User Manual

IM3D: A 3D Parallel Monte Carlo Simulation Code for Ion Irradiation of Nanostructured Materials

V1.0.0

Yonggang Li (Y.G. Li)

* * *

Research Laboratory for Computational Materials Sciences
Institute of Solid State Physics, Chinese Academy of Sciences

June 18, 2015

Declaration of Authorship

IM3D stands for Ion Irradiation of Nanostructured Materials - a 3D Parallel Monte Carlo Simulation Code.

 Coded by Yonggang Li (Y.G. Li), ygli@theory.issp.ac.cn, 2014, ISSP, ACS.

- IM3D is a 3D Monte Carlo simulation code designed to run efficiently on serial or parallel computers. It can simulate the transport of ions and subsequent radiation damages in arbitrary complex 3D nanostructured materials. It is an open-source code, distributed freely under the terms of the GNU Public License (GPL).
- IM3D was sponsored by Prof. Ju Li and Prof. Zhi Zeng, and mainly developed by Dr. Yonggang Li during his visiting to MIT in 2014. The supporting of CSG/FETM geometric algorithms form Prof. Zejun Ding as well as useful contributions and discussions from Dr. Machal Short and Yang Yang are very appreciate.
- If you have any questions, please contact to Dr. Yonggang Li, who can be emailed at ygli@theory.issp.ac.cn. The IM3D Website at ISSP and MIT has more information about the code and its uses.

Acknowledgements

This code was mainly supported by the scholarship from China Scholarship Council and Institute of Solid State Physics, Chinese Academy of Sciences (CAS) and partly by Massachusetts Institute of Technology (MIT). The subsequent supports come from the National Science Foundation of China under Grant Nos. 11275229, 11475215 & NSAF U1230202, the Special Funds for Major State Basic Research Project of China (973) under Grant No. 2012CB933702, the Hefei Center for Physical Science and Technology under Grant No. 2012FXZY004, and Director Grants of CASHIPS. Part of the tests were performed at the Center for Computational Science of CASHIPS, the ScGrid of Supercomputing Center, and the Computer Network Information Center of the Chinese Academy of Sciences. Y.G. Li is very grateful to Prof. Xiaohong Zheng and Prof. Yongsheng Zhang for their helpful suggestions on this code and works on website design.

Contents

Chapter 1

Introduction

This section provides an overview of what IM3D can and can't do, describe what it means for IM3D to be an open-source code.

1.1 What is IM3D

IM3D is an open-source parallel 3D Monte Carlo (MC) code for rapidly simulating the transportation of ions and the production of defects in nanostructured materials. It is an accurate, efficient and universal 3D version of MC model developed based on the standard SRIM databases[1], the fast database indexing technique[2] and MPI parallel algorithm as well as the 3D structural algorithms of Constructive Solid Geometry (CSG) / Finite Element Triangulated Mesh (FETM) methods[3–8]. It can model arbitrary-complex 3D targets made of different geometric elements each of which with different materials. Both the 3D distribution of ions and also all kinetic phenomena associated with the ion's energy loss, i.e., amorphization, damage, sputtering, ionization and phonon production, can be calculated by IM3D with following all target atom cascades in detail. Thus, IM3D code provides a general and robust theoretical approach to analysis the effects in primary damage processes and the corresponding 3D space-distributions of primary defects in nanostructured materials under ion beam irradiation.

The development of IM3D is mainly includes three aspects, i.e. the accurate physical models, the universal 3D structural models and the efficient calculation algorithms. The physical parameters used in the code, such as, the electronic stopping power and energy straggling parameters, are generated from SRMModule.exe provided by SRIM package[1]. The 3D nanostructured samples can be generated by graphical softwares beforehand and traced by the sophisticated 3D structural algorithms based on the CSG/FETM methods[3–8]. In order to further increasing the efficiency of the code, we

introduced the fast database indexing technique proposed in Corteo[2] to sampling the scattering and azimuthal angles as well as a linear speed-up MPI parallel algorithm or a multi-threading parallel algorithm. Detailed description of the physical basement can be found in our papers or in Appendix A.

IM3D can run efficiently on different platforms, including not only single- or multi-processors desktop or laptop machines but also parallel computers. Simultaneous multi-threading technique has be included in IM3D code to run on a multi-processors system. It can also run on any parallel machine that compiles plain C and supports the MPI message-passing library.

IM3D is a freely-available open-source code except for the geometric modules (copyrights belong to Prof. Zejun Ding), distributed under the terms of the GNU Public License, which means you can use or modify the code however you wish but commercial purposes. In order to part rights reserved, the routines related to CSG/FETM models are compiled to static libraries in IM3D package. In addition, a part of the modules in IM3D refer to the open-source codes, Iradina[9] and Corteo[2].

1.2 IM3D Features

This section highlights IM3D features:

- open-source distribution with highly portable C;
- ion with atomic number of $1-92$ and energy of 10 $eV-2$ GeV/amu, as well as different ion beam shape distribution, i.e., random, centered, defined position, random square around predefined position, Gaussian beam and etc.
- arbitrary complex targets constructed by the 3D geometric algorithms of CSG/FETM methods[3–8], with complex materials including single elements ($1-92$), alloys and compounds;
- generate input shapes in the form of different formats (e.g. *opengl*, *ply*2 and etc.) with different standard finite element softwares, e.g. Gmsh[10], Cubit and etc.;
- 1D (bulk and multi-layers) or 3D systems with or without semi-infinite substrate;
- runs from an input script or four separate input files (i.e., temp_compfile.im3d, temp_configfile.im3d, temp_matfile.im3d and temp_structfile.im3d);
- runs on a single processor or in parallel with distributed-memory message-passing parallelism (MPI);

- electronic energy loss and straggling are based on the standard SRIM databases[1], and Bragg's rule[43] is used to estimate the stopping power of a compound by the linear combination of the stopping powers of its individual elements;;
- uses fast database indexing technique (see Corteo[2]) or the MAGIC approximation formula [1] for sampling in terms of accuracy and efficiency;
- uses the analytical modified Kinchin-Pease (KP) model[11, 12] or the computationally full cascade (FC) simulation for defect generation processes;
- uses a screened repulsive Coulomb potential described by a dimensionless screening function, such as the Thomas-Feimi potential[35], the Lenz-Jensen potential[36], the Moliere potential[37], the Bohr potential[38] and the universal Ziggler-Biersack-Littmark (ZBL) potential[1] to describe the interaction potential between two atoms;
- output primary damage information including 1D (depth) and 3D distributions of electronic and nuclear energy depositions, back-scattering/implanted ions, *dpa*, interstitials, vacancies and sputtering atoms, etc.
- the output distribution files are in the format of $.cfg$, $.msh$ or $.vtk$, which can be viewed by various pre- and post-processing tools such as AtomEye[13], Gmsh[10], ParaView, Cubit and etc.

1.3 IM3D Non-features

IM3D is designed to efficiently simulate the tracing of ions in static arbitrary complex 3D systems. Some features that IM3D does not yet (maybe in the future version) support are list below:

- real-time tracing plot;
- restart;
- 64-bit system compatibility;
- dynamic version.

The serial version of the code can output xyz-coordinates of ions/atoms, which can give the plot of the tracing trajectories after the simulation. In the code, we introduced the fast database indexing technique (see Corteo[2]) which is mainly based on a 32-bit system. Thus, -m32 or -arch i386 should be used in the makefile when compiling. We

will also introduced a 64-bit version in the future. IM3D is a static version of the system without change the geometry shape of the system during irradiation.

Otherwise, many of the tools needed to pre- and post-process the data for 3D geometry shapes are not included in the IM3D kernel. Specifically, IM3D itself does not:

- run thru a GUI (would be included in the future version);
- build 3D geometry structures;
- perform sophisticated analyses;
- visualize simulation results;
- plot output data.

A few tool for constructing the input 3D geometry structures are provided as part of the IM3D package, as described in section X. Although users can be write their own tools for these tasks, but we recommend to use our tool to generate the CSG format geometry structures and the open-source software Gmsh to generate the FETM format geometry structures. For high-quality visualization we recommend the AtomEye[13], Gmsh[10] or ParaView softwares.

1.4 Open-source Distribution

IM3D comes with no warranty of any kind. It is a copyrighted code that is distributed free-of-charge, under the terms of the GNU Public License (GPL). This is often referred to as open-source distribution - see www.gnu.org or www.opensource.org for more details. The legal text of the GPL is in the LICENSE file that is included in the IM3D distribution.

Here is a summary of what the GPL means for IM3D users:

(1) Anyone is free to use, modify, or extend IM3D in any way they choose, without including for commercial purposes.

(2) If you distribute a modified version of IM3D, it must remain open-source, meaning you distribute it under the terms of the GPL. You should clearly annotate such a code as a derivative version of IM3D.

(3) If you release any code that includes IM3D source code, then it must also be open-sourced, meaning you distribute it under the terms of the GPL.

(4) If you give IM3D files to someone else, the GPL LICENSE file and source file headers (including the copyright and GPL notices) should remain part of the code.

In the spirit of an open-source code, these are various ways you can contribute to making IM3D better. Any questions please send email to Yonggang Li (Y.G. Li) .

1.5 Citations

Please cite our paper if you use subroutines in this package. Thanks.

Y.G. Li, Y. Yang, M. Short, Z.J. Ding, Z. Zeng and J. Li, Fast simulation of primary damages in arbitrarily complexed nanostructured materials under ion irradiation, to be published;

Y.G. Li, Y. Yang, M. Short, Z.J. Ding, Z. Zeng and J. Li, IM3D: A 3D Parallel Monte Carlo Simulation Code for Ion Irradiation of Nanostructured Materials, to be published.

Chapter 2

Getting Started

This section describes how to build and run IM3D.

2.1 What is in the IM3D Distribution

When you download IM3D you will need to unzip and untar the downloaded file with the following commands, after placing the file in an appropriate directory.

gunzip im3d.tar.gz*

tar xvf im3d.tar*

This will create an IM3D directory containing two files and several sub-directories:

- README Text file, general information;
- LICENSE The GNU General Public License (GPL);
- bin Executable files;
- data SRIM databases, input indexing tables, material properties, etc.;
- doc User manual;
- examples Tests and examples;
- lib Static libraries used for IM3D code, geometric modules.
- src Source codes;
- tools Useful tools for generating input script, shapes, databases and etc.

2.2 Making IM3D

2.2.1 Steps to build an IM3D executable file

Systems: *Windows*, *Linux* or *Mac*.

Compiler: Microsoft Visual Studio - *c* / *mingw*32, *gcc* and etc.

Libraries & packages: Standard *C* library and *MPICH* / *OpenMP* package.

The src directory contains the plain C source and header files for IM3D. It also contains a makefile file for linux/Mac systems:

This makefile can be used to compile im3d.
On Linux/UNIX or Mac systems, gcc is recommend for compilation.
On Windows system, mingw32 is recommend for compilation.

#CC = gcc # serial, linux or Mac
#CC = mingw32-gcc # serial, win
CC = mpicc # mpi, linux or Mac

#CFLAGS = -O2 -Wall -ansi -pedantic
#CFLAGS = -O2 -Wall -pedantic
#CFLAGS = -O1 -Wall
CFLAGS = -O1 -Wall -m32 # for 64-bit systems, -m32 or -arch i386 must be included
LDFLAGS = -lm -m32 # for 64-bit systems, -m32 or -arch i386 must be included
#LIBFLAG = -L. -lstruct_s # include static library, serial
LIBFLAG = -L. -lstruct_m # include static library, mpi

Notes on warning level:
Using -Wall and -pedantic will return many warnings, because of non-allowed
comment styles in the codes. I recommend using just -Wall.

Notes on the compiler optimization options -OX:
It is not recommended to use -O2 or -O3. On Linux systems no problems using
these options have been observed so far; however, on windows systems porgram
crashes and hang-ups did occur when using -O2 or -O3.
In windows, the inverse sqr tables will not work with -O2.
If unsure, compile all with -O1 only.

im3d im3d.o mpimod.o const.o init.o material.o target.o matrix.o index.o magic.o fileio.o cfgwriter.o mshwriter.o bulk.o utils.o random.o

$(CC) $(LDFLAGS) -o iran3d im3d.o mpimod.o const.o init.o material.o target.o matrix.o index.o magic.o fileio.o cfgwriter.o mshwriter.o bulk.o utils.o random.o $(LIBFLAG)

im3d.o: im3d.h im3d.c

mpimod.o: mpimod.h mpimod.c

const.o: const.h const.c

init.o: init.h init.c

material.o: material.h material.c

target.o: target.h target.c

matrix.o: matrix.h matrix.c

index.o: index.h index.c

magic.o: magic.h magic.c

fileio.o: fileio.h fileio.c

cfgwriter.o: cfgwriter.h cfgwriter.c

mshwriter.o: mshwriter.h mshwriter.c

bulk.o: bulk.h bulk.c

utils.o: utils.h utils.c

random.o: random.h random.c

.c.o :
$(CC) -c $(CFLAGS) $<

clean:
rm -f iran3d *.o

clear:
rm *.o

Then, you can just type "make" to compile the code:

make

Finally, an executable file "im3d" should be generated when the build is complete.

2.2.2 Command switch

The geometry modules are not open-source but can be called in the terms of static libraries, that is

struct_s.a, struct_m.a.

where s and m denote as serial and parallel version of the libraries, respectively. Thus, please switch 'LIBFLAG = -L. -lstruct_s' to 'LIBFLAG = -L. -lstruct_m', when use the MPI parallel version.

2.2.3 Errors that can occur when making IM3D

Link errors: all of the libraries used in makefile should be 32-bit version, including *gcc* and *mpich* libraries, etc.

2.3 Runing IM3D

IM3D can be run from a command-line options in a serial computer or a MPI parallel super computer.

serial-single-thread:

./im3d -command

serial-multi-threads:

./multithread.sh

MPI parallel:

e.g., *mpirun -n $cpu ./im3d -command*

2.4 Command-line Options

IM3D can be run from a command line, so that you can check the output. If no parameters are provided, IM3D assumes that there is a configuration file named Config.in in the current directory and loads the configuration from this file. However, you can provide various command line arguments (they are all optional):

-h Prints the help (basically like this table).

-l Display the license file.

-c FILENAME Instructs IM3D to use FILENAME as the configuration file instead of the default Config.in.

-p NUMBER Specify how much info to print to console. -2 means very little, 2 means a lot, with various possibilities in between.

-n NUMBER Default is 0. Specifies how many ions are simulated (overrides the setting speci- fied in the config file).

-E NUMBER Sets the energy of the ions. This option overrides the setting from the config file.

-w Wait for return key before exiting (useful if started from another program and you still want to see output).

-m -d Do not simulate anything, only estimate memory usage (roughly). Print details for memory usage (only useful when -m option also supplied).

-g STRING Generate and update status file while running. See details below.

The program mostly does not check whether the input parameters make any sense. In case some parameters are missing or completely out of range, the program will most likely crash or create strange results.

The -g option is useful when letting IM3D do background work for other programs. It instructs IM3D to output its current status to a file, which can be monitored by other programs (for example by a user interface). If it is set, every 200 ions IM3D writes exactly four lines to the file ir_state.dat. The first line contains the word IM3D and the version of IM3D. The second line contains the string submitted on the command line after the -g option. The third line contains a status word. The fourth line contains the number of ions that have been simulated. The status words and their meanings are:

init0 IM3D has started.

init1 The config files have been read.

init2 Everything is initialized and ready.

sim Simulation is running.

simend Simulation has finished.

end Simulation results have been stored and IM3D will terminate immediately.

2.5 IM3D Screen Output

As IM3D reads an input script, it prints information to the screen about significant actions it tanks to setup a simulation. When the simulation is ready to begin, IM3D performs various initializations and prints the feed-back information. It also prints the details of the initial geometry structure of the system. During the run itself, completed status is printed periodically, every constant ions numbers. When the run concludes, IM3D appends statistics about the CPU time approximately. An example set of the screen output is shown here:

```
**************************************************************************
IM3D: Ion IRadiation of Nanostructured materials
– a 3D parallel Monte Carlo simulation code

IM3D Version 1.0.0, Apr. 29th, 2014
–
by Yonggang Li (Y.G. Li), 2014, ygli@theory.issp.ac.cn & ygli@mit.edu;
Institute of Solid Status of Physics, Chinese Academy of Sciences;
& Nuclear Science and Engineering, Massachusetts Institute of Technology.

Refers to:
H.M. Li, H.Y. Wang, Y.G. Li & Z.J. Ding*'s CSG/FETM geometry methods;
& C. Borschel's OSS 'iradina' and F. Schiettekatte's OSS 'corteo'.

This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions, see LICENCE for details.
**************************************************************************

Seed in node 2 is: 39419295 93145296
Seed in node 1 is: 39419294 93145295
Seed in node 3 is: 39419296 93145297
Seed in node 0 is: 39419293 93145294
Configuration read from temp_configfile.im3d.
Corteo scattering matrix loaded.
Lists of random numbers generated.
Chu's straggling data read.
Invsere Erf list read.
Invsere Erf list randomized.
```

Prepare scattering matrices for ion on target collisions... finished
Prepare scattering matrices for recoil on target collisions...
i: 0; my_shape: 1; is_full: 1
i: 1; my_shape: 2; is_full: 1
i: 2; my_shape: 3; is_full: 1
i: 3; my_shape: 4; is_full: 1
i: 4; my_shape: 5; is_full: 1
i: 5; my_shape: 6; is_full: 1
i: 6; my_shape: 7; is_full: 1
i: 7; my_shape: 8; is_full: 1
i: 8; my_shape: 9; is_full: 1
z0_max_csg: 101 (nm), must larger than the max depth of goemetry structure!
i: 0; my_shape: 1; is_full: 1
i: 1; my_shape: 2; is_full: 1
i: 2; my_shape: 3; is_full: 1
i: 3; my_shape: 4; is_full: 1
i: 4; my_shape: 5; is_full: 1
i: 5; my_shape: 6; is_full: 1
i: 6; my_shape: 7; is_full: 1
i: 7; my_shape: 8; is_full: 1
i: 8; my_shape: 9; is_full: 1
z0_max_csg: 101 (nm), must larger than the max depth of goemetry structure!
finished
Materials read from temp_matfile.im3d.
Initializing target.
Target structure definition file: temp_structfile.im3d

Target size is:
x: 60 cells, 10 nm per cell, 600 nm in total.
y: 60 cells, 10 nm per cell, 600 nm in total.
z: 20 cells, 5 nm per cell, 100 nm in total.
Total: 72000 cells in 3.6e+07 nm$\hat{3}$.
———————————————-
i: 0; my_shape: 1; is_full: 1
Solid Sphere:
Radius(nm): 50.000
Center(nm): 100.000 100.000 50.000
i: 1; my_shape: 2; is_full: 1
Solid Tetrahedrona:

Vertex(nm): 300.000 110.000 0.100
300.000 50.000 100.000
250.000 140.000 100.000
350.000 140.000 100.000
i: 2; my_shape: 3; is_full: 1
Solid Cuboid:
Vertex(nm):
450.000 50.000 0.000
Side Length(nm):
100.000 100.000 100.000
Orientation:
1.000 0.000 0.000
0.000 1.000 0.000
0.000 0.000 1.000
i: 3; my_shape: 4; is_full: 1
Solid Ellipsoid:
Semi-axes(nm):
75.000 50.000 40.000
Shift(nm): 100.000 300.000 50.000
i: 4; my_shape: 5; is_full: 1
Solid Taper:
Vertex Angle(D): 30.000
Height(nm): 80.000
Shift(nm): 300.000 300.000 0.100
i: 5; my_shape: 6; is_full: 1
Solid Column:
Radius(nm): 50.000
Height(nm): 100.000
Shift(nm): 500.000 300.000 0.000
i: 6; my_shape: 7; is_full: 1
Solid Polyhedron:
6
4
50.000 425.000 100.010
50.000 575.000 100.010
59.963 575.000 0.000
59.963 425.000 0.000
4
59.963 425.000 0.000

59.963 575.000 0.000
144.963 575.000 0.000
144.963 425.000 0.000
4
144.963 425.000 0.000
144.963 575.000 0.000
154.926 575.000 100.010
154.926 425.000 100.010
4
50.000 425.000 100.010
50.000 575.000 100.010
154.926 575.000 100.010
154.926 425.000 100.010
4
50.000 425.000 100.010
59.963 425.000 0.000
144.963 425.000 0.000
154.926 425.000 100.010
4
50.000 575.000 100.010
59.963 575.000 0.000
144.963 575.000 0.000
154.926 575.000 100.010
i: 7; my_shape: 8; is_full: 1
Solid Paraboloid:
Radius(nm): 50.000
Height(nm): 100.000
Shift(nm): 300.000 500.000 0.000
i: 8; my_shape: 9; is_full: 1
Solid Hyperboloid:
Distance(nm): 20.000
Radius(nm): 50.000
Height(nm): 80.000
Shift(nm): 500.000 500.000 0.000
z0_max_csg: 101 (nm), must larger than the max depth of goemetry structure!

This is the CSG geometry version of im3d.

Target composition read from temp_compfile.im3d.

Target structure read from temp_structfile.im3d.
Normalization factor: 1

Starting simulation of irradiation...
i: 0; my_shape: 1; is_full: 1
i: 1; my_shape: 2; is_full: 1
i: 2; my_shape: 3; is_full: 1
i: 3; my_shape: 4; is_full: 1
i: 4; my_shape: 5; is_full: 1
i: 5; my_shape: 6; is_full: 1
i: 6; my_shape: 7; is_full: 1
i: 7; my_shape: 8; is_full: 1
i: 8; my_shape: 9; is_full: 1
z0_max_csg: 101 (nm), must larger than the max depth of goemetry structure!
Completed: 0%
Completed: 0.8%
Completed: 1.6%
...
...
...
Completed: 97.6%
Completed: 98.4%
Completed: 99.2%
Storing final results: ...
done.

Run time: 71.000000 seconds.

Chapter 3

Input

3.1 IM3D Input Script

IM3D input script is in the *iradina* format, which reads the input from four input files: the first file is a general configuration file describing how the program is supposed to run (contains mostly simulation parameters etc.). The second file describes the material properties of all materials found in the target. The third file defines the structure of the simulation volume. The fourth file holds the description of the 3D target (i.e. the shapes of CSG or FETM nanosized object).

The first three files look similar to other configuration file types (and are mostly human-readable): empty lines or lines starting with a #–sign are ignored. All other lines can either denote the beginning of a new section by [section name] or can contain a parameter definition: parname=value(s). Note that omitting any parameter might result in undefined behavior or crashes of the program. Upper and lower case letters cannot be exchanged! Recommendation: Use existing example files and adapt them according to your needs.

Alternatively, it is possible to put all the information from the four input files into one combined input file (Config.in), see details in section 3.1.5.

3.2 Input Script Structure

3.2.1 Configuration File: temp_configfile.im3d

The contents of the configuration file will be explained line-by-line. Detailed explanations can be found in *gen_config* tool (./tool/gen_configs).

1: # Configuration file for im3d

2: [IonBeam]

3: max_no_ions=50000

the number of protons of the ion, range of $(1, 10^8)$, generally 10^5 is enough and recommended;

4: ion_Z= 14

the mass in atomic mass units, range of $(1, 92)$;

5: ion_M=28.0

the atomic weight of ion;

6: ion_E0=50000

the primary energy of the ion in unit of eV, range of $(10eV, 2GeV)$;

7: ion_vx=0

8: ion_vy=0

9: ion_vz=1

the incident direction (vx, vy, vz) of the ion; range of $(0, 1)$;

10: ion_distribution=0

the distribution type of the ion: 0-random, 1-centered, 2-defined position, 3-random square around predefined position, 4-Gaussian beam, sigma = bean_spread;

11: enter_x=20

12: enter_y=20

13: enter_z=-10

the enter position (x, y, z) of ions when random beam is note selected, z must be higher than the target;

14: beam_spread=1.5

the spread of the beam in xy-space for Gaussian beam;

15: [Simulation]

16: OutputFileBaseName=./output/

output path, ./output/ is selected in default;

17: output_format=1

output_file format: 0-iradina, 1-cfg, 2-msh, 3-vtk;

18: normalize_output=0

normalize output results or not;

19: display_interval=100

the interval of display, range of $(1, max\_no\_ions)$;

20: storage_interval=1000

the storage of display, range of $(1, max\_no\_ions)$;

21: store_transmitted_ions=0

store transmitted ions or not;

22: store_exiting_recoils=0

store existing recoils or not;

23: store_exiting_limit=100

the maximum number of exiting recoils to be stored;

24: store_energy_deposit=1

array with deposited energy are created and stored or not;

25: store_ion_paths=0

store the exact ion paths or not, =1 only for the serial version;

26: store_path_limit = 100;

the maximum number of stored paths, range of $(1, 1000)$ is recommended, =-1 when store_ion_paths=0;

27: transport_type=1

transport type, =0 full and accurate projectile transport, =1 fast projectile transport;

28: multiple_collisions=1

the maximum number of multiple collisions, 0 means just 1;

29: flight_length_type=0

the flight length type: 0-¿Random Poisson dist., 1-¿Constant;

30: flight_length_constant=0.3

if flight_length_type=1, set its flight_length_constant;

31: scattering_calculation=0

the scattering calculation type: 0-¿SRIM-Corteo database, 1-¿MAGIC approximation;

32: tracing_recoil_or_not=1

tracing the exact recoils cascades or not, 0-KP, 1-FC;

33: store_recoil_cascades=0

store the exact recoils cascades or not;

34: detailed_sputtering=1

detailed calculation of sputtering or not;

35: single_ion_sputter_yields=0

if detailed_sputtering=1, store sputter yields for single ions or not;

36: do_not_store_damage=0

store damages or not;

37: min_energy=5

the minimum energy below which all projectiles are stopped, range of $(0, ion\_E0)$;

38: seed1=39419293

39: seed2=93145294

random seeds, 8-digit integers.

40: [Target]

41: geometry_type=1

the geometry type: 0-¿bulk, 1-¿csg, 2-¿fetm;

42: no_substrate=1

with substrate or not;

43: gen_shape_or_not=1

if geometry_type == 2, generate fetm shape from ply2 file by IM3D or pre-generated fetm shape by triangle.f90 code;

44: straggling_model=3

the straggling model: 0-¿No straggling, 1-¿Bohr, 2-¿Chu, 3-¿Chu+Yang;

45: MaterialsFileName=Materials.in

filename that defines the materials in the target, in default;

46: TargetstructureFileName=Structure.in

filename that define the structure of the target, in default.

3.2.2 Materials File: temp_matfile.im3d

1: [GaAs]

name;

2: element_count=2

the number of elements in the material;

3: density=4.43e22

the density of the material;

4: elements_Z=31,33

the atomic numbers of the elements in the material;

5: elements_M=69.72,74.92

the atomic weights of the elements in the material;

6: elements_conc=0.5,0.5

the atomic contents of the elements in the material;

7: elements_dispEnergy=20.0,25.0

the displacement energies of the elements in the material;

8: elements_latt_energy=3.0,3.0

the bulk lattice energies of the elements in the material;

9: elements_surf_energy=2.0,1.2

the surface lattice energies of the elements in the material;

10: ion_surf_energy=2.0

the surface lattice energies of the ion in the material.

3.2.3 Structure File: temp_structfile.im3d

1: # Structure definition file for im3d

2: [Target]

3: cell_count_x=60

4: cell_count_y=60

5: cell_count_z=20

the numbers of cells along (x, y, z)-axis, respectively;

6: cell_size_x=10

7: cell_size_y=10

8: cell_size_z=5

the intervals of cells along (x, y, z)-axis, respectively;

9: sub_surf_z=101

the z-position of the substrate surface;

10: CompositionFileType=0

only used for iradina file type: 0-¿one column, 1-¿four column;

11: CompositionFileName=testwire.conc.in

filename that defines the composition in the target, in default.

3.2.4 Composition File: temp_compfile.im3d

IM3D has two different types of composition files related to CSG and $FETM$ methods, respectively.

For CSG method, explanations can be found in in gen_config tool (./tool/gen_configs) in detail. In the composition file it includes:

```
 1        # Sphere
1
100.000 100.000  50.000  50.000
0
0
2        # Tetrahedron
```

```
1
300.000 110.000   0.100
300.000  50.000 100.000
250.000 140.000 100.000
350.000 140.000 100.000
0
0
3       # Cuboid
1
450.000  50.000   0.000
100.000 100.000 100.000
1.000   0.000   0.000
0.000   1.000   0.000
0.000   0.000   1.000
4       # Ellipsoid
1
75.000  50.000  40.000
-1
100.000 300.000  50.000
0
0
0
5       # Taper
1
30.000  80.000
-1
300.000 300.000   0.100
0
0
0
6       # Column
1
50.000 100.000
-1
500.000 300.000   0.000
0
0
0
7       # Polyhedron
```

```
1
6
4
50.000 425.000 100.010
50.000 575.000 100.010
59.963 575.000   0.000
59.963 425.000   0.000
4
59.963 425.000   0.000
59.963 575.000   0.000
144.963 575.000   0.000
144.963 425.000   0.000
4
144.963 425.000   0.000
144.963 575.000   0.000
154.926 575.000 100.010
154.926 425.000 100.010
4
50.000 425.000 100.010
50.000 575.000 100.010
154.926 575.000 100.010
154.926 425.000 100.010
4
50.000 425.000 100.010
59.963 425.000   0.000
144.963 425.000   0.000
154.926 425.000 100.010
4
50.000 575.000 100.010
59.963 575.000   0.000
144.963 575.000   0.000
154.926 575.000 100.010
8       # Paraboloid
1
50.000 100.000
-1
300.000 500.000   0.000
0
0
```

```
0
9       # Hyperboloid
1
20.000  50.000  80.000
-1
500.000 500.000   0.000
0
0
0
0
0.0
```

For FETM method, in the composition file it includes:

```
50.0  50.0  122.0       # box_start_x0, box_start_y0, box_start_z0;
11  11  11              # box_count_x, box_count_y, box_count_z ;
10.1  10.1  10.1        # box_size_x, box_size_y, box_size_z ;
# space subdivision for FETM targets;
2                       # file format, 1-file1.dat generated by opengl, 2-file1.ply2 generated
by Gmsh;
1                       # the number of materials or file1s;
1  12                   # material_type, number_of_data
100.0  100.0  100.0     # the scaling of the target, scaling_x, scaling_y, scaling_z;
0.0  0.0  0.0           # the transformation of the target, trans_x, trans_y, trans_z.
```

And in another input file (file1.ply2), shapes are described like (take a cube as example):

```
8
12
0 0 0
1 0 0
0 1 0
1 1 0
0 0 1
1 0 1
0 1 1
1 1 1
3 0 1 3
3 0 2 3
```

3 0 4 6
3 0 2 6
3 0 1 5
3 0 4 5
3 2 6 3
3 6 7 3
3 1 3 5
3 3 5 7
3 5 6 7
3 4 5 6

3.2.5 Combined Input File

You can also put all the information from the four input files into one combined input file, i.e., Config.in. Before running, Config.in will automatically spits into four temporary input files: temp_configfile.im3d, temp_matfile.im3d, temp_structfile.im3d and temp_compfile.im3d.

Chapter 4

Output

All of output files can be found in the path of

./output/.

4.1 List of Output Files

In the output path, there would be a list of output files as defined in input script, i.e.,

- aiv.xyz.cfg
- cascades
- depth_dist_functions.dat
- disp.mat∗.cfg, disp.mat∗.msh or disp.mat∗.vtk
- energy.deposit.cfg, energy.deposit.msh or energy.deposit.vtk
- int.mat∗.cfg, int.mat∗.msh or int.mat∗.vtk
- ion_paths
- ions.replacements.cfg, ions.replacements.msh or ions.replacements.vtk
- ions.total.cfg, ions.total.msh or ions.total.vtk
- leaving.mat∗.cfg, leaving.mat∗.msh or leaving.mat∗.vtk
- leaving_directions.ions
- leaving_directions.sum

- leaving_directions.z∗1.m∗2.mat∗3.elem∗4
- radial_dist_functions.dat
- repl.mat∗.cfg, repl.mat∗.msh or repl.mat∗.vtk
- transmitted.ions
- vac.mat∗.cfg, vac.mat∗.msh or vac.mat∗.vtk

4.1.1 aiv.xyz.cfg

aiv.xyz.cfg is an output file of the detailed position of ions and defects generated in materials, which includes 7 columns, i.e., x, y, z, defect type, defect belong to which ion, material and element. It is in the format of .cfg at present and can be viewed by AtomEye software directly. An example of this output file is list below:

Number of particles = 801809
A = 10 Angstrom (basic length-scale)
H0(1,1) = 1000 A
H0(1,2) = 0.0 A
H0(1,3) = 0.0 A
H0(2,1) = 0.0 A
H0(2,2) = 1000 A
H0(2,3) = 0.0 A
H0(3,1) = 0.0 A
H0(3,2) = 0.0 A
H0(3,3) = 1000 A
.NO_VELOCITY.
entry_count = 7
auxiliary[0] = type (A-I-V)
auxiliary[1] = tab (A-ion)
auxiliary[0] = mater
auxiliary[1] = element
4
aiv_pos
0.500743 0.500435 0.001079 2 0 1 0
0.500792 0.500568 0.001067 1 0 1 1
0.500740 0.500452 0.001087 2 0 1 1
0.500473 0.501270 0.001831 1 0 1 0
0.500484 0.500152 0.001079 2 0 1 0

0.500476 0.499380 0.001042 1 0 1 1

... ...

4.1.2 Output: cascades

4.1.3 Output: depth_dist_functions.dat

4.1.4 Output: disp.mat∗.cfg/.msh/.vtk

4.1.5 Output: energy.deposit.cfg/.msh/.vtk

4.1.6 Output: int.mat∗.cfg/.msh/.vtk

4.1.7 Output: ion_paths

4.1.8 Output: ions.replacements.cfg/.msh/.vtk

4.1.9 Output: ions.total.cfg/.msh/.vtk

4.1.10 Output: leaving.mat∗.cfg/.msh/.vtk

4.1.11 Output: leaving_directions.ions

4.1.12 Output: leaving_directions.sum

4.1.13 Output: leaving_directions.z∗1.m∗2.mat∗3.elem∗4

4.1.14 Output: radial_dist_functions.dat

4.1.15 Output: repl.mat∗.cfg/.msh/.vtk

4.1.16 Output: transmitted.ions

4.1.17 Output: vac.mat∗.cfg/.msh/.vtk

4.2 Output Format

Three types of output formats are given in IM3D code till now, including *.cfg*, *.msh* and *.vtk*.

4.2.1 *.cfg* format

4.2.2 *.msh* format

4.2.3 *.vtk* format

4.3 Output Visualization

Different kinds of softwares can be feasibly used to visualize the 3D distributions of primary defects in output files, such as, AtomEye, Gmsh and ParaView, etc. AtomEye software is a easy tool to quickly visualize the *.cfg* format files. While for much higher qualities, Gmsh or ParaView is commented to be used with the output files in the formats of *.msh* and *.vtk*, respectively.

Chapter 5

Accelerating IM3D Performance

5.1 General Strategies

In order to further increasing the efficiency of the code, we introduce the fast database indexing technique proposed by Schiettekatte in Corteo code[2] to sampling the scattering and azimuthal angles and the stopping powers as well as two numerical acceleration algorithms to implement parallel computing.

5.2 Fast Database Indexing Techniques

For the calculation of the classical binary atomic scattering angle, IM3D code thus introduce the fast routines from Corteo which are based on both the fast indexing technique and the MAGIC approximation[1] alternatively in terms of accuracy, efficiency and memory usage. For the calculation of the stopping power, the same algorithm of computing the classical binary atomic scattering angle can also be used by firstly generating the tables of stopping power values and directly finding a desired value form these tables in use. For the case of the number of elements included in a target is not too big, the memory burden is no more than several MB. The detailed treatment of the fast indexing techniques can be found in Ref.[2, 9].

5.2.1

5.2.2

5.2.3

5.3 MPI Parallel and Multi-threading

Furthermore, in order to further enhance the computational efficiency, the MPI parallel and multi-threading methods have been integrated into IM3D code. The code can be easily parallelized by dividing the number of incident ions and offer an almost linear speed-up ratio with the number of processors. While the multi-threading technique can be implemented by just using a shell in-script, which is much feasible for the code running on a multi-core platform.

5.3.1 MPI Parallel method

5.3.2 Multi-threading method

5.4 Measuring Performance

Based on the above acceleration techniques, a typical simulation of 10^5 ions with energy of keV to MeV consumes only seconds to minutes in a modern serial computer even for complex 3D structures, and would be more faster when using the parallel or multi-threading version in a super computer. Generally, IM3D code is faster than TRIM code by at least two to three orders of magnitude, depending on the simulation parameters and the acceleration techniques.

5.4.1

5.4.2

Chapter 6

Example Problems

The objective of IM3D code is to accurately and quickly calculate the 3D space-distributions of primary radiation damages in arbitrary-complex geometric targets containing different shapes and components, including electronic and nuclear energy depositions, backscatted/implanted ions, *dpa*, interstitials, vacancies and sputtering atoms, etc. In the following, some verifications and examples are given to demonstrate the validity and capability of IM3D code.

6.1 Verifications

To verify the accuracy of IM3D code, three examples including the ion/damage depth-distributions in bulk/multi-layer targets are performed and compared with that of TRIM code and experiments. Furthermore, some confinements in using TRIM-like codes are discussed and clarified.

As shown in Fig.6.1, the depth-distributions of ion deposition are calculated for Si bulk under Ar ion beam irradiation with different irradiation energies. A typical Gaussian-type profile of ion depth-distributions are obtained, which is in good agreement with that of TRIM code for all three ion energies, even for the absolute intensity values.

The vacancy depth-distributions for Ni bulk under He ion beam irradiation with different incident energies are given in Fig.6.2. Again, good agreements between IM3D and TRIM codes are obtained for all three ion energies. The intensity ratios between the predictions of the FC and KP methods is around 2 as estimated by TRIM code[14]. By comparing to the standard reference values estimated by MD and NRT model, Stoller et al. pointed out that there could be a fundamental problem in the SRIM model used to calculate the number of vacancies created[14]. Borschel et al. also found an obvious discrepancy from

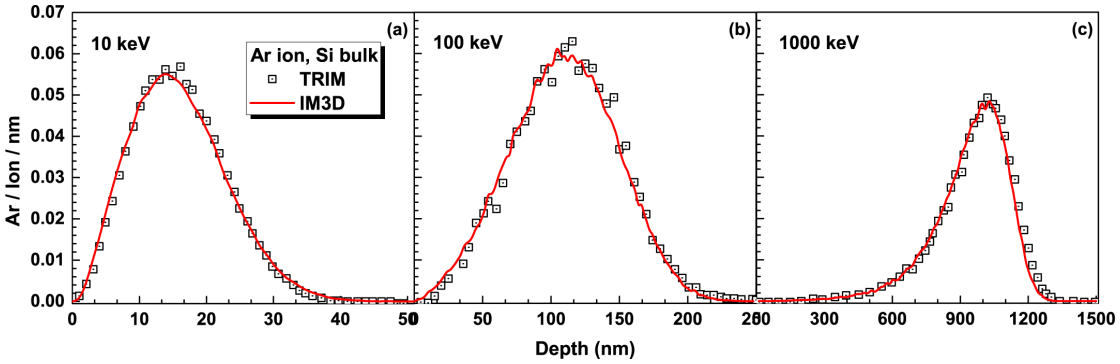

FIGURE 6.1: Comparison of IM3D results with TRIM predictions for Ar ion depth-distributions in Si bulk, under Ar ion implantation with different energies of (a) 10 keV, (b) 100 keV and (c) 1000 keV and normal incidence at the center point of the target.

Iradina code to TRIM code for the damage production within collision cascades[9]. Since TRIM is not an open-source software, the certain fine details of its defect generation algorithms are not clearly described in both SRIM's manual and published papers[1]. Generally, this discrepancy should be due to the difference of the replacement fractions in the total displacement events which are determined only by a sharp cut-off threshold energy[9]. This threshold energy is usually set as the displacement energy (E_d) as given in SRIM's manual and elsewhere[1, 9]. While the replacement process should occur only when the energy of the trajectory atom after replacement is lower than the binding energy (E_b) in bulk. Otherwise, the trajectory atom after replacement will leave the lattice site and migrate to other position, which generate a pair of interstitial and vacancy. Thus, by using the binding energy E_b instead of the displacement energy E_d as the threshold of replacements, we directly reproduced the exact depth profiles with the same absolute values and also the same vacancy intensity ratio between the FC and KP methods. In addition, dynamic annealing process should occur at finite temperatures and the larger number of primary point defects at 0 K predicted by the FC method would finally decay to MD, KP or experimental values.

People found that the range of defect depth-distributions calculated by SRIM package are usually shallower than that of experiments more or less, which generally attributed to an overestimation of the electronic stopping powers used in SRIM package (especially for low-energy heavy ions)[15–17]. A reduced target density is simply employed to compensate for the overestimation of electronic stopping power and thus to reduce the discrepancy[15]. Through setting the target density lower, good agreements are obtained comparing to TRIM code and experimental results for 305 nm ZrO2 film on Si bulk under 2.0 MeV Au ion irradiation, as shown in Fig.6.3. However, artificially changing of target densities is just a phenomenological treatment with no sound physical basis. Other influences should also be considered to understand this range discrepancy. When ion fluence is high enough, the defect depth-distributions would be physically changed by

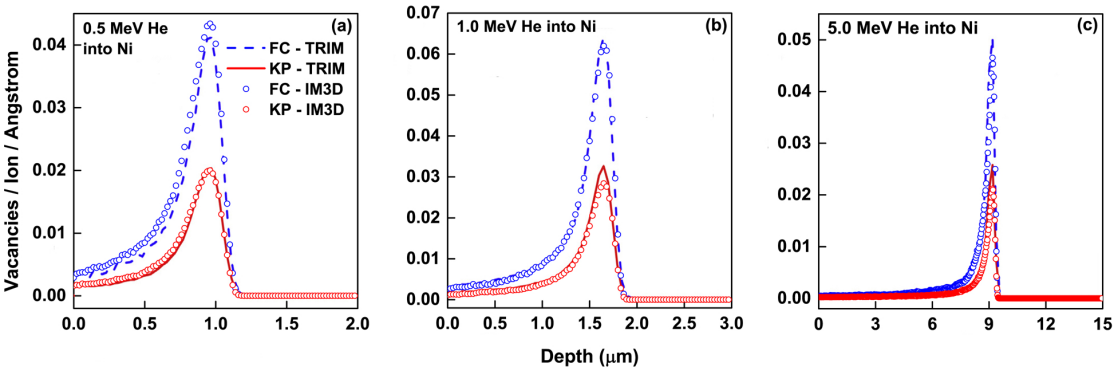

FIGURE 6.2: Comparison of IM3D results with TRIM predictions for vacancy depth-distributions predicted by FC and KP methods in Ni bulk, under He ion irradiation with different energies of (a) 0.5 MeV, (b) 1.0 MeV and (c) 5.0 MeV and normal incidence at the certer point of the target.

the evolution of the target density due to ion irradiation. While when ion fluence is not too high, the change of the target density in experiments can be neglected. Moreover, the profiles can further broaden into depth to approach to experiments when taking into account of the diffusion and reaction effects as discussed in our previous paper[18]. Therefore, it is not only the change of target densities (or the electronic stopping powers) but most of all the diffusion and reaction effects cause the discrepancy, because the temperature in practice is finite other than $T = 0\ K$ set in TRIM-like codes. The diffusion and reaction effects become prominent especially for the case of ions with much high diffusivity in the target[18]. These effects are non-equilibrium dynamic processes taking account of temperature and time, which should be described by the meso-scale simulation methods such as the kinetic Monte Carlo (KMC) and cluster dynamics (CD) models.

6.2 Two effects: nano-size and geometric effects

As the ion range or dimension of the collision cascades becomes comparable to the size of the nanoobject itself, the high surface-volume ratio of nanostructured materials will induce two new effects, i.e., the so-called nano-size effect and geometric effect.

Due to the surface reconstruction of nanostructure materials, the thermodynamic properties change with the size reduction, which cause the nano-size effect. In Fig.6.4(a), the amount of defects along with irradiation energies calculated by IM3D code has the similar trend to that of the analytical model and MD simulation (with the maximum amount produced at energies around 3 keV), while is smaller than that of MD simulations. This discrepancy should come from the nano-size effects like the difference of

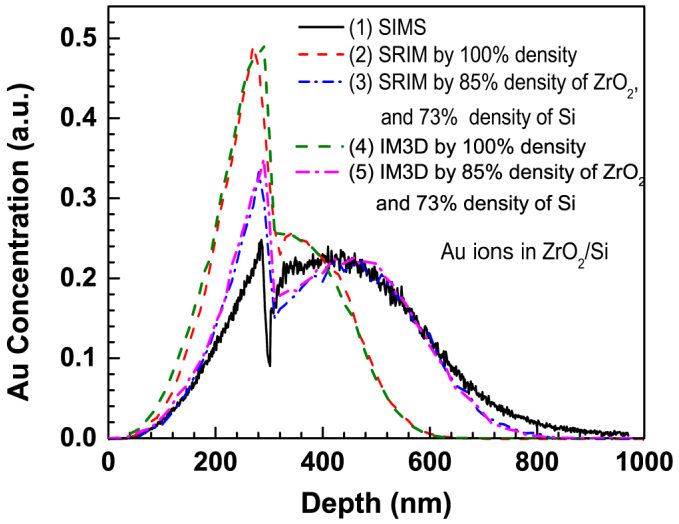

FIGURE 6.3: Comparison of IM3D results with TRIM predictions and SIMS measurement[15] for Au ion depth-distribution in ZrO_2/Si sample, under Au ion irradiation with the energy of 2.0 MeV and normal incidence at the center point of the sample. IM3D and TRIM predictions under the assumption of the reduced density are also given, where 15% reduced density for ZrO_2 (4.6 g/cm^3) and 27%reduced density for Si (1.7 g/cm^3) are used.

stopping powers and the decrease of energy thresholds (i.e., the displacement and binding energies) between nanostrucutres and bulk, etc. While these energy thresholds are usually lower with the size reduction of nanostructures and follow a universal relation as predicted for the cohesive energy of nanoparticles as in Ref.[19]. It would underestimate the total number of defects for both 3 and $4-nm$ nanowires if the bulk energy thresholds are used for determining the capability of defect generation. This viewpoint can just be proved by using the half values of bulk energy thresholds to decrease the discrepancy between IM3D and MD simulations for the $3-nm$ nanowire. The interstitials, vacancies and sputtering atoms are directly relate to the energy thresholds (especially the surface/lattice binding energies), which are influenced by the nano-size effect mostly. The discrepancy between IM3D and MD simulations thus becomes more obvious when irradiation energy is larger (see Fig.6.4(b)).

In IM3D code, we consider the bulk parameters are valid at least when the target size is larger than $\sim$ 10 nm, because in this case the thermodynamic properties close to constants and the nano-size effect is vanishing[20]. For the target size smaller than $\sim$ 10 nm, IM3D code can also be employed to estimate the primary radiation damages at the first approximation by using a set of modified parameters considered the nano-size effects, while MD method would prefer to be used to give a more accurate estimation because of the acceptable calculation consuming. Furthermore, Fig.6.4 (b) shows that both the nano-size and geometric effects determine the number of defects, that is, the amount of different types of defects for the flat case is higher than that of edge due

to the geometric effect. The difference between the flat and edge cases is smaller than that of MD results[21], because no channeling effect is included in IM3D code for an amorphous target.

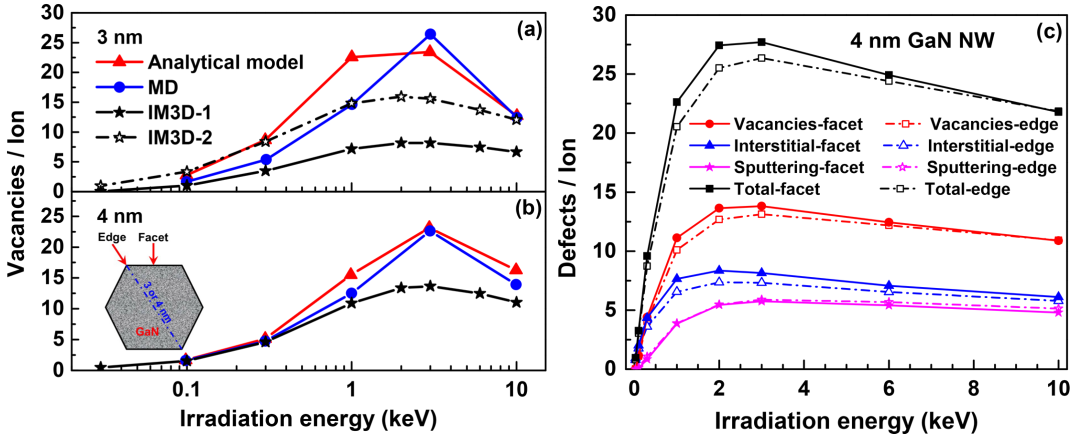

FIGURE 6.4: (a) Comparison of the number of vacancies in GaN nanowire along with Ar ion energies predicted by IM3D code, the analytical model based on TRIM code and MD simulations. The curves for the analytical model are artificially scaled to have the same area as the ones obtained from MD simulations, for the absolute amount of damage cannot be estimated reliably from the results based on SRIM calculations[21]. IM3D-1 and IM3D-2 are corresponding to the bulk energy thresholds and half values of the bulk thresholds used in the simulations, respectively. (b) The respect and total numbers for three different types of defects (vacancies, interstitials and sputtered atoms) along with Ar ion energies, for the 3 and 4 *nm*-diameter GaN nanowire under Ar ion irradiation at the edge and facet of GaN nanowire.

The geometric effect will influence the trajectory of an ion when it transport through the interface between two different material zones, such as trajectory emission, re-entering, sputtering and shading, etc. In order to illustrate the geometric effect contrast to the bulk counterpart, we simulate the vacancy depth-distributions in Au column target under two types of ion beams (i.e. center and random incidence), as shown in Fig.6.5. The ranges of vacancy depth-distributions and the total amount of vacancies are both increase with the increasing of column diameters, and finally approach to the bulk values after the critical diameters (around 200 *nm* and 1000 *nm* for center and random ion incidence, respectively).

Assumes that the radial distribution of damages (e.g. vacancies) $I(r)$ produced by an ion in a half-infinite bulk target follows a exponential decay (as shown by the inserted figure in Fig.6.5(a)),

$$I(r) = I_0 e^{-r/t}, \quad r \in [0, \infty), \tag{6.1}$$

where, I_0 is the intensity of damages at $r = 0$ and t is the effective attenuation length of damages in radial direction (25 *nm* for center ion incidence). The amount of damages

in a nanowire for the center ion beam incidence can be obtained by integrating Eq.(6.1) over the volume of the antisymmetric column (see Fig.6.5) that,

$$N_d = \int_0^R \int_0^{2\pi} I(r)\, dr d\theta = \frac{2\pi I_0}{t}\left(1 - e^{-R/t}\right), \tag{6.2}$$

where R is the radius of the column.

Similarly, we can give the analytical function for the random ion beam incidence, by additionally considering the effective attenuation range t (32 nm for random ion incidence) of the spread ion beam. For $R \leq t$, we assumed that the amount of damages also follows Eq.(6.2) approximately but with $t = 32\ nm$ instead. While for $R > t$, only ions bombarding in the effective attenuation range t from the colume side with the area ratio of $\left(\frac{2t}{R} - \frac{t^2}{R^2}\right)$, can cause a half loss of defects with the ratio of $0.5 \cdot \left(1 - \frac{1}{e}\right)$ as estimated approximately from Eq.(6.2) with $R = t$. Otherwise, no loss of the defects when ions bombarding in the central zone $(0,\ R - t)$ of the column. Thus, the amount of damages follows that,

$$N_d = \frac{2\pi I_0}{t}\left[1 - 0.5 \cdot \left(\frac{2t}{R} - \frac{t^2}{R^2}\right)\left(1 - \frac{1}{e}\right)\right],\ R > t. \tag{6.3}$$

It can be found as shown in Fig.6.5 that these two simple analytical estimations can fit the calculated vacancy-diameter relation much well, which directly illustrates that the geometric effect is the main factor to the distributions of primary radiation damages in nanostructured materials.

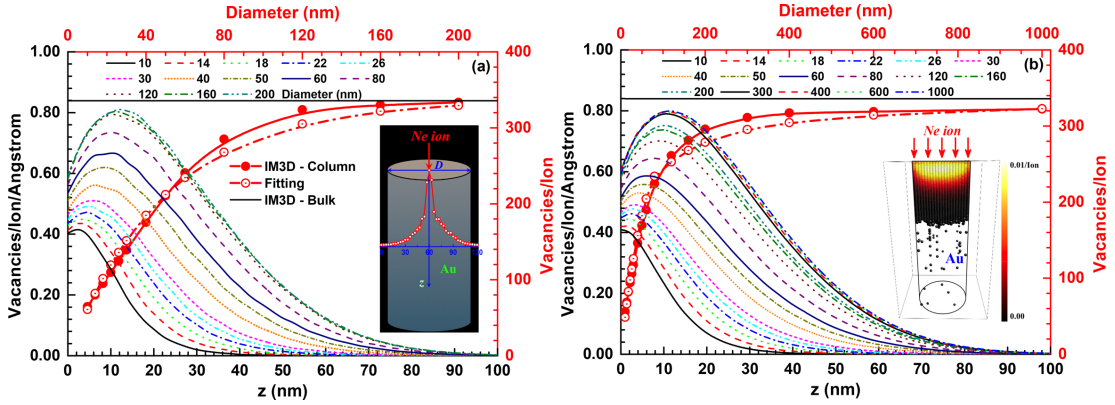

FIGURE 6.5: The depth-distributions and the total number of vacancies for Au nanowaire with different diameters under 45 keV Ne ion irradiation with the (a) center and (b) random normal-incidence beams. Eqs.(6.2) and (6.3) are plotted to fit the relations of the total number of vacancies with different diameters. The inserted figures in (a) and (b) show the vacancy radial-distribution in Au bulk and the vacancy space-distribution in Au nanowire with the diameter of 120 nm, respectively.

Column is a typical model of a class of nanostructured materials, such as nanowire, nanoporous and "fuzz". Bringa et al. performed a serial of experimental studies on the anti-irradiation behaviors of nanoporous materials and proposed an anti-irradiation window by simply considering of the effective vacancy diffusivity and lower critical size due to melting. However, they did not consider the contribution from the change of primary radiation damages by the geometric effect and just assumed that the primary radiation damages is a constant value. As shown here, the geometric effect can introduce a dramatical reduction of the primary radiation damages, which should be taken into account in the estimation of the anti-irradiation window. Furthermore, by considering the primary damage distributions as well as the diffusion and reaction effects at $T > 0\ K$, CD model would be performed to give the more accurate anti-irradiation windows of nanostructure materials in the future.

6.3 Applications

6.3.1 Arbitrary complex targets based on CSG and FETM methods

To our knowledge, IM3D is the most universal and robust code for simulating of the primary radiation damages in arbitrary complex targets with different shapes and components till now. Based on CSG geometric model, nine basic shapes (in Fig.6.6(a)) and their assemblies (in Fig.6.6(b)) can be constructed at present to model many regular targets with different materials. The geometric effect makes the distinctions of the defect space-distributions among different shapes.

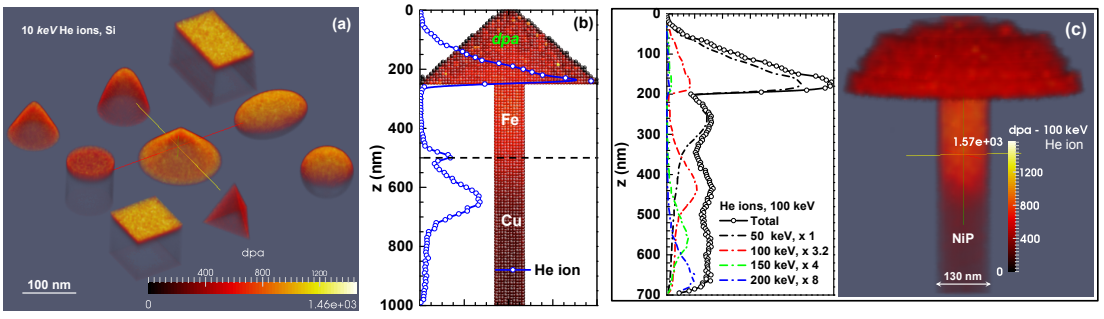

FIGURE 6.6: (a) *Dpa* space-distributions of nine basic shapes based on CSG geometric method and composed of Si under 10 *keV* He ion irradiation with the random normal-incidence beam. (b) *Dpa* space-distribution of for a 100 nm-diameter Cu/Fe nano-bicrystal constructed by CSG geometric method under 200 *keV* He ion irradiation with the random normal-incidence beam. The insert scatter line is the corresponding He ion depth-distribution. (c) The respect and total He ion depth-distributions for different energies He ions (left) and the *dpa* (right) space-distribution for 100 *keV* He ions with the random normal-incidence beam irradiated on a 130 *nm* NiP metallic glass nanostructure constructed by FETM geometric method.

By introducing the geometric effect, more reliable information can be obtained when estimate the defect (such as ion and *dpa*) distributions. For bulk/multi-layer systems, people usually employ TRIM code to give the first approximation of ion/dpa distributions. However, for complex nanostructures as mentioned in Refs.[22, 23], errors would be introduced when using TRIM code. As shown in Fig.6.6(b), more practical *dpa* space-distribution is obtained for a Cu/Fe nano-bicrystal based on CSG geometric model. For NiP metallic glass nanostructures (in Fig.6.6(c)), more accurate ratio of ion fluence can be estimated to generate a uniform distribution damage based on FETM geometric model, with $1 : 3.2 : 4 : 8$ instead of $3.5 : 4.0 : 2.8 : 5.5$ in Ref.[22]. Thus, the geometric effect can dramatically change the behaviors of ion irradiation and can not be neglected.

6.3.2 Nano-yttria in ODS steels under ion-irradiation

Recently, ODS steels (with a great mount of yttria (Y_2O_3) nanopartiles embedded in a steel matrix) have been termed as a new class of high anti-irradiation and high strength nanostructured nuclear materials[24–29]. However, the high radiation resistant mechanisms and the role of the embedded nanoparticles are still unclear and sometimes controversial. Moreover, the impact of primary damage processes on their high radiation resistant properties should also be discussed in detail. Here, we assumes a perfect iron matrix in which Y_2O_3 nanoparticles are embedded as a simplified model of an ODS steel and simulate the space-distributions of the primary radiation damages with the displacement energies of 40 eV[1] for Fe and 57 eV for Y and O[29].

Based on IM3D code, the ion trajectories and 3D space-distributions in a target of Fe matrix with a void and two Y_2O_3 nanoparticles of different sizes embedded can be given as shown in Fig.6.7. Spheres with different materials/vacuum can change ion trajectories (Fig.6.7(a)) by influencing the energy losses and the production rate of defects, and finally induce the different damage distributions (in Fig.6.7(b)). Comparing to bulk regions, ions in void transport straightly without energy loss and penetrate into depths to generate more damages in the deep (i.e., the enhancement effect, similar to the MD results**??**). Similarly, ions can also loss less energy and generate less damages in Y_2O_3 nanoparticles. Thus, Y_2O_3 nanoparticles are play the same role as voids to suppress the production of primary radiation damages but without losing the strength of the steel. The anti-irradiation of Y_2O_3 nanoparticles is mainly due to the lower defect generation rate with their much lower atomic density (1.10934×10^{22} $atoms\ cm^{-3}$) comparing to that of the steel matrix (8.388×10^{22} $atoms\ cm^{-3}$), the higher displacement energies of Y and O (57 eV) to that of Fe (40 eV), as well as the attraction of defects to the nanoparticle interface during annealing. The enhancement effect of *dpa* intensity behind the void and the bigger-size Y_2O_3 nanoparticle comes from the overlapping of

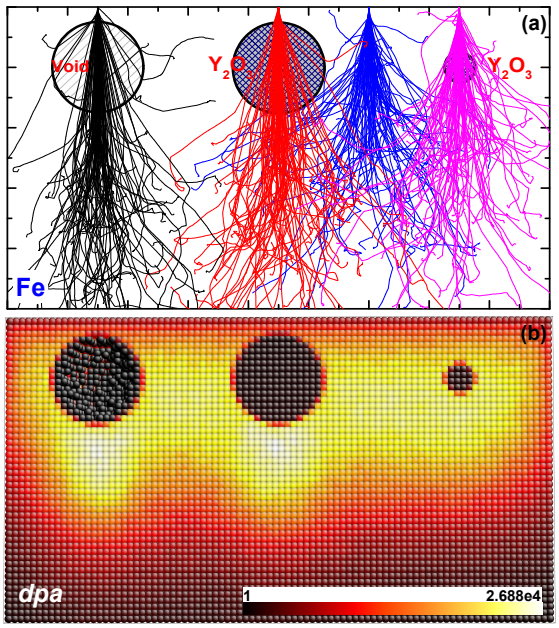

FIGURE 6.7: The (a) ion trajectories and (b) *dpa* distribution cross-section for a target of Fe matrix with a void (30 nm) and two Y_2O_3 nanoparticles (30 nm and 10 nm) of different sizes embedded under 150 keV Fe ion irradiation with the random normal-incidence beam.

the *dpa* generated by the penetrating ion though the void / nano-particle and the ion directly transporting in iron matrix nearby, as shown in Fig.6.7(a). This enhancement effect would induce a little more serious primary damages to Fe bulk matrix, which introduce a negative effect on the radiation resistant. While the enhancement effect can be reduced/removed by decreasing the diameter of Y_2O_3 nanoparticles from 30 nm to 10 nm as shown in Fig.6.7(b), which has also been proved by the high resolution TEM measurements[29]. The high sink strength of the interface between Y_2O_3 nanoparticles and Fe matrix would also neutralize the enhancement effect and finally reduce the total damages in ODS steel after annealing.

6.3.3 Ion beam sputtering induced the bending of W nanowire

Under ion beam irradiation with high fluence, nanowires have been observed to bend towards and finally align with the ion beam[30]. The primary radiation damages should play an important role on this effect, which can also be studied by IM3D code feasibly. By generating a target with 3D surface mesh with FETM geometric method beforehand (in Fig.6.8(a)) and inputting it into IM3D code, the 3D distribution of defects can be given such as vacancies for a bended W nanowire under the irradiation of Ga ion with the incident direction of 40 degrees and the energy of 150 keV, as shown in Fig.6.8(b). More primary vacancies generated on the side towards to the ion beam, making the density on this side lower than the other side and thus bend the column line. During

the ion irradiation of W nanowire at finite temperatures, most of interstitials with a low migration energy (0.013 eV[31]) would anneal with vacancies immediately, diffuse to the other side or deposit directly. While a little part of vacancies without annealing by interstitials are nearly immobile and stay constant with a much higher migration energy (1.66 eV[32]). Here, in order to consider the final remaining damages in W nanowire, we use the difference value of vacancies minus interstitials by assuming annihilation of defects only occur in each cell ($10 \times 10 \times 10$ nm^3). An inhomogeneous distribution of the remaining defects is given in Fig.6.8(c), where excess vacancies remain on the side towards to the ion beam and excess interstitials remain at the opposite side. Thus, a bending momentum (under inner stress due to an inhomogeneous expansion of the nanowire) towards the ion beam is induced to compensate the density difference until the the direction of the column line along the direction of ion beam, as observed in the experiment[30]. Moreover, the shading/shadowing effect (shading of the ions on their incident path by a particle leads to a decrease of damages behind the particle) can also be seen in Fig.6.8(b) with dark area in the substrate appear behind the nanowire. During the plasma-surface interaction (PSI) process in PFMs (as shown below), these two typical effects (i.e., the bending and shading effects) should also occur at the reconstructed surface, causing the formation of complex "fuzz" nanostructure finally.

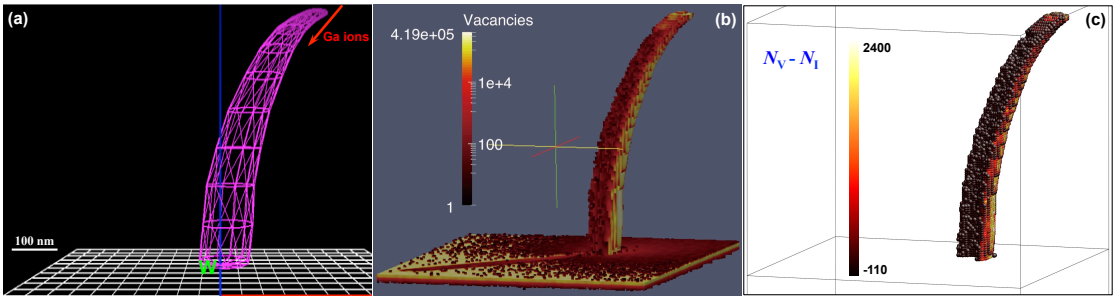

FIGURE 6.8: The (a) 3D surface mesh with FETM geometric method, (b) vacancy space-distribution and (c) remaining excess vacancies for a bended W nanowire under randomly distributed Ga ion sputtering with the incident direction of 40 degrees and the energy of 150 keV.

6.3.4 D retention in W with roughness surface

Plasma-surface interaction (PSI) is one of the most important issue in nuclear fusion instruments. Under low energy (in the range of $10-100$ eV), high flux (up to the order of 10^{24} $m^{-2}s^{-1}$) D/T/He plasma loads in ITER[33], the near surface morphology of PFMs will be dramatically changed and reconstructed to roughness or even more complex nanostructures like "fuzz". These roughness nanostrucutes would further increase the retention of D/T/He. While T retention is also another key problem need to be widely studied. Thus, the calculation of the primary retention of H isotopes in W with roughness

structures is much helpful for the understanding of the mechanisms of surface damage and T retention in PFMs.

In Fig.6.9, we performed the W bulk with different rough-surface under D ion irradiation. A simple geometric model based on FETM method are used here to simulate the roughness structures[4]. The rough-surface is constructed in a finite element triangulated mesh by using a Gaussian function to describe the distribution of the amplitude (3σ) of the random roughness peaks as well as a uniform square mesh with a lattice constant ($a = 50\ nm$) to describe the density of the roughness peaks. The depth-distributions of D in W are given for different rough amplitudes (3σ) from 9 to 1000 nm (in Fig.6.9(b)). We found that the range of D ions in W increase with the increasing of rough amplitude. The distributions relate to Gaussian distribution directly, which can be approximated by the convolution of the effective interaction range and the geometric distribution of rough-surface, as shown in Fig.6.9(b). Also, the geometric and shading effects dominate the behavior of D retention in W bulk with roughness surface.

The relation of D retention rate with the rough amplitude is also given in Fig.6.9(c). With the increasing of the rough amplitude, D retention rate decreases at first and then increase after a critical point at about $3\sigma = 50\ nm$. It should be due to the competition of the enhancements of both the backscattering at glancing incidence and the shading effect by rough peaks. In order to describe the exact relation of the retention rate with the geometric and shading effects, a simple model is introduced to fit this relation. For the amplitude and distance of the roughness peaks are set as 3σ and a, the effective incident angle of ions related to rough-surface is $\alpha \doteq arccos\left(\frac{a}{\sqrt{(3\sigma)^2+(a)^2}}\right)$. The backscattering coefficient $\eta(\alpha)$ is the function of this effective incident angle α, which can be calculated by IM3D code directly. Thus, for the zero-order approximation, the primary retention rate of ions is equal to,

$$R_0(\alpha) \doteq 1 - \eta(\alpha), \quad all\ 3\sigma. \tag{6.4}$$

Here, we assumed that it is actually true when $3\sigma \leq z_0$, where $z_0 = 50\ nm$ is used. While for $3\sigma > z_0$, a fraction of backscattered ions would be shaded and redeposited by the roughness peaks, the shading probability can be estimated by the geometric effect as $P_s = \int_0^\alpha sin(\theta)\, d\theta = 1 - cos\alpha$. Thus, by assuming that all of the shaded ion are redeposited by the roughness peaks for the first-order approximation, the primary retention rate of ions can be described by,

$$R_1(\alpha) \doteq (1 - \eta(\alpha)) + \eta(\alpha)(1 - cos\alpha) = 1 - \eta(\alpha)\, cos\alpha, \quad 3\sigma > z_0. \tag{6.5}$$

If considering there is still some probability (R_1 is approximately used here) for the escaping of the shaded ions by roughness peaks, a more accurate estimation of the primary

retention rate of ions in rough-surface can be given by a second-order approximation base on the predictor-corrector method,

$$\begin{aligned} R_2(\alpha) &\doteq (1-\eta(\alpha)) + R_1(\alpha)\,\eta(\alpha)\,(1-cos\alpha) \\ &= 1-\eta(\alpha)\,cos\alpha - \eta^2(\alpha)\,cos\alpha + \eta^2(\alpha)\,cos^2\alpha, \quad 3\sigma > z_0. \end{aligned} \tag{6.6}$$

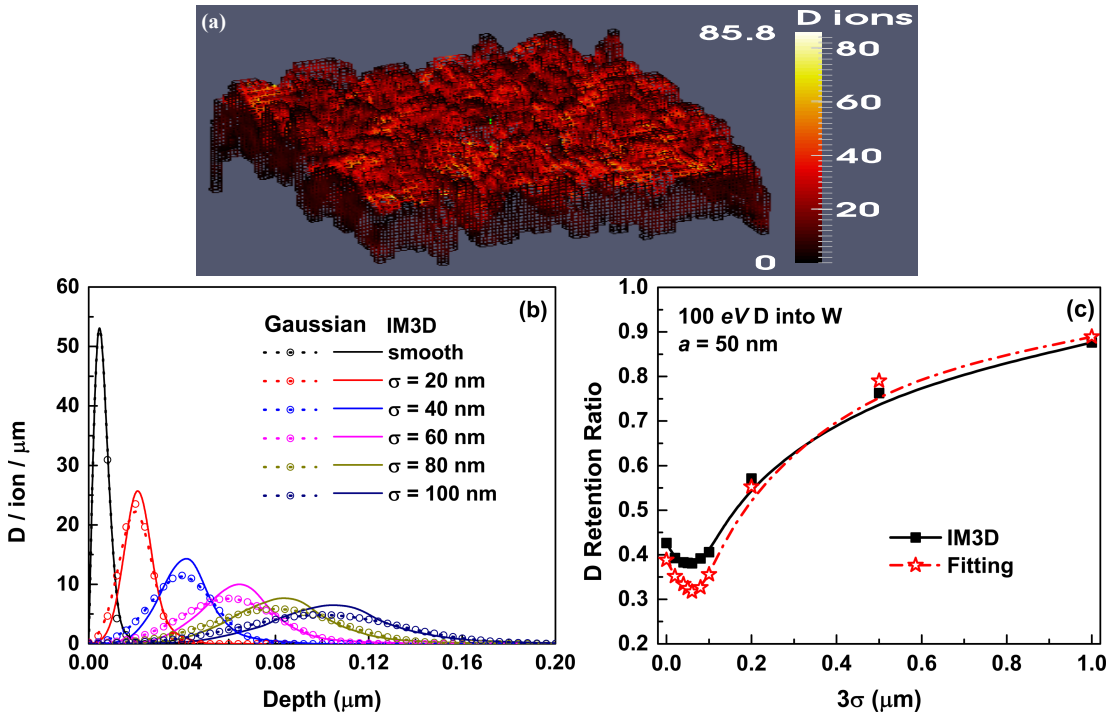

FIGURE 6.9: (a) The D ion space-distribution for W roughness surface with the roughness amplitude of $3\sigma = 60\ nm$ and the roughness constant of $a = 50\ nm$. (b) The D ion depth-distributions and the fitting profiles by Gaussian functions for W roughness surface with different roughness amplitudes (3σ) and the roughness constant of $a = 50\ nm$. (c) The D retention ratio and the fitting Eqs.(6.4) and (6.6) along with the roughness amplitudes for W roughness surface with the roughness constant of 50 nm. Here, all W roughness surfaces are under the irradiation of 100 eV D ion beam with the random normal-incidence.

A good agreement can be given based on the rough estimations by Eq.(6.4) for $3\sigma \leq z_0$ and Eq.(6.6) for $3\sigma > z_0$ (as shown in Fig.6.9(c)), which demonstrates that the geometric and shading effects are the main contributions to the enhancement of the primary D retention in W. At finite temperature or especially high temperatures (typically $400 - 800\ ^\circ C$ in ITER[33]) in fusion instruments, interstitial D atoms will diffusion quickly, most of which would desorb from surface. While, if the influence of the diffusion effect is fixed, the primary retention rate would be the only key factor to the D retention, especially when the surface of PFMs becomes roughness. Furthermore, the retention of D/T/He in "fuzz" structures should also follow the same trend at a first glance.

Chapter 7

Additional Tools

7.1 Config.in file generation code

gen_config_in

7.2 Shape files generation code

gen_shape

7.2.1 CSG

7.2.2 FETM

7.3

Chapter 8

Errors and Warnings

8.1

8.2

8.2.1

8.2.2

8.2.3

8.3

8.3.1

8.4

8.4.1

8.4.2

8.5

8.6

Appendix A

Physical models

In IM3D code, the simulation of ion transportation in matter proceeds is similar to the well-established TRIM-like codes, which basically introduces the random phase approximation (RPA), the binary collision approximation (BCA) and the central potential approximation (CPA)[1, 2, 34]. The code simulates numerical random trajectory histories of ions to present statistically meaningful calculation results. Each trajectory corresponds to a particle (ion or target atom) with a specified starting position, a given direction and an incident energy. The particle is tracked as a random sequence of straight free-flight-paths that end with a binary nuclear collision event where the particle changes its direction of movement and/or reduces energy as a result of nuclear (elastic collision process) and electronic (inelastic collision process) energy losses. The energy and direction of the particle from incident direction are thus updated from the conservation of energy and momentum. Where, the probability of energy losses depends on the target atom density, as well as the nuclear and electronic stopping powers which can be assumed to be independent. Meanwhile, point defects could be produced in elastic collision events. Finally, the trajectory is terminated till the energy of the particle drops below a specified value E_{min} or the particle leaves the target. A program chart of the ions tracing and defects generation in a target is given in Fig.A.1 and the detailed physical background can also be found elsewhere[1].

For the elastic collision process, the classical binary atomic scattering angle θ_{CM} in the center-of-mass (CM) coordinate system can be evaluate accurately from the famous "scattering integral" as,

$$\theta_{CM} = \pi - 2\int_{r_0}^{\infty} \frac{p \cdot dr}{r^2\sqrt{1 - \frac{V(r)}{E_c} - \left(\frac{p}{r}\right)^2}} \tag{A.1}$$

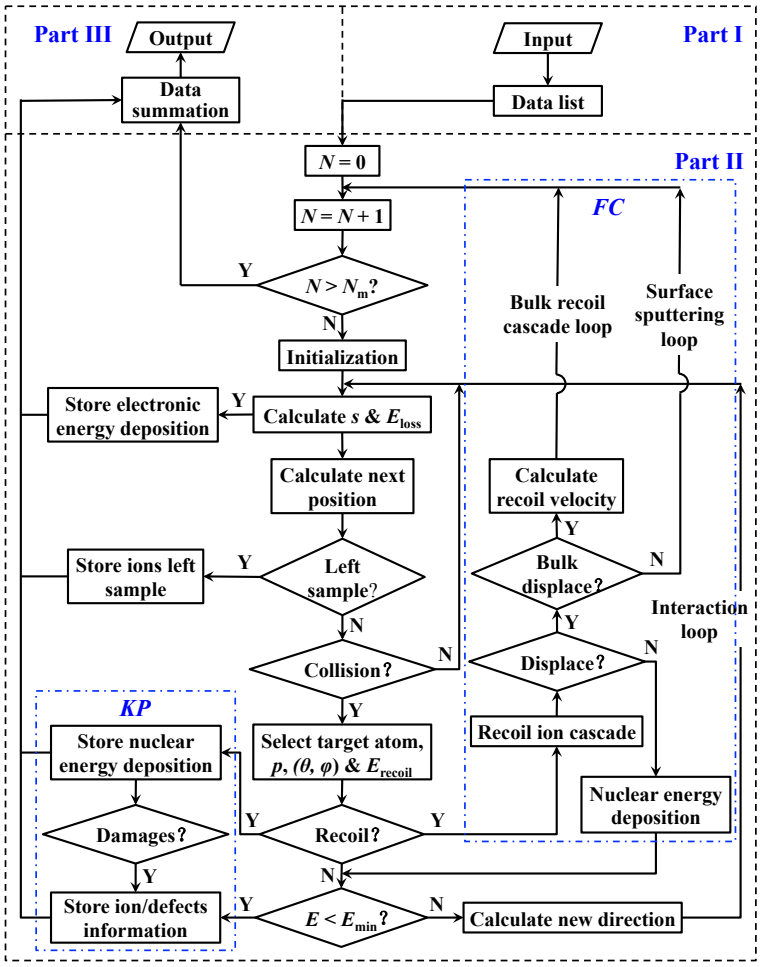

FIGURE A.1: IM3D program chart of ions tracing and defects generation in a sample.

where, p is the impact parameter, r_0 is the closet distance (r) between two atoms during the collision, $V(r)$ is the screened inter-atomic potential as listed below and E_c is the kinetic energy of the incident atom in the center-of-mass frame. This integral can not be calculated analytically for inter-atomic screening potentials and hence a time-consuming numerical integration must be used instead[1]. An analytical approximation formula (such as the MAGIC approximation[1]) or a lookup table method (such as the fast indexing technique[2]) can be used in terms of accuracy and efficiency, as described in Section 2.3. The interaction potential $V(r)$ between two atoms is a screened repulsive Coulomb potential described by a dimensionless screening function, such as the Thomas-Feimi potential[35], the Lenz-Jensen potential[36], the Moliere potential[37], the Bohr potential[38] and the universal Ziggler-Biersack-Littmark (ZBL) potential[1]. In addition, the recoil energy of a target atom due to elastic nuclear collisions can be evaluated by the BCA between two charged particles involved in one scattering process.

For the inelastic collision process, the ion energy reduce uniformly along the free-flight-paths though the electronic energy losses accounting for the energy straggling. In IM3D

code, the physical parameters such as electronic energy stopping powers are generated from SRModule.exe provided by SRIM package[39], in the form of pre-calculated tables as implemented in Corteo[2]. Either Bohr[40], Chu[41] or Wang[42] formula can be selected to considering the energy straggling. Furthermore, IM3D code also employ a linear addition of stopping powers (known as Bragg's rule[43]) for determining the stopping power in alloys, compounds and mixtures.

The generation of point defects (i.e. interstitials and vacancies) can be modeled by the analytical modified Kinchin-Pease (KP) model[11, 12] or the computationally full cascade (FC) simulation. Assume a trajectory atom with atomic number Z_1 and energy E collide with a target atom with atomic number Z_2. After the collision event, the energy of the trajectory atom change to E_1 and the target atom obtains energy E_2, and thus different damage generation processes would also occur: (1) if $E_2 > E_d$ (E_d is the displacement energy of the target atom), a displacement occurs so that the target atom has enough energy to leave the site; (2) if both $E_1 > E_d$ and $E_2 > E_d$, a vacancy occurs; (3) if $E_2 < E_d$, the target atom without enough energy will vibrate back to its site releasing energy E_2 as phonons; (4) if $E_1 < E_b$ (E_b is the lattice/surface binding energy of the target atom), $E_2 > E_d$ and $Z_1 = Z_2$, a replacement collision occurs with energy E_1 releasing as phonons; (5) if ($E_1 < E_{min}$ *or* E_b, $E_2 > E_d$ and $Z_1 \neq Z_2$) or ($E_1 < E_{min}$ *or* E_b and $E_2 < E_d$), the trajectory atom becomes an interstitial atom.

Appendix B

3D structural models

The arbitrary complex 3D nanostructured targets can be generated by graphical softwares beforehand and traced by two types of sophisticated 3D structural algorithms in IM3D code, that is, CSG and FETM methods[6, 8]. Then the different types of targets are set in the corresponding coordinate systems as shown in Fig.B.1, respectively. The ion beams with different atomic numbers and incident directions follow different types of space-distributions (i.e., specified point, center point, uniform or Gaussian random distributions and so on). The targets are considered as composing of different geometric elements each of which with different amorphous materials (i.e., elements, alloys and compounds).

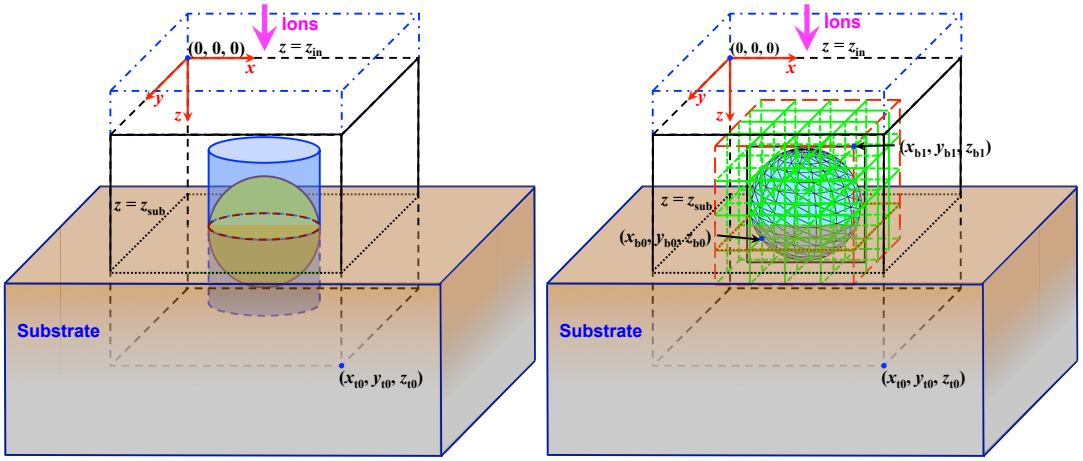

FIGURE B.1: CSG and FETM geometric models and the corresponding coordinate systems.

On the one hand, CSG method[3, 5] is simply introduced in IM3D code which uses simple geometries to build a complex structure with Boolean operations: union, difference and intersection, etc. By CSG modeling, a complex geometric structure can be constructed with some basic and simple 3D bodies as elements that can be analytically described

with a few parameters. The basic elements, such as sphere, tetrahedron, cuboid, ellipsoid, taper, column, polyhedron, paraboloid, hyperboloid and so on, can be easily implemented into subroutines to allow efficient calculation of intersecting points of an ion trajectory with a geometric surface. Detailed construction process can be found in Refs.[3, 5] . The limits of this method are mainly from two sides: firstly, it is obvious that, with the limited number of parameters, one cannot in practice build an arbitrary-complex geometric structures. Fortunately, most of the simple geometric nanostructures can be modeled by the present algorithm. Secondly, for the step of ion/atom trajectories, one has to compute the intersection points with every possible basic shape. A large number of such judging procedures in a MC simulation would cause a heavy computing cost. An efficient parallel algorithm is needed for the simulation of very complex targets constructing with this geometric model.

On the other hand, FETM method[4, 7] in computer graphics is also employed in IM3D code by using a finite element triangle mesh to construct a sample surface, and furthermore, by using the space subdivision method to accelerate the calculation. In principle, it allows an easy construction of an arbitrary-complex geometric structure with smooth or roughness surface. A closed 3D geometric structure can be constructed by just joining a series of mesh points that outline the 3D geometric structure, which can be generated easily by different algorithms with different graphical softwares (such as Gmsh software[10]) or even user's own definition. The accepted formats of FETM shape files are *opengl* and *ply*2 at present. The reason to use a triangulated mesh is due to its advantage on easier judging of intersection points of a velocity vector with a local triangular plane when considering an ion incidence into a sample surface or emission from a surface. This method is more realistic and unified for simulating much complex targets while on the premise of spending less consuming time.

In IM3D code, the trajectories are traced step by step in a complex 3D target. The free-flight-paths of an ion between two successive collision events follow the Poisson distribution with a mean free-flight-path of $l = n^{(-1/3)}$ (where n is the atomic density of the target) or a constant value specified by user. Afterwards, a ray-tracing technique[3, 5] for an inhomogeneous specimen with a complex geometric structure and the space subdivision method are introduced to accelerate the calculations[4, 7]. Furthermore, when ions transport in a specimen, three physical quantities, i.e. the free-flight-path, the direction deflection, and the kinetic energy change after refraction at surface/interface, must be treated appropriately, especially at the boundaries of complex geometric structures[4].

Bibliography

[1] J. F. Ziegler, M. D. Ziegler, and J. P. Biersack. *Nucl. Inst. Meth. Phys. Res. B*, 268:1818–1823; SRIM website, http://www.srim.org, 2010.

[2] F. Schiettekatte. *Nucl. Inst. Meth. Phys. Res. B*, 266:1880–1885, 2008.

[3] H. M. Li and Z. J. Ding. *Scanning*, 27:254–267, 2011.

[4] Y. G. Li, S. F. Mao, H. M. Li, S. M. Xiao, and Z. J. Ding. *J. Appl. Phys.*, 104: 064901, 2008.

[5] Y. G. Li, Z. J. Ding, and Z. M. Zhang. *J. Appl. Phys.*, 106:024316, 2009.

[6] Y. G. Li, S. F. Mao, and Z. J. Ding. *Applications of Monte Carlo Method in Science and Engineering: Chapter 11. Monte Carlo Simulation of SEM and SAM Images.* Eds. S. Mark and S. Mordechai, InTech-Open Access Publisher, 2011.

[7] P. Zhang, H. Y. Wang, Y. G. Li, S. F. Mao, and Z. J. Ding. *Scanning*, 33:1–6, 2011.

[8] Y. G. Li, P. Zhang, and Z. J. Ding. *Scanning*, 35:127–139, 2013.

[9] C. Borschel and C. Ronning. *Nucl. Inst. Meth. Phys. Res. B*, 269:2133–2138, 2011.

[10] C. Geuzaine and J.-F. Remacle. *Inter. J. Nume. Meth. Eng.*, 79:1309–1331; http://geuz.org/gmsh/., 2009.

[11] G. H. Kinchin and R. S. Pease. *Rep. Prog. Phys.*, 18:1–51, 1955.

[12] M. J. Norgett, M. T. Robinson, and I. M. Torrens. *Nucl. Eng. Des.*, 33:50–54, 1975.

[13] J. Li. *Modelling Simul. Mater. Sci. Eng.*, 11:173–177; http://li.mit.edu/Archive/Graphics/A/., 2003.

[14] R. E. Stoller, M. B. Toloczko, G. S. Was, A. G. Certain, S. Dwaraknath, and F. A. Garner. *Nucl. Inst. Meth. Phys. Res. B*, 310:75–80, 2013.

[15] Y. Q. Chang, Y. W. Zhang, Z. H. Zhu, P. D. Edmondson, and W. J. Weber. *Nucl. Inst. Meth. Phys. Res. B*, 286:173–179, 2012.

[16] Y. W. Zhang, I. T. Bae, K. Sun, C. M. Wang, M. Ishimaru, Z. H Zhu, W. L. Jiang, and W. J. Weber. *J. Appl. Phy.*, 105:104901, 2009.

[17] H. Paul. *AIP Conf. Proc.*, 1525:309–313, 2013.

[18] N. Li, M. Nastasi, and A. Misra. *Inter. J. Plastic.*, 32-33:1–16, 2012.

[19] S. C. Vanithakumari and K. K. Nanda. *Phys. Lett. A*, 372:6930–6934, 2008.

[20] G. Ouyang, X. L. Li, X. Tan, and G. W. Yang. *Nanotech.*, 19:045709, 2008.

[21] W. Ren, A. Kuronen, and K. Nordlund. *Phys. Rev. B*, 86:104114, 2012.

[22] R. Liontas, X. W. Gu, E. G. Fu, Y. Q. Wang, N. Li, N. Mara, and J. R. Greer. *Nano Lett*, 14:5176–5183, 2014.

[23] P. Landau, Q. Guo, K. Hattar, and J. R. Greer. *Adv. Funct. Mater.*, 23:1281–1288, 2014.

[24] J. Chen, P. Jung, J. Henry, Y. de Carlan, T. Sauvage, F. Duval, M. F. Barthe, and W. Hoffelner. *J. Nucl. Mater.*, 437:432–437, 2013.

[25] D. Brimbal, S. Miro, V. de Castro, S. Poissonnet, P. Trocellier, Y. Serruys, and L. Beck. *J. Nucl. Mater.*, 447:179–182, 2014.

[26] L. Fave, M. A. Pouchon, M. Dobeli, M. Schulte-Borchers, and A. Kimura. *J. Nucl. Mater.*, 445:235–240, 2014.

[27] Z. J. Huang, A. Harris, S. A. Maloy, and P. Hosemann. *J. Nucl. Mater.*, 451: 162–167, 2014.

[28] T. Lazauskas, S. D. Kenny, R. Smith, G. Nagra, M. Dholakia, and M. C. Valsakumar. *J. Nucl. Mater.*, 437:317–325, 2013.

[29] M-L. Lescoat, J. Ribis, A. Gentils, O. Kaitasov, Y. de Carlan, and A. Legris. *J. Nucl. Mater.*, 428:176–182, 2012.

[30] A. Cui, J. C. Fenton, W.X. Li, Tiehan H. Shen, Z. Liu, Q. Luo, and C.Z. Gu. *Appl. Phys. Lett.*, 102:213112, 2013.

[31] P. M. Derlet, D. Nguyen-Manh, and S. L. Dudarev. *Phys. Rev. B*, 76:054107, 2007.

[32] C. S. Becquart and C. Domain. *Nucl. Instr. Methods Phys. Res. B*, 255:23–26, 2007.

[33] B. D. Wirth, K. D. Hammond, S. I. Krasheninnikov, and D. Maroudas. *J. Nucl. Mater.*, in press, 2014.

[34] J. P. Biersack and L. G. Haggmark. *Nucl. Inst. Meth.*, 174:257–269, 1980.

[35] A. Sommerfeld. *Z. Phys.*, 78:283–309, 1948.

[36] W. Lenz. *Z. F. Physik*, 77:713–722, 1932.

[37] G. Moliere. *Z. Naturforschung*, 2a:133–145, 1947.

[38] N. Bohr. *Mat. Fys. Medd. Dan. Vid. Selsk.*, 18:142–143, 1948.

[39] J. F. Ziegler. *Nucl. Inst. Meth. Phys. Res. B*, 219-220:1027–1036, 2004.

[40] N. Bohr and K. Dan. *Vid. Selsk. Mat.-Fys. Medd.*, 18:8, 1948.

[41] W. K. Chu. *Phys. Rev. A*, 13:2057, 1976.

[42] Q. Yang, D. J. O'Connor, and Z. Wang. *Nucl. Inst. Meth. Phys. Res. B*, 62:149, 1991.

[43] W. H. Bragg and R. Kleeman. *Phil. Mag.*, 10:318, 1905.