_atom2_o);
                            end
                            if initiate2 == true
                                initiate2 = false;
                                if strcmp(motion2,'right')
                                    [x_atom2_start, h_atom2_start] = line_intersection(...
                                        [x_atom_impact, h_atom_impact, x_atom2,
h_atom2],...
```

```
                                    [surf_xa(2), surf_heighta(2), surf_xa(2),
surf_heighta(2)+1]);
                                    if h_atom2_start < surf_heighta(2)
                                        error('wrong initialization of atom2 with x_ion %d,
phi %d, phi_atom %d, phi_atom2 %d',...
                                            x_ion_debug, phi, phi_atom_o, phi_atom2_o')
                                    end
                                elseif strcmp(motion2,'left')
                                    [x_atom2_start, h_atom2_start] = line_intersection(...
                                        [x_atom_impact, h_atom_impact, x_atom2,
h_atom2],...
                                    [surf_xa(1), surf_heighta(1), surf_xa(1),
surf_heighta(1)+1]);
                                    if h_atom2_start < surf_heighta(1)
                                        error('wrong initialization of atom2 with x_ion %d,
phi %d, phi_atom %d, phi_atom2 %d',...
                                            x_ion_debug, phi, phi_atom_o, phi_atom2_o')
                                    end
                                end
                                if x_atom2_start >= xmax
                                    x_atom2_start = x_atom2_start - xmax + xmin;
                                elseif x_atom2_start <= xmin
                                    x_atom2_start = xmax - (xmin - x_atom2_start);
                                end
                                x_atom2 = x_atom2_start;
                                h_atom2 = h_atom2_start;
                            else
                                % identify the region where x_atom belongs
                                xreg = x;
                                hreg = h_bin(:,3)';
                                [~, iddd] = mink(abs(xreg - x_atom2),2); %
                                iddd = sort(iddd); % since x is always increasing
                                surf_xa2 = xreg(iddd);
                                if surf_xa2(1) >= surf_xa2(2)
                                    error('needs to order correctly and not the same
xpoint');
                                end
                                surf_heighta2 = hreg(iddd);
                                if size(surf_xa2) > 2
                                    error('cannot have three points')
                                end
                                % find the intersection between the ion path and the slope
                                [x_atom2_impact, h_atom2_impact] =
line_intersection([x_atom2_start, h_atom2_start, x_atom2, h_atom2],...
                                    [surf_xa2(1), surf_heighta2(1), surf_xa2(2),
surf_heighta2(2)]);

                                % check to see wheter the ion_impact is within the region
                                if x_atom2_impact > surf_xa2(1) && x_atom2_impact <=
surf_xa2(2)
                                    deposit_cond2 = true;
                                    break;
                                end
                                if x_atom2 >= xmax
                                    x_atom2 = x_atom2 - xmax + xmin;
                                elseif x_atom2 <= xmin
                                    x_atom2 = xmax - (xmin - x_atom2);
                                end
                                if x_atom2 < xmin || x_atom2 > xmax
                                    error('x_atom out of zone, check codes');
                                end
                                if h_atom2 < surf_heighta2(1) && h_atom2 < surf_heighta2(2)
                                    error('dips 2 below with x_ion %d, phi %d, phi_atom %d,
phi_atom2 %d',...
                                        x_ion_debug, phi, phi_atom_o, phi_atom2_o)
                                end
                                x_atom2_start = x_atom2;
```

```
                                h_atom2_start = h_atom2;
                            end
                        end
                        if deposit_cond2 == true
                            height_id_change(n_i_c,1) = x_atom2_impact;
                            height_id_change(n_i_c,2) = 1;
                            n_i_c = n_i_c + 1;
                        else
                            atom_out = atom_out+1;
                        end
                    end
                end
            else
                % increase atom count, add in the sputtered angle bin
                atom_out = atom_out + 1;
%                 [~, idm] = min(abs(angle_bar(:,1)-phi_atom));
%                 angle_bar(idm,2) = angle_bar(idm,2) + 1;
            end
        end
        hc_store{i} = height_id_change;
    end

    % change surface height based on height_id_change storage
    height_id_change = [];
    parfor i = 1:size(hc_store,1)
        if ~isempty(hc_store{i})
            height_id_change = [height_id_change;hc_store{i}];
        end
    end

    for iii = 1:size(height_id_change,1)
        x_change = height_id_change(iii,1);
        [~, iddd] = mink(abs(x - x_change),2); %
        iddd = sort(iddd); % since x is always increasing
        x_add = x(iddd);
        if x_add(1) >= x_add(2)
            error('needs to order correctly and not the same xpoint');
        end
        xh_x = x_add(2) - x_change;
        x_xl = x_change - x_add(1);
        if xh_x < 0 || x_xl < 0
            error('something wrong in linear interpolation');
        end
        xl_add_ratio = xh_x/dx;
        xh_add_ratio = x_xl/dx;
        if abs((xl_add_ratio + xh_add_ratio) - 1) > 10^-7
            error('the two ratios are mismatch')
        end
        h_bin(iddd(1),3) = h_bin(iddd(1),3) +
height_id_change(iii,2)*h_p_atom*xl_add_ratio;
        if iddd(1) == 1
            h_bin(end,3) = h_bin(end,3) + height_id_change(iii,2)*h_p_atom*xl_add_ratio;
        elseif iddd(1) == size(h_bin,1)
            error('impossible, something wrong from the code')
        end
        h_bin(iddd(2),3) = h_bin(iddd(2),3) +
height_id_change(iii,2)*h_p_atom*xh_add_ratio;
        if iddd(2) == 1
            error('impossible, something wrong from the code')
        elseif iddd(2) == size(h_bin,1)
            h_bin(1,3) = h_bin(1,3) + height_id_change(iii,2)*h_p_atom*xh_add_ratio;
        end
    end
    if h_bin(end,3) ~= h_bin(1,3)
        error('periodic condition does not satisfied')
    end
end
```

Support Function 1

```
function [yield_spline, md_y, vel_dis, angle_asso, en_dis, resput_prob, resput_dis] =
get_per_element_sputter_info(distribution_mode,...

    yield_mode,tt_eV,secondary_sputtering)

    if strcmp(distribution_mode, 'MD')
        if tt_eV ~= 8
            load velocity_delta_100_200_350_500_1000_1500_2000_v1_distribution_out_v2
            angle_asso = [0, 15, 30, 45, 60, 75, 90.0001];
            tt = tt_eV;
            vel_dis = vel_dis_l{tt};
            vel_dis(:,end+1) = 0;
            %vel_dis(:,end+1) = 0;
        else
            load 700eV_vel_dist
            angle_asso = [0, 30, 45, 60, 90.0001];
            vel_dis(:,end+1) = 0;
        end
    elseif strcmp(distribution_mode,'gaussian')
        distribution = @(u1,s1,u2,s2,a1,x) a1/(2*pi*s1^2)^(1/2).*(exp(-1/2*((x-
u1)/s1).^2))+...
            (1-a1)/(2*pi*s2^2)^(1/2).*(exp(-1/2*((x-u2)/s2).^2));
        angle_asso = [0, 15, 30, 45, 60, 75, 90.0001];
        vel_dis(:,1) = -90:0.1:90;
        vel_dis(:,2) = distribution(39.59,36.16,-44.11,32.48,0.54,vel_dis(:,1));
        vel_dis(:,3) = distribution(49.78,29.20,-37.34,35.78,0.56,vel_dis(:,1));
        vel_dis(:,4) = distribution(49.96,25.21,-43.38,29.66,0.73,vel_dis(:,1));
        vel_dis(:,5) = distribution(52.86,24.34,-37.30,30.62,0.75,vel_dis(:,1));
        vel_dis(:,6) = distribution(53.33,22.18,-35.82,33.31,0.76,vel_dis(:,1));
        vel_dis(:,7) = distribution(54.61,20.02,-35.18,35.70,0.81,vel_dis(:,1));
    else
        error('distribution mode not yet implemented')
    end

    if strcmp(yield_mode, 'interpolate')
        if tt_eV~=8
            md_y_l = [0, 15, 30, 45, 60, 75, 90;
                    0.004,0.006,0.0109,0.0185,0.0287,0.031,0;
                    0.032,0.0365,0.0884,0.146,0.2725,0.451,0;
                    0.072,0.109,0.146,0.31,0.87,1.25,0;
                    0.118,0.164,0.197,0.46,1.26,2.41,0;
                    0.25,0.28,0.54,1.21,2.29,4.99,0;
                    0.281,0.316,0.521,1.306,2.745,7.081,0;
                    0.38,0.41,0.79,1.56,4.79,9.96,0];
            tt = tt_eV;
            md_y = md_y_l([1 tt+1],:);
            md_y = md_y';
        else
            md_y = [0, 15, 30, 45, 60, 75, 90;
                0.15, 0.24, 0.39, 0.76, 1.63, 1.5, 0]';
        end
        yield_spline = spline(md_y(:,1),[0; md_y(:,2); 0]);
    elseif strcmp(yield_mode, 'interpolate_2')
        if tt_eV~=8
            md_y_l = [0, 15, 30, 45, 60, 75, 90;
                    0.004,0.006,0.0109,0.0185,0.0287,0.031,0;
                    0.032,0.0365,0.0884,0.146,0.2725,0.451,0;
                    0.072,0.109,0.146,0.31,0.87,1.25,0;
                    0.118,0.164,0.197,0.46,1.26,2.41,0;
                    0.25,0.28,0.54,1.21,2.29,4.99,0;
                    0.281,0.316,0.521,1.306,2.745,7.081,0;
                    0.38,0.41,0.79,1.56,4.79,9.96,0];
            tt = tt_eV;
```

```
            md_y = md_y_l([1 tt+1],:);
            md_y(2,end) = md_y(2,6); %md_y(2,5) or md_y(2,6)
            md_y = md_y';
        else
            md_y = [0, 15, 30, 45, 60, 75, 90;
                0.15, 0.24, 0.39, 0.76, 1.63, 1.5, 0]';
        end
        yield_spline = spline(md_y(:,1),[0; md_y(:,2); 0]);
    elseif strcmp(yield_mode,'eckstein')
        % model constant
        zi = 54; % Xe
        zs = 6; % C
        mi = 131.293; %g/mol - Xe
        ms = 12.011; %g/mol - C
        a0 = 0.529; %Angstrong
        e0 = 1.42*(10^(-40)); %C^2/eV/Angstrong
        ec = 1.602*10^(-19); %C
        q = 2.64;
        lamb = 3.96;
        muy = 2.08;
        eth = 11.2;
        f = 6.42;
        b = 2.17;
        c = 0.77;

        al = ((9*pi^2)/128)^(1/3)*a0*(zi^(2/3)+zs^(2/3))^(-1/2);
        ex = @(E) (al*4*pi*e0*ms)/(zi*zs*ec^2*(mi+ms))*E;
        sn = @(E) (0.5*log(1+1.2288*ex(E)))/(ex(E)+0.1728*ex(E)^(1/2)+0.008*ex(E)^0.1504);
        w = @(E) ex(E) + 0.1728*ex(E)^(1/2) + 0.008*ex(E)^0.1504;
        Y = @(E) q*sn(E)*((E/eth-1)^muy/(lamb/w(E)+(E/eth-1)^muy));
        Yp_eck = @(theta) (cos((theta/180*pi)^c))^(-f)*exp(b*(1-(cos((theta/180*pi)^c))^(-
1)));
        yield_spline = @(E,theta) Y(E)*Yp_eck(theta);

        md_y_l = [0, 15, 30, 45, 60, 75, 90;
                0.004,0.006,0.0109,0.0185,0.0287,0.031,0;
                0.032,0.0365,0.0884,0.146,0.2725,0.451,0;
                0.072,0.109,0.146,0.31,0.87,1.25,0;
                0.118,0.164,0.197,0.46,1.26,2.41,0;
                0.25,0.28,0.54,1.21,2.29,4.99,0;
                0.281,0.316,0.521,1.306,2.745,7.081,0;
                0.38,0.41,0.79,1.56,4.79,9.96,0];
        tt = tt_eV;
        md_y = md_y_l([1 tt+1],:);
        md_y = md_y';
    else
        error('yield mode not yet implemented')
    end

    if secondary_sputtering == true
        % get sputtered energy distribution data ** careful, not yet
        % implemented for 700 eV !!!!!!!!!
        mulist = [0.890803127257171 -1.25505011508176   -1.76150001924593
1.10330703600546    -0.535671611814807  2.14047056691307
                1.19711356745624    1.15126132171995    1.63386192971055
1.49288821621925    1.95247347760052    2.36698861954943
                0.952629607859864   1.05381750530434    1.89567274608419
2.00447266410960    2.29623750830951    2.77442304052233
                1.43712784596855    1.31660421095363    2.29347227796452
2.42406606923559    2.62354325258495    2.84642073845672
                1.91901127487733    1.97141790936741    2.18135377288743
2.52618762056674    2.73996029230724    3.00886943877295
                2.16823380520004    2.44550969967077    2.62033431787996
2.79250364726138    2.81057410247807    3.29273205733135
                2.47015157581941    2.65123521649329    2.69021144559772
2.95674040576943    3.03712015479775    3.29869404246938];
```

```
        silist = [0.744655157681104 1.92414914004126    1.98291298683157
1.14331286972502    2.17254407219369    0.234504626022444
                0.924394757151639   1.15689191709250    0.859649910005411
1.05193035661515    1.14856385414666    0.876138176103096
                1.00404637134491    1.24977153386207    1.07539727014040
1.18794551306472    1.12446572446714    0.958987439345509
                1.22178152432515    1.17059523446356    1.04650575107528
0.944370741476859   1.10828751078880    1.04841176240630
                1.11918462862484    1.19950998084321    1.39036049122000
1.36992493179008    1.37706316900451    1.22375147564802
                1.28026969934775    1.22417988590849    1.19476164766033
1.22563880497840    1.33411130032173    1.22547642306157
                0.939710489205547   0.976967846502947   1.20440374078511
1.32185349440752    1.38319533625632    1.35142225233991];
        en = @(mu,si,x) 1./(2*pi*x.^2*si^2).^(1/2).*(exp(-1/2*((log(x)-mu)/si).^2));
        en_dis(:,1) = 0:0.1:250;
        en_dis(:,2) = en(mulist(tt_eV,1),silist(tt_eV,1),en_dis(:,1));
        en_dis(:,3) = en(mulist(tt_eV,2),silist(tt_eV,2),en_dis(:,1));
        en_dis(:,4) = en(mulist(tt_eV,3),silist(tt_eV,3),en_dis(:,1));
        en_dis(:,5) = en(mulist(tt_eV,4),silist(tt_eV,4),en_dis(:,1));
        en_dis(:,6) = en(mulist(tt_eV,5),silist(tt_eV,5),en_dis(:,1));
        en_dis(:,7) = en(mulist(tt_eV,6),silist(tt_eV,6),en_dis(:,1));
        en_dis(:,8) = en_dis(:,7); % assumed velocity distribution at 90 degree is similar
to the one at 75 degree ***
        en_dis(1,:) = 0;
        % get resputtered property
        resput_prob = [-10 0 1 2 5 15 30 50 80 100 150
                    0   0 0.033333333333333    0.133333333333333   0.133333333333333
0.066666666666667   0.100000000000000   0.100000000000000   0.066666666666667
0.133333333333333   0.300000000000000
                    30  0 0.100000000000000 0.200000000000000   0.133333333333333
0.100000000000000   0.166666666666667   0.066666666666667  0.233333333333333
0.433333333333333   0.366666666666667
                    60  0 0.100000000000000 0.066666666666667  0.266666666666667
0.300000000000000   0.466666666666667   0.400000000000000   0.600000000000000
0.733333333333333   0.900000000000000
                    75  0 0.066666666666667    0.033333333333333  0.233333333333333
0.300000000000000   0.400000000000000   0.533333333333333   0.833333333333333
0.666666666666667   1.40000000000000
                    90  0 0 0 0 0 0 0 0 0 0];
        % get angle distribution of resputtering - assumed a cosine
        % distribution
        resput_dis(:,1) = -90:0.1:90;
        resput_dis(:,2) = cosd(resput_dis(:,1))/2;
        resput_dis(:,2) = resput_dis(:,2)/(trapz(resput_dis(:,1),resput_dis(:,2)));
    else
        en_dis = -1000;
        resput_prob = -1000;
        resput_dis = -1000;
    end
end
```

Support Function 2

```
%=========================================================================
%
% line_intersection  Finds the intersection of two lines.
%
%   [x_int,y_int] = line_intersection(line1,line2)
%
% See also polyxpoly.
%
% Copyright © 2021 Tamas Kis
% Last Update: 2021-09-05
% Website: https://tamaskis.github.io
```

```
% Contact: tamas.a.kis@outlook.com
%
% TECHNICAL DOCUMENTATION:
% https://tamaskis.github.io/documentation/Intersection_of_Two_Lines.pdf
%
%--------------------------------------------------------------------------
%
% ------
% INPUT:
% ------
%   line1   - (1×1, 1×2, 1×3, or 1x4 double) parameters defining line 1:
%               --> x1: vertical line form (x-intercept of vertical line)
%               --> [m1,b1]: slope-intercept form
%               --> [x1,y1,m1]: point-slope form
%               --> [x1,y1,x2,y2]: two point form
%   line2   - (1×1, 1×2, 1×3, or 1x4 double) parameters defining line 2:
%               --> x2: vertical line form (x-intercept of vertical line)
%               --> [m2,b2]: slope-intercept form
%               --> [x2,y2,m2]: point-slope form
%               --> [x3,y3,x4,y4]: two point form
%
% -------
% OUTPUT:
% -------
%   x_int   - (1×1 double) x-coordinate of intersection of the two lines
%   y_int   - (1×1 double) y-coordinate of intersection of the two lines
%
% -----
% NOTE:
% -----
%   --> vertical line form: x = x0
%   --> slope-intercept form: y = mx + b
%   --> point-slope form: y - y0 = m(x - x0)
%   --> two point form: (x1,y1) and (x2,y2) are two points on the line
%
% -----------
% EDGE CASES:
% -----------
%   --> Vertical lines can be defined using the vertical line form or the
%       two point form.
%   --> If the two lines are parallel (but not collinear), the function
%       returns positive or negative infinity for the coordinates of the
%       intersection, and a warning is displayed.
%   --> If the two lines are collinear (but not vertical), the function
%       returns "[NaN,NaN]" and displays a warning.
%   --> If the two lines are vertical and collinear, the function returns
%       the x-coordinate of the line for the x-coordinate of the
%       intersection, and "NaN" for the y-coordinate of the intersection.
%
%==========================================================================
function [x_int,y_int] = line_intersection(line1,line2)
    % determines point and slope for both lines
    [x1,y1,m1] = get_point_slope(line1);
    [x2,y2,m2] = get_point_slope(line2);

    % both lines vertical
    if (abs(m1) == Inf) && (abs(m2) == Inf)
        if x1 == x2
            x_int = x1;
            y_int = NaN;
            warning('The two lines are collinear.');
        else
            x_int = Inf;
            y_int = Inf;
        end

    % line 1 vertical
```

```
elseif abs(m1) == Inf
    x_int = x1;
    y_int = y2+m2*(x_int-x2);

% line 2 vertical
elseif abs(m2) == Inf
    x_int = x2;
    y_int = y1+m1*(x_int-x1);

% neither line vertical
else
    x_int = ((m1*x1-m2*x2)-(y1-y2))/(m1-m2);
    y_int = m1*(x_int-x1)+y1;
end

% displays warning if two (non-vertical) lines are collinear (results
% in (x,y) = (NaN,NaN))
if isnan(x_int) && isnan(y_int)
    warning('The two lines are collinear.');
end

% displays warning if the two lines are parallel (results in x = ±∞)
if abs(x_int) == Inf
    warning('The two lines are parallel.')
end

%=========================================================================
% get_point_slope  Given an input line of any form, determines its
% point-slope form.
%-------------------------------------------------------------------------
%
% ------
% INPUT:
% ------
%   line    - (1×1, 1×2, 1×3, or 1x4 double) params defining line 1:
%               --> x0: vertical line form (x-intercept of vertical
%                     line)
%               --> [m,b]: slope-intercept form
%               --> [x0,y0,m]: point-slope form
%               --> [x1,y1,x2,y2]: two point form
%
% -------
% OUTPUT:
% -------
%   x       - (1×1 double) x-coordinate of point on line
%   y       - (1×1 double) y-coordinate of point on line
%   m       - (1×1 double) slope of line
%
% -----
% NOTE:
% -----
%   --> If line is vertical, the function returns [x,NaN,Inf], where
%       "x" is the x-intercept of the vertical line.
%
%=========================================================================
function [x,y,m] = get_point_slope(line)

    % input given in vertical line form
    if length(line) == 1
        x = line(1);
        y = NaN;
        m = Inf;

    % input given in slope-intercept form
    elseif length(line) == 2
        x = 0;          % x-coordinate of the y-intercept
        y = line(2);    % y-coordinate of the y-intercept
```

```
            m = line(1);

        % input given in point-slope form
        elseif length(line) == 3
            x = line(1);
            y = line(2);
            m = line(3);

        % input given in two point form
        else
            m = (line(4)-line(2))/(line(3)-line(1));
            if abs(m) == Inf
                x = line(1);    % x-intercept of vertical line
                y = NaN;
                m = Inf;
            else
                x = line(1);    % x-coordinate of first point
                y = line(2);    % y-coordinate of first point
            end
        end

    end

end
```

Support Function 3

```
function parsave(fname,x)
    save(fname,'x');
end
```

Required additional data for sputterants ejection angle distribution

---

File name 1: 700eV_vel_dist.mat

Description: $N \times 5$ matlab matrix
Column 1: sputterants ejection angle, N points from -90 to 90 (° units)
Column 2: PDF for sputterants ejection angle at ion incidence of $0°$
Column 3: PDF for sputterants ejection angle at ion incidence of $30°$
Column 4: PDF for sputterants ejection angle at ion incidence of $45°$
Column 5: PDF for sputterants ejection angle at ion incidence of $60°$

File name 2: velocity_delta_100_200_350_500_1000_1500_2000_v1_distribution_out_v2.mat

Description: 1×7 cell, each cell is a $N \times 7$ matlab matrix
Cell 1: ion incidence at 100 eV
Cell 2: ion incidence at 200 eV
Cell 3: ion incidence at 350 eV
Cell 4: ion incidence at 500 eV
Cell 5: ion incidence at 1000 eV
Cell 6: ion incidence at 1500 eV
Cell 7: ion incidence at 2000 eV

For each cell,
Column 1: sputterants ejection angle, N points from -90 to 90 (° units)
Column 2: PDF for sputterants ejection angle at ion incidence of $0°$
Column 3: PDF for sputterants ejection angle at ion incidence of $15°$
Column 4: PDF for sputterants ejection angle at ion incidence of $30°$
Column 5: PDF for sputterants ejection angle at ion incidence of $45°$
Column 6: PDF for sputterants ejection angle at ion incidence of $60°$
Column 7: PDF for sputterants ejection angle at ion incidence of $75°$

A.2 MATLAB Code for Modified Fitness Function in GPTIPS2 for f-term in the New Sputtering Threshold Energy

Main Input and Run Code

```
clear;
clc;
close all

warning('off','all')

global trainvar_forsr

%% All term
addpath('XXXXX\gptips2\gptips2');
addpath('XXXXX');

trainvar_forsr = '123';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_mimt_zizt_zipzt_Eth_v1','gp');
clear gp

trainvar_forsr = '12';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_mimt_zizt_Eth_v1','gp');
clear gp

trainvar_forsr = '13';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_mimt_zipzt_Eth_v1','gp');
clear gp

trainvar_forsr = '23';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_zizt_zipzt_Eth_v1','gp');
clear gp

trainvar_forsr = '1';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_mimt_Eth_v1');
clear gp

trainvar_forsr = '2';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_zizt_Eth_v1');
clear gp

trainvar_forsr = '3';
gp = rungp(@configuration_mimt_zizt_zipzt);
save('LJ_zipzt_Eth_v1');
clear gp


function gp = configuration_mimt_zizt_zipzt(gp)

    global trainvar_forsr
    %run control
    gp.runcontrol.pop_size = 250; %population of 250 models
    gp.runcontrol.runs = 200; %perform 2 runs that are merged at the end
    gp.runcontrol.timeout = 180; %each run terminates after 60 seconds
    gp.runcontrol.parallel.auto = true; %enable Parallel Computing if installed
    %gp.runcontrol.num_gen = 150;
```

```
    %selection
    gp.selection.tournament.size = 150;
    gp.selection.tournament.p_pareto = 0.4; %encourages less complex models
    gp.selection.elite_fraction = 0.33; % approx. 1/3 models copied to next gen

    %fitness
    %gp.fitness.terminate = true;
    %gp.fitness.terminate_value = 7;

    %genes
    gp.genes.max_genes = 4;

    %constants
    %gp.nodes.const.p_ERC = 0.05;
    %gp.nodes.const.range = [-10 10];
    %gp.nodes.const.p_int= 0.5;

    %maximum depth of trees
    gp.treedef.max_depth = 3;
    gp.treedef.max_mutate_depth = 3; %maximum depth of sub-trees created by mutation
operator3
    gp.userdata.fitnessmod = true;

    %gp.fitness.fitfun = @quartic_fitfun;

    %genes
    %gp.genes.multigene = false;

    %function nodes
%     gp.nodes.functions.name = {'times','minus','plus','rdivide','square',...
%         'sin','cos','exp','mult3','add3','sqrt','cube','power','negexp',...
%         'neg','abs','log'};
    gp.nodes.functions.name = {'times','minus','plus','rdivide','square',...
       'sqrt','cube','power'};

    % constant
    epo_e2 = (1.42*10^(-40))/(1.6052*10^(-19))^2; %A/eV
    ao = 0.529; %angstrom
    epo_e2_ao = epo_e2/ao; %1/eV

    % get data
    data = readcell('XXX\Potential_LJ_summary_CORRECTED_v2.xlsx',"Sheet","mass study fix");

    plid = 1;

    for j = 3:703%size(data,1)
        % individual data
        mi = data{j,15};
        zi = data{j,14};
        radi = data{j,16};
        mt = data{j,6};
        zt = data{j,5};
        radt = data{j,10};
        alat = data{j,9};
        us = data{j,11};
        es = data{j,12};
        Ethmd = data{j,17};
        lattype = data{j,8};

        if contains(lattype,'fcc')
            afp = 1-0.74;
        elseif contains(lattype,'bcc')
            afp = 1-0.68;
        else
            error('lattype not yet implemented')
        end
```

```
%          if strcmp(lattype,'fcc')
%              latid = 1;
%              afp = 0.7854;
%          elseif strcmp(lattype,'bcc')
%              latid = 4;
%              afp = 0.589;
%          elseif strcmp(lattype,'fcc_111')
%              latid = 2;
%              afp = 0.907;
%          elseif strcmp(lattype,'bcc_110')
%              latid = 5;
%              afp = 0.83;
%          elseif strcmp(lattype,'fcc_110')
%              latid = 3;
%              afp = 0.555;
%          else
%              error('not yet implemented')
%          end

          % get plotting quantity
          if zi/mi <= 10 && zt/mt <= 10 && mi/mt <= 13 && mt/mi <= 13
              if ~ismissing(data{j,17})
              %if contains(lattype,'fcc')
                  etype{plid,1} = data{j,2};
                  ionid{plid,1} = data{j,1};
                  %ionid{plid,1} = data{j,13};
                  z_t(plid,1) = zt;
                  z_i(plid,1) = zi;
                  m_t(plid,1) = mt;
                  m_i(plid,1) = mi;

                  mi_mt(plid,1) = mi/mt;
                  zi_zt(plid,1) = zi/zt;
                  zizt(plid,1) = zi*zt;
                  nucl_rad(plid,1) = 1./(zi^(0.23) + zt^(0.23));
                  radi_radt(plid,1) = radi/radt;
                  lat_param(plid,1) = alat;
                  zzm(plid,1)= 1/(zi*zt)*(mt/(mi+mt));
                  AA = mi/mt;
                  gamma(plid,1) = 4*AA/(1+AA)^2;
                  atom_volf(plid,1) = afp;

                  Eth_MD(plid,1) = Ethmd;
                  Us_MD(plid,1) = us;
                  Es_MD(plid,1) = es;
                  %Ethmd_es(plid,1) = Ethmd/es;
                  plid = plid+1;
              %end
              end
          end
      end

      % reduce the data so other can serve as validation
      train_rat = 0.8;
      id_train = unique(ionid,'stable');
      index = randperm(size(id_train,1),size(id_train,1));
      id_train = id_train(index(1:train_rat*floor(size(index,2))));

      gap = 1.25;
      %index = randperm(size(etype,1),size(etype,1));
      %train_index = index(1:train_rat*floor(size(index,2)));
      train_index = 1:1:size(Eth_MD,1);

      %pregap = (2.5*zeros(size(Eth_MD,1),2));

      pregap(:,1) = real( (Eth_MD).^(1) - ((Eth_MD-gap)).^(1))./Us_MD;
```

```
    pregap(:,2) = (((Eth_MD+gap)).^(1) - (Eth_MD).^(1))./Us_MD;
    pregap(pregap(:,1)<0,1) = 0;
    % gap low, gap high

%     mi_train = m_i(train_index);
%     mt_train = m_t(train_index);
    mi_mt_train = mi_mt(train_index);
    zi_zt_train = zi_zt(train_index);
    zizt_train = zizt(train_index);
    Es_t_train = Es_MD(train_index);
    Us_t_train = Us_MD(train_index);
    gamma_train = gamma(train_index);
    zzm_train = zzm(train_index);
    lat_param_train = lat_param(train_index);
    Eth_MD_train = Eth_MD(train_index);
    nucl_rad_train = nucl_rad(train_index);
    atom_volf_train = atom_volf(train_index);

    gp.userdata.idinterest = id_train;
    gp.userdata.train_index = index;

    %input1 = zi_zt_train;
    input2 = nucl_rad_train;
    input3 = m_t./m_i;
    input4 = zizt_train;

    gp.userdata.pregaptrain = pregap(train_index,:); %modified this if modifed ytrain
    gp.userdata.ionidtrain = ionid(train_index);

    gp.userdata.ytrain = (Eth_MD_train);%./Us_t_train;

    if strcmp(trainvar_forsr,'123')
        gp.userdata.string_in_out = {'mt/mi','zizt','1/(zi^0.23+zt^0.23)','eth'};
        gp.userdata.xtrain = [input3, input4, input2];
    elseif strcmp(trainvar_forsr,'12')
        gp.userdata.string_in_out = {'mt/mi','zizt','eth'};
        gp.userdata.xtrain = [input3, input4];
    elseif strcmp(trainvar_forsr,'13')
        gp.userdata.string_in_out = {'mt/mi','1/(zi^0.23+zt^0.23)','eth'};
        gp.userdata.xtrain = [input3, input2];
    elseif strcmp(trainvar_forsr,'23')
        gp.userdata.string_in_out = {'zizt','1/(zi^0.23+zt^0.23)','eth'};
        gp.userdata.xtrain = [input4, input2];
    elseif strcmp(trainvar_forsr,'1')
        gp.userdata.string_in_out = {'mt/mi','eth'};
        gp.userdata.xtrain = [input3];
    elseif strcmp(trainvar_forsr,'2')
        gp.userdata.string_in_out = {'zizt','eth'};
        gp.userdata.xtrain = [input4];
    elseif strcmp(trainvar_forsr,'3')
        gp.userdata.string_in_out = {'1/(zi^0.23+zt^0.23)','eth'};
        gp.userdata.xtrain = [input2];
    else
        error('not yet implemented');
    end

    %gp.userdata.normalize = input5(train_index);
    gp.userdata.normalize = ones(size(Eth_MD,1),1);

end
```

Replacing the gpmodel2struct.m and regressmulti_fitfun.m in GPTIPS2 folder

```
function gpmodel = gpmodel2struct(gp,ID,tbxStats,createSyms,modelStruc,fastSymMode)
%GPMODEL2STRUCT Create a struct describing a multigene regression model.
%
%    GPMODEL = GPMODEL2STRUCT(GP,ID) gets the multigene regression model
%    specified by the identifier ID from the GPTIPS struct GP and returns
%    model info and performance data as a struct GPMODEL.
%
%    The GPMODEL2STRUCT struct contains a variety of model information
%    including symbolic  object versions of the multigene regression  model
%    and its constituent genes as well as model performance metrics for the
%    training, validation and test data sets (if present). This struct - as
%    well as being a convenient store of model performance data - may also
%    be used as an input to several other GPTIPS model analysis functions,
%    e.g.
%
%    GPPRETTY(GP,GPMODEL)
%
%    GPMODELREPORT(GP,GPMODEL)
%
%    RUNTREE(GP,GPMODEL)
%
%    GPMODEL2SYM(GP,GPMODEL)
%
%    GPMODEL2MFILE(GP,GPMODEL)
%
%    GPMODEL2FUNC(GP,GPMODEL)
%
%    GPMODELGENES2MFILE(GP,GPMODEL)
%
%    DRAWTREES(GP,GPMODEL)
%
%    Note:
%
%    The GPMODEL struct is functionally identical to that returned by the
%    function GPGENES2MODEL wherein a 'new' multigene regression model may
%    be constructed from the unique genes in a population.
%
%    Additionally:
%
%    GPMODEL = GPMODEL2STRUCT(GP,'best') gets info on the 'best' model in
%    the population (as evaluated on training data).
%
%    GPMODEL = GPMODEL2STRUCT(GP,'valbest') gets info on the 'best' model
%    (as evaluated on the validation data set (if this data exists).
%
%    GPMODEL = GPMODEL2STRUCT(GP,'testbest') gets info on the 'best' model
%    (as evaluated on the validation data set (if this data exists).
%
%    GPMODEL = GPMODEL2STRUCT(GP,ID,TBXSTATS) where TBXSTATS is TRUE does
%    the same but computes additional model performance stats (on the
%    training data) using the Statistics Toolbox (TBXSTATS default is
%    FALSE). These may be found in the field GPMODEL.TRAIN.TBXSTATS which
%    contains the output of the Statistics Toolbox REGSTATS function.
%
%    GPMODEL = GPMODEL2STRUCT(GP,ID,TBXSTATS,CREATESYMS) where CREATESYMS is
%    FALSE does the same but does not generate any symbolic math objects
%    using the Symbolic Math Toolbox. (CREATESYMS default is TRUE).
%
%    Remarks:
%
%    This function always returns a GPMODEL struct even if the supplied
%    model identifier is invalid (e.g. if 'valbest' is specified but there
%    is no validation data) or if the model is invalid in some other way
```

```
%    (e.g. there were no gene weights computed for it due to non-finite gene
%    outputs on the training data). The field GPMODEL.VALID in the returned
%    struct is TRUE for valid models and FALSE otherwise. If the model is
%    invalid then the reason for the model's invalidity can always be found
%    in the field GPMODEL.INVALIDREASON. E.g. 'Invalid model index
%    supplied.'
%
%    Furthermore - for a valid model - if there was 'test' set data supplied
%    during the GP run then the field GPMODEL.TEST.WARNING will be TRUE if
%    there was a problem predicting the test set values amd the reason for
%    the problem will be in the field GPMODEL.TEST.WARNINGREASON (e.g.
%    'Non-finite or non-real predictions on testing data.'). Similarly for
%    validation data, if there was a problem predicting it then the field
%    GPMODEL.VAL.WARNING is TRUE and the reason found in
%    GPMODEL.VAL.WARNINGREASON.
%
%    It is quite possible for a model to be formally 'valid' (i.e. predicts
%    OK on the training data) but fails on the test or validation data (e.g.
%    due to division by zero etc.)
%
%    This function computes additional model performance stats using the
%    sub-function REGRESSMULTI_FITFUN_FULL_STATS. So, if you change the
%    multigene regression fitness function REGRESSMULTI_FITFUN then you will
%    also need to change REGRESSMULTI_FITFUN_FULL_STATS.
%
%    Copyright (c) 2009-2015 Dominic Searson
%
%    GPTIPS 2
%
%    See also GPMODEL2MFILE, GPMODEL2FUNC, GENES2GPMODEL, GPMODEL2SYM,
%             GPMODELREPORT, DRAWTREES, RUNTREE, GPMODELGENES2MFILE

if nargin < 2
    gpmodel.valid = false;
    gpmodel.invalidReason = 'The function GPMODEL2STRUCT requires at least 2 arguments,
e.g. GPMODEL2STRUCT(GP,''BEST'')';
    return;
end

if ~strncmpi('regressmulti',func2str(gp.fitness.fitfun),12)
    error('GPMODEL2STRUCT may only be used to extract model data from a GP structure with a
population containing multigene symbolic regression models.');
end

if nargin < 6 || isempty(fastSymMode)
    fastSymMode = false;
end

%scan model genes for input frequency, depth, complexity? (default = yes).
%(setting to false is a bit quicker but doesn't compute the structural info)
if nargin < 5 || isempty(modelStruc)
    modelStruc = true;
end

%create symbolic objects of overall model and genes? (default = yes).
%(setting to false is significantly quicker but you don't get the symbolic
%math object for each gene)
if nargin < 4 || isempty(createSyms)
    createSyms = true;
end

%compute statistics toolbox stats? (default = no).
if nargin < 3 || isempty(tbxStats)
    tbxStats = false;
end

%set up default test, train and validation values
```

```
gpmodel.train.r2 = 0;
gpmodel.train.rmse = Inf;
gpmodel.train.mse = Inf;
gpmodel.train.sse = Inf;
gpmodel.train.mae = Inf;
gpmodel.train.maxe = Inf;
gpmodel.train.err = [];
gpmodel.train.ypred = [];
gpmodel.train.gene_outputs = [];
gpmodel.train.datapoints = [];
gpmodel.train.warning = false;
gpmodel.train.warningReason = '';

gpmodel.val.r2 = 0;
gpmodel.val.rmse = Inf;
gpmodel.val.mse = Inf;
gpmodel.val.sse = Inf;
gpmodel.val.mae = Inf;
gpmodel.val.maxe = Inf;
gpmodel.val.err = [];
gpmodel.val.ypred = [];
gpmodel.val.gene_outputs = [];
gpmodel.val.datapoints = [];
gpmodel.val.warning = false;
gpmodel.val.warningReason = '';

gpmodel.test.r2 = 0;
gpmodel.test.rmse = Inf;
gpmodel.test.mse = Inf;
gpmodel.test.sse = Inf;
gpmodel.test.mae = Inf;
gpmodel.test.maxe = Inf;
gpmodel.test.err = [];
gpmodel.test.ypred = [];
gpmodel.test.gene_outputs = [];
gpmodel.test.datapoints = [];
gpmodel.test.warning = false;
gpmodel.test.warningReason = '';

%encoded genes supplied in cell array
cellgenes = false;

%parse user input for supplied numerical model identifier
if isnumeric(ID) && numel(ID) == 1

    if  ~mod(ID,1) && ID > 0 && ID <= gp.runcontrol.pop_size

        %get encoded trees, eval trees and return values
        treestrs = gp.pop{ID};
        evaltreestrs = tree2evalstr(gp.pop{ID},gp);
        rtnVals = gp.fitness.returnvalues{ID};
    else
        gpmodel.valid = false;
        gpmodel.invalidReason = 'Invalid model index supplied.';
        return;
    end

elseif ischar(ID) && strcmpi(ID,'best') %or 'best' on training

    treestrs = gp.results.best.individual;
    evaltreestrs = gp.results.best.eval_individual;
    rtnVals = gp.results.best.returnvalues;

elseif  ischar(ID) && strcmpi(ID,'valbest') %or 'best' on validation data

    % check that validation data is present
    if ~isfield(gp.results,'valbest')
```

```
        gpmodel.valid = false;
        gpmodel.invalidReason ='No validation data was found.';
        return;
    end

    treestrs = gp.results.valbest.individual;
    evaltreestrs = gp.results.valbest.eval_individual;
    rtnVals = gp.results.valbest.returnvalues;

elseif  ischar(ID) && strcmpi(ID,'testbest') %or 'best' on test data

    % check that test data is present
    if ~isfield(gp.results,'testbest')
        gpmodel.valid = false;
        gpmodel.invalidReason ='No test data was found.';
        return;
    end

    treestrs = gp.results.testbest.individual;
    evaltreestrs = gp.results.testbest.eval_individual;
    rtnVals = gp.results.testbest.returnvalues;


    %if user supplied list of genes in encoded form,e.g. {'n(x1,f(x8),x11)','c(x2,x8)'}
elseif iscell(ID)

    treestrs = ID;
    evaltreestrs = tree2evalstr(treestrs,gp);
    cellgenes = true;

    %return values need to be computed by the fitness function
    gp.state.force_compute_theta = true;
    gp.state.run_completed = false;
    gp.userdata.showgraphs = false;
    [fitness,gp,rtnVals] = feval(gp.fitness.fitfun,evaltreestrs,gp);

    if isinf(fitness)
        gpmodel.valid = false;
        gpmodel.invalidReason ='Supplied gene list gave non-finite values for predictions
of training data';
        return
    end

elseif isstruct(ID)

    gpmodel.valid = false;
    gpmodel.invalidReason ='Models already in struct format unsupported by gpmodel2struct';
    return

else %otherwise user did not supply a valid multigene model selector

    gpmodel.valid = false;
    gpmodel.invalidReason ='Invalid model selector supplied.';
    return
end

%get multigene regression stats
wstate = warning; warning off;
gpmodel = regressmulti_fitfun_full_stats(gpmodel,evaltreestrs,gp,rtnVals,tbxStats);
warning(wstate);

%compute some tree structural information
if modelStruc

    gpmodel.numNodes = getnumnodes(treestrs);
    gpmodel.expComplexity = getcomplexity(treestrs);
    inputVec = gpmodelvars(gp,ID);
```

```
    %inputs
    count = 0;
    for i=1:numel(inputVec)
        if inputVec(i) > 0
            count = count + 1;
            if ~isempty(gp.nodes.inputs) && i <= numel(gp.nodes.inputs.names) &&
~isempty(gp.nodes.inputs.names{i})
                gpmodel.inputs{count} = strtrim(gp.nodes.inputs.names{i});
            else
                gpmodel.inputs{count} = ['x' num2str(i)];
            end
        end
    end
    gpmodel.numInputs = numel(gpmodel.inputs);
    gpmodel.output = gp.nodes.output.name;

    %max depth
    maxDepth = 1;
    for i=1:numel(treestrs)
        maxDepth = max(maxDepth,getdepth(treestrs{i}));
    end
    gpmodel.maxDepth = maxDepth;

end

%create sym objects
if createSyms && gp.info.toolbox.symbolic

    if cellgenes
        ID = horzcat(ID,rtnVals);
    end

    [gpmodel.sym,gpmodel.genes.geneSyms] = gpmodel2sym(gp,ID,fastSymMode,true);
else
    gpmodel.sym = [];
    gpmodel.genes.geneSyms = [];
end

%original encoded genes
gpmodel.about = 'A struct representing a multigene regression model.';
gpmodel.genes.geneStrs = treestrs;
gpmodel.genes.geneWeights = rtnVals;
gpmodel = orderfields(gpmodel);
gpmodel.source = 'gpmodel2struct';

function gpmodel = regressmulti_fitfun_full_stats(gpmodel, evalstr,gp,theta,tbxStats)
%REGRESSMULTI_FITFUN_FULL_STATS updates GPMODEL struct with stats for an existing multigene
regression model.
%
%    GPMODEL = REGRESSMULTI_FITFUN_FULL_STATS(EVALSTR,GP,THETA,GRAPHS,TBXST
%    ATS) updates a structure GPMODEL containing performance data about the
%    multigene symbolic regression individual represented by EVALSTR and
%    THETA.
%
%    Remarks:
%
%    Utility function. Reproduces functionality of REGRESSMULTI_FITFUN in a
%    more useful (but slower) form for offline use. It calculates a variety
%    of performance statistics which can be accessed via the fields of
%    GPMODEL. Also sets warning flags for 'test' and 'validation' data sets
%    if the model fails to predict for them.

%statistics toolbox stats off by default
if nargin < 4
    tbxStats = false;
end
```

```
%ensure gene weights in matrix form
if iscell(theta)
    theta = theta{1};
end

gpmodel.valid = true;
gpmodel.invalidReason = '';

%if the gene weights vector is empty it means that during the
%fitness evaluation it could not be generated due to either
%non-finite or complex model outputs OR numerical problem in the
%SVD least squares computation of weights. This is a model "showstopper"
%and so the model is marked as 'invalid'
if numel(evalstr) ~= (numel(theta)-1)
    gpmodel.valid = false;
    gpmodel.invalidReason = 'Empty gene weights vector: probably because model output
contained complex or non-finite values on training data.';
    return
end

%process evalstr with regex to allow direct access to data matrices
pat = 'x(\d+)';
evalstr = regexprep(evalstr,pat,'gp.userdata.xtrain(:,$1)');
numTrainData = numel(gp.userdata.ytrain);
numGenes = length(evalstr);
gpmodel.genes.num_genes = numGenes;

%set up a matrix to store the tree outputs plus a bias column of ones
geneOutputsTrain = ones(numTrainData,numGenes+1);

%eval each gene in the current individual on training data
for i=1:numGenes
    ind = i + 1;
    eval(['geneOutputsTrain(:,ind)=' evalstr{i} ';']);
end

%add raw gene outputs to struct (not bias term)
gpmodel.train.gene_outputs = geneOutputsTrain(:,2:end);
gpmodel.train.datapoints = numTrainData;

%check for nonsensical answers - exit with invalid model if any found
if  any(any(~isfinite(geneOutputsTrain))) || any(any(~isreal(geneOutputsTrain)))
    gpmodel.valid = false;
    gpmodel.invalidReason = 'Non-finite or non-real gene output values on training data.';
    return
end

%calc contribution of weighted individual genes on training data
gpmodel.train.ypred_genes = geneOutputsTrain.*repmat(theta',numTrainData,1);

%calc overall prediction of training data
gpmodel.train.ypred = geneOutputsTrain*theta;

%error (training data)
errTrain = gp.userdata.ytrain - gpmodel.train.ypred;
gpmodel.train.err = errTrain;

%SSE (training data)
gpmodel.train.sse = (errTrain'*errTrain);

%MSE (training data)
gpmodel.train.mse = gpmodel.train.sse/numTrainData;

%RMS prediction error - training data
gpmodel.train.rmse = sqrt(gpmodel.train.mse);
```

```
%r2 for training data
r2train = 1 - (gpmodel.train.sse /sum( (gp.userdata.ytrain-mean(gp.userdata.ytrain)).^2 )
);
gpmodel.train.r2 = r2train;

%mean absolute error (training)
gpmodel.train.mae = mean(abs(errTrain));

%max abs error (training)
gpmodel.train.maxe = max(abs(errTrain));

%process validation data if present
if isfield(gp.results,'valbest')

    evalstr = strrep(evalstr,'.xtrain','.xval');
    numValData = length(gp.userdata.yval);

    geneOutputsVal = zeros(numValData,numGenes+1);
    geneOutputsVal(:,1) = ones;

    for i=1:numGenes
        ind = i + 1;
        eval(['geneOutputsVal(:,ind)=' evalstr{i} ';']);
    end

    gpmodel.val.gene_outputs = geneOutputsVal(:,2:end);
    gpmodel.val.datapoints = numValData;

    %flag warning if non-real or complex predictions on validation data
    if  any(any(~isfinite(geneOutputsVal))) || any(any(~isreal(geneOutputsVal)))
        gpmodel.val.rmse = Inf;
        gpmodel.val.warning = true;
        gpmodel.val.warningReason = 'Non-finite or non-real predictions on validation
data.';
    end

    %compute model stats and predictions if no warning
    if ~gpmodel.val.warning

        gpmodel.val.ypred = geneOutputsVal*theta; %create the prediction on the validation
data

        %error validation data
        errVal = gp.userdata.yval - gpmodel.val.ypred;
        gpmodel.val.err = errVal;

        %sse for validation data
        gpmodel.val.sse = errVal'*errVal;

        %MSE validation data
        gpmodel.val.mse = gpmodel.val.sse/numValData;

        %rmse validation data
        gpmodel.val.rmse = sqrt(gpmodel.val.mse);

        %R2 for validation data
        gpmodel.val.r2 = 1 - (gpmodel.val.sse/sum( (gp.userdata.yval -
mean(gp.userdata.yval)).^2 ));

        %mean absolute error for validation data
        gpmodel.val.mae = mean(abs(errVal));

        %max abs error (validation)
        gpmodel.val.maxe = max(abs(errVal));

    end
```

```
    evalstr = strrep(evalstr,'.xval','.xtrain');

else
    gpmodel.val.rmse = Inf;
    gpmodel.val.warning = true;
    gpmodel.val.warningReason = 'No validation data was found.';
end %end of validation data calcs


%process test data
if (isfield(gp.userdata,'xtest')) && (isfield(gp.userdata,'ytest')) && ...
        ~isempty(gp.userdata.xtest) && ~isempty(gp.userdata.ytest)

    evalstr = strrep(evalstr,'.xtrain','.xtest');
    numTestData = length(gp.userdata.ytest);

    geneOutputsTest = zeros(numTestData,numGenes+1);
    geneOutputsTest(:,1) = ones;

    for i=1:numGenes
        ind = i + 1;
        eval(['geneOutputsTest(:,ind)=' evalstr{i} ';']);
    end

    gpmodel.test.gene_outputs = geneOutputsTest(:,2:end);
    gpmodel.test.datapoints = numTestData;

    if  any(any(~isfinite(geneOutputsTest))) || any(any(~isreal(geneOutputsTest)))
        gpmodel.test.rmse = Inf;
        gpmodel.test.warning = true;
        gpmodel.test.warningReason = 'Non-finite or non-real predictions on testing data.';
    end

    if ~gpmodel.test.warning

        gpmodel.test.ypred = geneOutputsTest*theta;

        %error test data
        errTest = gp.userdata.ytest - gpmodel.test.ypred;
        gpmodel.test.err = errTest;

        %sse for test data
        gpmodel.test.sse = errTest'*errTest;

        %MSE test
        gpmodel.test.mse = gpmodel.test.sse/numTestData;

        %RMSE test data
        gpmodel.test.rmse = sqrt(gpmodel.test.mse);

        %r2 test data
        gpmodel.test.r2 = 1- (gpmodel.test.sse/sum( (gp.userdata.ytest -
mean(gp.userdata.ytest)).^2 ));

        %mean absolute error test data
        gpmodel.test.mae = mean(abs(errTest));

        %max abs error (testing)
        gpmodel.test.maxe = max(abs(errTest));

    end
else
    gpmodel.test.rmse = Inf;
    gpmodel.test.warning = true;
    gpmodel.test.warningReason = 'No test data was found.';
end
```

```
%calc statistical analysis of gene significance & other stats on training data
%(if stats toolbox is present)
if tbxStats && gp.info.toolbox.stats
    stats = regstats(gp.userdata.ytrain,geneOutputsTrain(:,2:end));
    gpmodel.train.pvals = stats.tstat.pval;
    gpmodel.train.tbxStats = stats;
else
    gpmodel.train.pvals = [];
    gpmodel.train.tbxStats = [];
end
```

```
function
[fitness,gp,theta,ypredtrain,fitnessTest,ypredtest,pvals,r2train,r2test,r2val,geneOutputs,g
eneOutputsTest,geneOutputsVal]=regressmulti_fitfun(evalstr,gp)
%REGRESSMULTI_FITFUN Fitness function for multigene symbolic regression.
%
%   This is the default fitness function for multigene symbolic regression
%   in GPTIPS.
%
%   [FITNESS,GP] = REGRESSMULTI_FITFUN(EVALSTR,GP) returns the FITNESS of
%   the symbolic expression(s) in the cell array EVALSTR using information
%   contained in the GP data struct. Here, FITNESS is the root mean squared
%   prediction error (RMSE) on the training data set.
%
%   [FITNESS,GP,THETA,YPREDTRAIN,FITNESS_TEST,YPREDTEST,PVALS,R2TRAIN,R2TEST,R2VAL]
%   = REGRESSMULTI_FITFUN(EVALSTR,GP) may be used post-run to return the
%   gene coefficients THETA, the prediction of the model on the training
%   data YPREDTRAIN, the RMSE fitness value FITNESS_TEST on the test data
%   set, the prediction of the model on the test data YPREDTEST, the
%   statistical p-values for bias and model terms are returned as PVALS
%   (PVALS only computed if the Statistics Toolbox is present, otherwise an
%   empty variable is returned). Additionally, coefficients of
%   determination (R^2) are returned as R2TRAIN, R2TEST and R2VAL.
%
%   Remarks:
%
%   Each observation of the response variable y is assumed to be an unknown
%   non-linear function of the corresponding observations of the predictor
%   variables x1,..xn.
%
%   Training data:
%
%   The GPTIPS configuration file should populate the following required
%   fields for the training data assuming 'Ntrain' observations on the
%   input and output data. GP.USERDATA.XTRAIN should be a (Ntrain X n)
%   matrix where the ith column contains the Ntrain observations of the ith
%   input variable xi. GP.USERDATA.YTRAIN should be a (Ntrain x 1) vector
%   containing the corresponding observations of the response variable y.
%
%   Testing data:
%
%   The following fields are optional and may be used, post-run, to see how
%   well evolved models generalise to an unseen test data set with Ntest
%   observations. They do not affect the model building process.
%   GP.USERDATA.XTEST should be a (Ntest X n) matrix where the ith column
%   contains the Ntest observations of the ith input variable xi.
%   GP.USERDATA.YTEST should be a (Ntest x 1) vector containing the
%   corresponding observations of the response variable y.
%
%   How multigene symbolic regression works:
%
%   In multigene symbolic regression, each prediction of y is formed by the
%   weighted output of each of the trees/genes in the multigene individual
%   plus a bias term. The number (M) and structure of the trees is evolved
%   automatically during a GPTIPS run (subject to user defined
```

```
%    constraints).
%
%    i.e. ypredtrain = c0 + c1*tree1 + ... + cM*treeM
%
%    where c0 = bias term
%          c1,..,cM are the weights
%          M is the number of genes/trees comprising the current individual
%
%    The weights (i.e. regression coefficients) are automatically determined
%    by a least squares procedure for each multigene individual and are
%    stored in GP.FITNESS.RETURNVALUES for future use.
%
%    Remarks:
%
%    Because the GP structure is modified within this function (i.e. the
%    field GP.FITNESS.RETURNVALUES is used to store the computed weighting
%    coefficients for each gene) the GP structure must be returned as an
%    output argument.
%
%    This fitness function is used for multigene symbolic regression for
%    GPDEMO2, GPDEMO3 and GPDEMO4 (the configuration files for these are
%    GPDEMO2_CONFIG.M and GPDEMO3_CONFIG.M respectively) but it can and
%    should be used for the user's own non-linear regression problems.
%
%    Copyright (c) 2009-2015 Dominic Searson
%
%    GPTIPS 2
%
%    See also REGRESSMULTI_FITFUN_VALIDATE, GPDEMO2_CONFIG, GPDEMO3_CONFIG,
%    GPDEMO4_CONFIG, GPDEMO2, GPDEMO3

%defaults in case of early exit
theta=[];ypredtrain=[];fitnessTest=[];ypredtest=[];pvals=[];
r2train=[];r2test=[];r2val=[];geneOutputs=[];geneOutputsTest=[];
geneOutputsVal=[];

% process evalstr with regex to allow direct access to data matrices
pat = 'x(\d+)';
evalstr = regexprep(evalstr,pat,'gp.userdata.xtrain(:,$1)');
y = gp.userdata.ytrain;
numData = gp.userdata.numytrain;
numGenes = numel(evalstr);

%set up a matrix to store the tree outputs plus a bias column of ones
geneOutputs = ones(numData,numGenes+1);

%eval each gene in the current individual
for i = 1:numGenes
    ind = i + 1;
    eval(['geneOutputs(:,ind)=' evalstr{i} ';']);

    %check for nonsensical answers and break out early with an 'inf' if so
    if  any(~isfinite(geneOutputs(:,ind))) || any(~isreal(geneOutputs(:,ind)))
        fitness = Inf;
        gp.fitness.returnvalues{gp.state.current_individual} = [];
        return
    end
end

%only calc. weighting coeffs during an actual run or if forced
if ~gp.state.run_completed || gp.state.force_compute_theta

    %set gp.userdata.bootSample to true to resample data for weights computation

    %prepare LS matrix
    if gp.userdata.bootSample
        sampleInds = bootsample(geneOutputs,gp.userdata.bootSampleSize);
```

```
        goptrans = geneOutputs(sampleInds,:)';
        prj = goptrans * geneOutputs(sampleInds,:);
        ysample = y(sampleInds);
    else
        goptrans = geneOutputs';
        prj = goptrans * geneOutputs;
    end

    %calculate tree weight coeffs using SVD based least squares
    %normal equation
    try
        if gp.userdata.bootSample
            theta = pinv(prj) * goptrans * ysample;
        else
            theta = pinv(prj) * goptrans * y;
        end
    catch
        theta = [];
        fitness = Inf;
        gp.fitness.returnvalues{gp.state.current_individual} = [];
        return;
    end

    %assign bad fitness if any coeffs NaN or Inf
    if any(isinf(theta)) || any(isnan(theta))
        theta = [];
        fitness = Inf;
        gp.fitness.returnvalues{gp.state.current_individual} = [];
        return;
    end

    %write coeffs to returnvalues field for storage
    gp.fitness.returnvalues{gp.state.current_individual} = theta;

else %if post-run, get stored coeffs from return value field
    theta = gp.fitness.returnvalues{gp.state.current_individual};
end

%calc. prediction of full training data set using the estimated weights
ypredtrain = geneOutputs * theta;

%calculate RMS prediction error (fitness)
pregaptrain = gp.userdata.pregaptrain;

ytrain = gp.userdata.ytrain;

conid = ~(ypredtrain < ytrain+pregaptrain(:,2) & ypredtrain > ytrain-pregaptrain(:,1));
err = zeros(size(ypredtrain,1),1);
err(conid) = ypredtrain(conid) - ytrain(conid);
err(err>0) = err(err>0)-pregaptrain(err>0,2);
err(err<0) = err(err<0)+pregaptrain(err<0,1);

ypredtrain = ytrain+err;

if gp.userdata.fitnessmod
    R_squared = 0;
    ionidtrain = gp.userdata.ionidtrain;
    idunique = gp.userdata.idinterest;
    normterm = gp.userdata.normalize;
    for kk = 1:size(idunique,1)
        xpred = [];
        ypred = [];
        m=1;
        for jj = 1:size(pregaptrain,1)
            if strcmp(ionidtrain{jj},idunique{kk})
                ypred(m) = ytrain(jj)*normterm(jj);
                xpred(m) = ypredtrain(jj)*normterm(jj);
```

```
                m=m+1;
            end
        end
        % get the best fit line
        [coefficients, S] = polyfitZero(xpred, ypred, 1);
        if coefficients(1) > 1000
            R_squared = 1000;
        else
            R_squared = R_squared + (1 - (S.normr/norm(ypred - mean(ypred)))^2);
            %yfit = polyval(coefficients, xpred);
%             SStot = sum((ypred-mean(ypred)).^2);                    % Total Sum-Of-
Squares
%             SSres = sum((ypred-yfit).^2);                         % Residual Sum-Of-Squares
%             Rsq = 1-SSres/SStot;
        end
    end
    if R_squared < 0
        fitness = 100;
    elseif R_squared/size(idunique,1) > 1
        fitness = 100;
    else
        fitness = 1 - R_squared/size(idunique,1);
    end
else
    fitness = sqrt(((err'*err)/numData));
end


%--below is for post-run evaluation of models, it is not used during a GPTIPS run--
if gp.state.run_completed

    %compute r2 for training data
    %ypredtrain = ytrain+err;
    if gp.userdata.fitnessmod
        r2train = fitness;
    else
        r2train = 1 - (err'*err)/sum( (gp.userdata.ytrain-mean(gp.userdata.ytrain)).^2 );
    end
    plotValidation = 0;

    %process validation data if present
    if (isfield(gp.userdata,'xval')) && (isfield(gp.userdata,'yval')) && ...
            ~isempty(gp.userdata.xval) && ~isempty(gp.userdata.yval)

        plotValidation = 1;
        evalstr = strrep(evalstr,'.xtrain','.xval');
        yval = gp.userdata.yval;
        numData = length(yval);

        %set up a matrix to store the tree outputs plus a bias column of ones
        geneOutputsVal = zeros(numData,numGenes + 1);
        geneOutputsVal(:,1) = ones;

        %eval each tree
        for i=1:numGenes
            ind = i+1;
            eval(['geneOutputsVal(:,ind)=' evalstr{i} ';']);
        end

        ypredval = geneOutputsVal*theta; %create the prediction  on the validation data

        pregapval = gp.userdata.pregapval;
        conid = ~(ypredval < yval+pregapval(:,2) & ypredval > yval-pregapval(:,1));
        err = zeros(size(ypredval,1),1);
        err(conid) = ypredval(conid) - yval(conid);
        err(err>0) = err(err>0)-pregapval(err>0,2);
        err(err<0) = err(err<0)+pregapval(err<0,1);
```

```
        fitness_val = sqrt(((err'*err)/numData));

        ypredval = yval +err;

        %compute r2 for validation data
        r2val = 1 - (err'*err)/sum( (gp.userdata.yval - mean(gp.userdata.yval)).^2 );
        evalstr = strrep(evalstr,'.xval','.xtrain');
    else
        r2val = [];
    end %end of validation data calcs


    %process test data if present
    plotTest = 0;
    if (isfield(gp.userdata,'xtest')) && (isfield(gp.userdata,'ytest')) && ...
            ~isempty(gp.userdata.xtest) && ~isempty(gp.userdata.ytest)

        plotTest = 1;
        evalstr = strrep(evalstr,'.xtrain','.xtest');
        ytest = gp.userdata.ytest;
        numData = length(ytest);

        %set up a matrix to store the tree outputs plus a bias column of ones
        geneOutputsTest = zeros(numData,numGenes+1);
        geneOutputsTest(:,1) = ones;

        %eval each tree
        for i=1:numGenes
            ind = i + 1;
            eval(['geneOutputsTest(:,ind)=' evalstr{i} ';']);
        end

        ypredtest = geneOutputsTest * theta; %create the prediction on the testing data

        pregaptest = gp.userdata.pregaptest;
        conid = ~(ypredtest < ytest+pregaptest(:,2) & ypredtest > ytest-pregaptest(:,1));
        err = zeros(size(ypredtest,1),1);
        err(conid) = ypredtest(conid) - ytest(conid);
        err(err>0) = err(err>0)-pregaptest(err>0,2);
        err(err<0) = err(err<0)+pregaptest(err<0,1);

        fitnessTest = sqrt(((err'*err)/numData));

        ypredtest = ytest+err;

        %compute r2 for test data
        r2test = 1 - (err'*err)/sum( (gp.userdata.ytest - mean(gp.userdata.ytest)).^2 );
    end


    %calc statistical analysis of gene significance on training data
    %(if stats toolbox is present)
    if gp.userdata.stats && gp.info.toolbox.stats
        % Regress tree outputs (and bias) against y train data and get stats
        wstate = warning;warning off;
        stats = regstats(y,geneOutputs(:,2:end));
        warning(wstate);
        pvals = stats.tstat.pval;
    else
        pvals = [];
    end
end

%if graphs required
if gp.state.run_completed && gp.userdata.showgraphs
```

```
    if ~isempty(gp.userdata.name)
        setname = ['Data: ' gp.userdata.name];
    else
        setname='';
    end

    %model predictions plots
    figure('name','GPTIPS 2 Multigene regression. Model predictions.','numbertitle','off');
    subplot(1+plotTest+plotValidation,1,1);
    plot(ypredtrain,'Color',[0.85 0.33 0.1]);
    hold on;
    plot(gp.userdata.ytrain,'Color',[0 0.45 0.74]);
    axis tight;
    ylabel('y');
    xlabel('Data point');
    legend('Predicted','Actual');
    title({setname,...
        ['RMS training set error: ' num2str(fitness) '  R^2: '
num2str(r2train)]},'interpreter','tex');
    hold off;

    if plotTest
        subplot(2+plotValidation,1,2);
        plot(ypredtest,'Color',[0.85 0.33 0.1]);
        hold on;
        plot(gp.userdata.ytest,'Color',[0 0.45 0.74]);
        axis tight;
        ylabel('y');
        xlabel('Data point');
        title(['RMS test set error: ' num2str(fitnessTest) '  R^2: '
num2str(r2test)],'interpreter','tex');
        hold off
    end
    if plotValidation
        subplot(2+plotValidation,1,3);
        plot(ypredval,'Color',[0.85 0.33 0.1]);
        hold on;
        plot(gp.userdata.yval,'Color',[0 0.45 0.74]);
        axis tight;
        ylabel('y');
        xlabel('Data point');
        title(['RMS validation set error: ' num2str(fitness_val) '  R^2: '
num2str(r2val)],'interpreter','tex');
        hold off
    end

    %scatterplots
    scatterFig = figure('name','GPTIPS 2 Multigene regression. Model prediction
scatterplot.','numbertitle','off');
    subplot(1+plotTest+plotValidation,1,1);
    minval = min([gp.userdata.ytrain;ypredtrain]);
    maxval = max([gp.userdata.ytrain;ypredtrain]);
    axis([minval maxval minval maxval]);
    ilineTr = line([minval maxval], [minval maxval]);
    set(ilineTr,'color','black','LineWidth',1);hold on;
    scatter(gp.userdata.ytrain,ypredtrain,'o','MarkerFaceColor',[0 0.45
0.74],'MarkerEdgeColor','none');
    box on;grid on;hold off;
    ylabel('Predicted');
    xlabel('Actual');
    title({setname,['RMS training set error: ' num2str(fitness) '  R^2: '
num2str(r2train)]},'interpreter','tex');

    %add scatter plot for test data, if present
    if plotTest
        subplot(2+plotValidation,1,2);
        minval = min([gp.userdata.ytest;ypredtest]);
```

```
        maxval = max([gp.userdata.ytest;ypredtest]);
        axis([minval maxval minval maxval]);
        ilineTest = line([minval maxval], [minval maxval]);
        set(ilineTest,'color','black','LineWidth',1);hold on;
        scatter(gp.userdata.ytest,ypredtest,'o','MarkerFaceColor',[0 0.45
0.74],'MarkerEdgeColor','none');
        box on;grid on;hold off;
        ylabel('Predicted');
        xlabel('Actual');
        title(['RMS test set error: ' num2str(fitnessTest) '  R^2: '
num2str(r2test)],'interpreter','tex');
    end

    %add scatter plot for validation data, if present
    if plotValidation
        figure(scatterFig);
        subplot(2+plotTest,1,2+plotTest);
        minval = min([gp.userdata.yval;ypredval]);
        maxval = max([gp.userdata.yval;ypredval]);
        axis([minval maxval minval maxval]);
        ilineVal = line([minval maxval], [minval maxval]);
        set(ilineVal,'color','black','LineWidth',1);hold on;
        scatter(gp.userdata.yval,ypredval,'o','MarkerFaceColor',[0 0.45
0.74],'MarkerEdgeColor','none');
        box on;grid on;hold off;
        title(['RMS validation set error: ' num2str(fitness_val) '  R^2: '
num2str(r2val)],'interpreter','tex');
        ylabel('Predicted');xlabel('Actual');
    end
    %gene weights & significance
    if  gp.info.toolbox.stats && gp.userdata.stats

        %generate x labels for bar graphs
        geneLabels = {'Bias'};
        for i = 1:numGenes
            geneLabels{i+1} = ['Gene ' int2str(i)];
        end

        %plot gene weights and offset
        statFig = figure; coeffsAx = subplot(2,1,1);
        set(statFig,'name','GPTIPS 2 P-values of model genes (on training
data)','numbertitle','off');
        geneBar = bar(coeffsAx,stats.beta);
        set(coeffsAx,'xtick',1:(numGenes+1));
        set(coeffsAx,'xticklabel',geneLabels);
        title(coeffsAx,{setname,'Gene weights'});

        %plot p-vals
        pvalsAx = subplot(2,1,2);
        pvalBar = bar(pvalsAx,stats.tstat.pval);

        if ~verLessThan('matlab','8.4') %R2014b
            pvalBar.FaceColor = [0 0.45 0.74];
            pvalBar.BaseLine.Visible = 'off';
            geneBar.FaceColor = [0 0.45 0.74];
        else
            pvalsBars = get(pvalsAx,'Children');
            set(pvalsBars,'FaceColor',[0 0.45 0.74],'ShowBaseLine','off');
            coeffsBars = get(coeffsAx,'Children');
            set(coeffsBars,'FaceColor',[0 0.45 0.74]);
        end
        set(pvalsAx,'xtick',1:(numGenes+1));
        set(pvalsAx,'xticklabel',geneLabels);
        title(pvalsAx,'P value');xlabel(['R^2 = ' num2str(r2train)]);

    end
end
```