User Manual for Poisson-Monte Carlo Ray-tracing Algorithm

Pankaj Priyadarshi and Neophytos Neophytou
School of Engineering, University of Warwick

November 24, 2024

Contents

1. Introduction

This manual provides detailed information about the Monte Carlo (MC) ray-tracing algorithm coupled self-consistently with 2D Poisson solver's, including installation, user instructions, and post-processing. The purpose of this software is to solve the Boltzmann transport equation (BTE) using MC approach for electron transport in real space considering nanostructured features. The simulator computes the thermoelectric transport coefficients. It considers charge carrier scattering with phonons, ionized dopants, and boundary scatterings [**?**].

It takes as input conduction band minima from the electronic bandstructure and certain scattering parameters for the transport. It also takes the few simulation input parameters as input- energy grid, Fermi energy grid, number of electron to be simulated. The complete simulation flow and inputs are outlined in the following boxes and depicted in the simulation flowchart below:

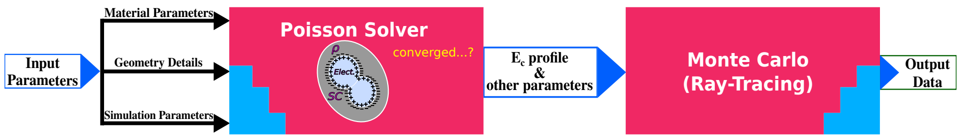


Figure 1.1: Simulation flow engine: Illustrates the input, output and flow sequence of the simulation process.

2. Installation

2.1 System Requirements

- Operating system: Windows, macOS, Linux
- RAM: at least 8GB
- Disk space: 500MB

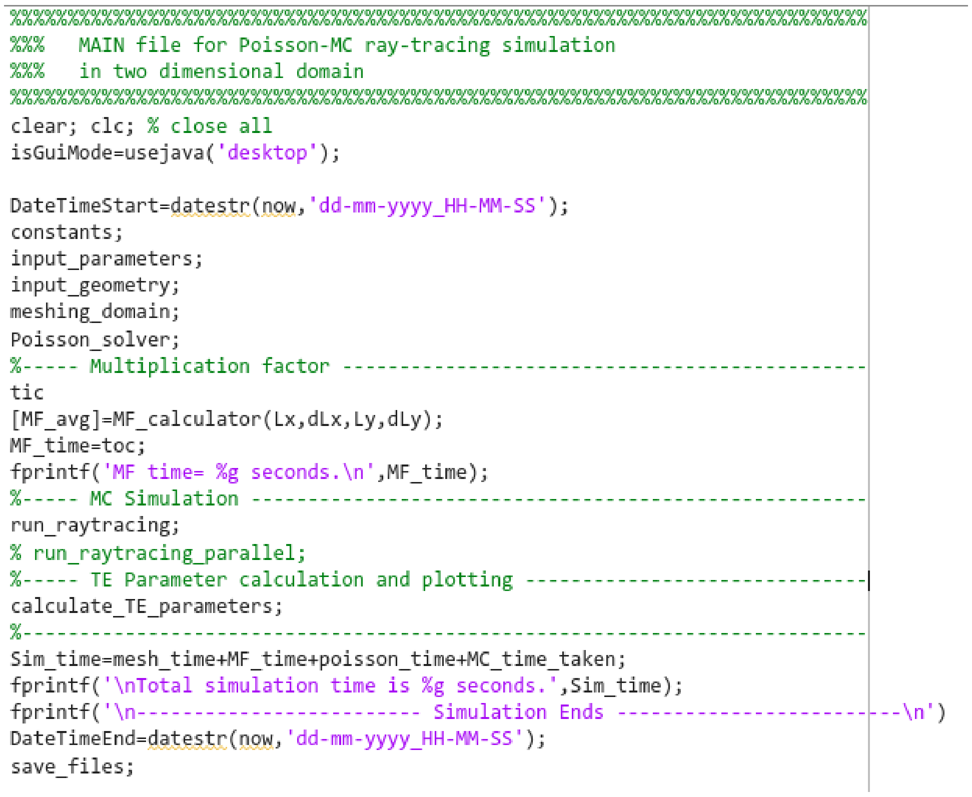
2.2 Installation Steps

1. Download the code and unzip.
2. Open MATLAB (if already installed on your machine)
3. The code is executed by running the main file, ‘main.m’ by clicking on ‘Run’ from the main MATLAB window.

3. Command file details and inputs

3.1 Structure of *main.m*

It contains a comprehensive list of all the individual files and functions required to execute the Poisson and MC simulations, as well as details for post-processing calculations. The files are executed sequentially or in series, with each process completing before the next begins. The progress is displayed in the MATLAB command window, along with the time taken to complete each steps. After completing the simulation and calculations, the data saves a ‘**.mat**‘ file with a unique filename in the folder of your choice. Here is a snapshot of ‘*main.m* file:



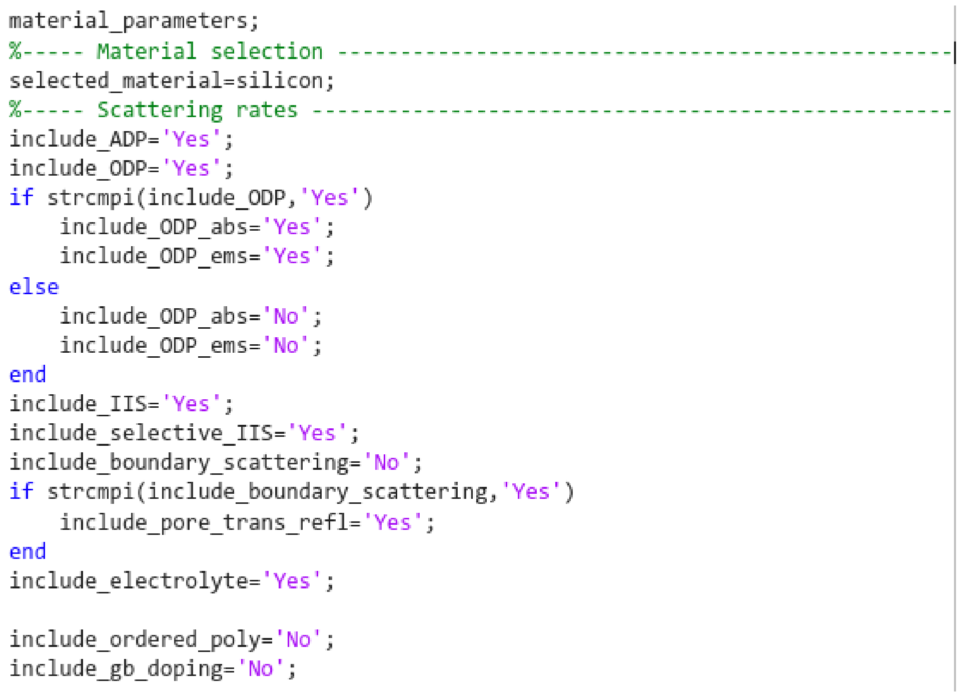
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%   MAIN file for Poisson-MC ray-tracing simulation
%%%   in two dimensional domain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; % close all
isGuiMode=usejava('desktop');

DateTimeStart=datestr(now,'dd-mm-yyyy_HH-MM-SS');
constants;
input_parameters;
input_geometry;
meshing_domain;
Poisson_solver;
%----- Multiplication factor ----------------------------------------------
tic
[MF_avg]=MF_calculator(Lx,dLx,Ly,dLy);
MF_time=toc;
fprintf('MF time= %g seconds.\n',MF_time);
%----- MC Simulation ------------------------------------------------------
run_raytracing;
% run_raytracing_parallel;
%----- TE Parameter calculation and plotting ------------------------------
calculate_TE_parameters;
%--------------------------------------------------------------------------
Sim_time=mesh_time+MF_time+poisson_time+MC_time_taken;
fprintf('\nTotal simulation time is %g seconds.',Sim_time);
fprintf('\n-------------------------- Simulation Ends -------------------------\n')
DateTimeEnd=datestr(now,'dd-mm-yyyy_HH-MM-SS');
save_files;
```

Figure 3.1: Structure of the *main.m* file for Poisson-MC simulations

In the main file, two input files are defined, which must be specified before starting the simulation. These files outline the material and simulation parameters, as well as the geometry inputs, which are detailed in the next section.

3.2 Structure of *input_parameters.m*

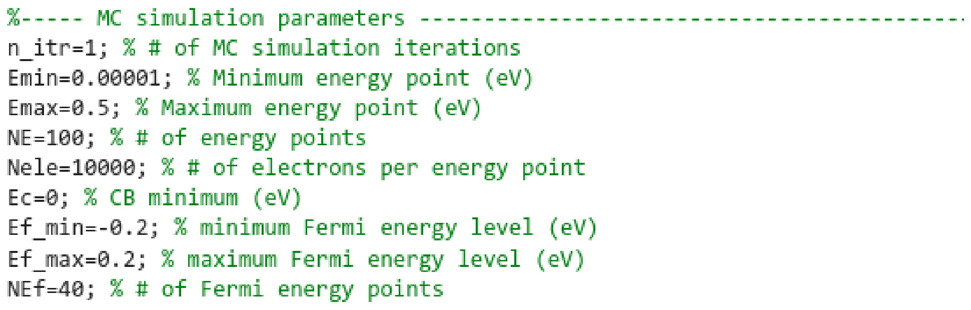
In this file, you need to specify the material of your choice (in the current code bundle, we have included material parameters for bulk silicon, Bi_2Te_3, and SnO_2). If no specific material is selected, silicon is used as the default. Additionally, you can define the scattering processes to be considered, including boundary scattering (partial transmission or reflection), particularly for nanostructured features. This package includes the details for nanoporous structures and grain boundary configurations. Below is a snapshot illustrating the selection choice:



```
material_parameters;
%----- Material selection --------------------------------------------------
selected_material=silicon;
%----- Scattering rates ----------------------------------------------------
include_ADP='Yes';
include_ODP='Yes';
if strcmpi(include_ODP,'Yes')
    include_ODP_abs='Yes';
    include_ODP_ems='Yes';
else
    include_ODP_abs='No';
    include_ODP_ems='No';
end
include_IIS='Yes';
include_selective_IIS='Yes';
include_boundary_scattering='No';
if strcmpi(include_boundary_scattering,'Yes')
    include_pore_trans_refl='Yes';
end
include_electrolyte='Yes';

include_ordered_poly='No';
include_gb_doping='No';
```

Figure 3.2: Section illustrate the *input_parameters.m* file

Afterward, you can define the MC simulation parameter settings, as shown in the snapshot below:

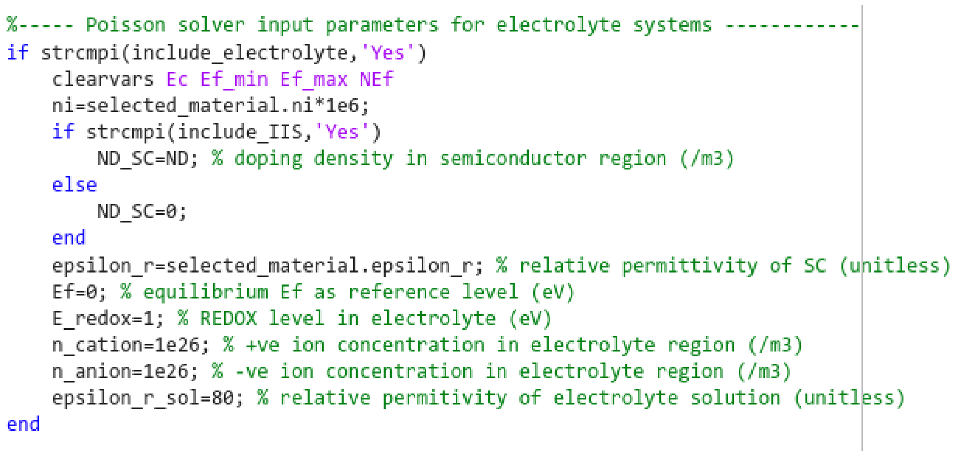


```
%----- MC simulation parameters ------------------------------------------
n_itr=1; % # of MC simulation iterations
Emin=0.00001; % Minimum energy point (eV)
Emax=0.5; % Maximum energy point (eV)
NE=100; % # of energy points
Nele=10000; % # of electrons per energy point
Ec=0; % CB minimum (eV)
Ef_min=-0.2; % minimum Fermi energy level (eV)
Ef_max=0.2; % maximum Fermi energy level (eV)
NEf=40; % # of Fermi energy points
```

Figure 3.3: Section illustrate the *input_parameters.m* file

Note:- Comments in the code are explaining the details.

In the next section of this file, you can adjust or specify the Poisson solver inputs for various structures and configurations. The specific settings for a nanoporous structure with an electrolyte configuration are shown in the snapshot of the file commands below.

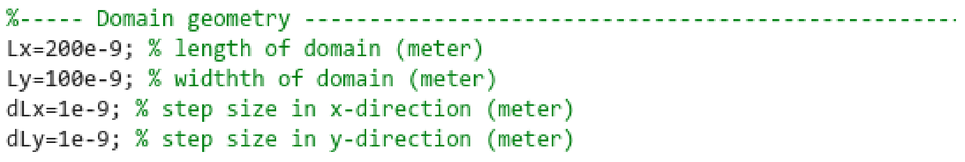


```
%----- Poisson solver input parameters for electrolyte systems -----------
if strcmpi(include_electrolyte,'Yes')
    clearvars Ec Ef_min Ef_max NEf
    ni=selected_material.ni*1e6;
    if strcmpi(include_IIS,'Yes')
        ND_SC=ND; % doping density in semiconductor region (/m3)
    else
        ND_SC=0;
    end
    epsilon_r=selected_material.epsilon_r; % relative permittivity of SC (unitless)
    Ef=0; % equilibrium Ef as reference level (eV)
    E_redox=1; % REDOX level in electrolyte (eV)
    n_cation=1e26; % +ve ion concentration in electrolyte region (/m3)
    n_anion=1e26; % -ve ion concentration in electrolyte region (/m3)
    epsilon_r_sol=80; % relative permitivity of electrolyte solution (unitless)
end
```

Figure 3.4: Section illustrate the *input_parameters.m* file

3.3 Structure of *input_geometry.m*

This file defines the domain dimensions, featuring nanostructured configurations in a two-dimensional domain. The first section outlines the overall domain dimensions and mesh grid size, as illustrated below:

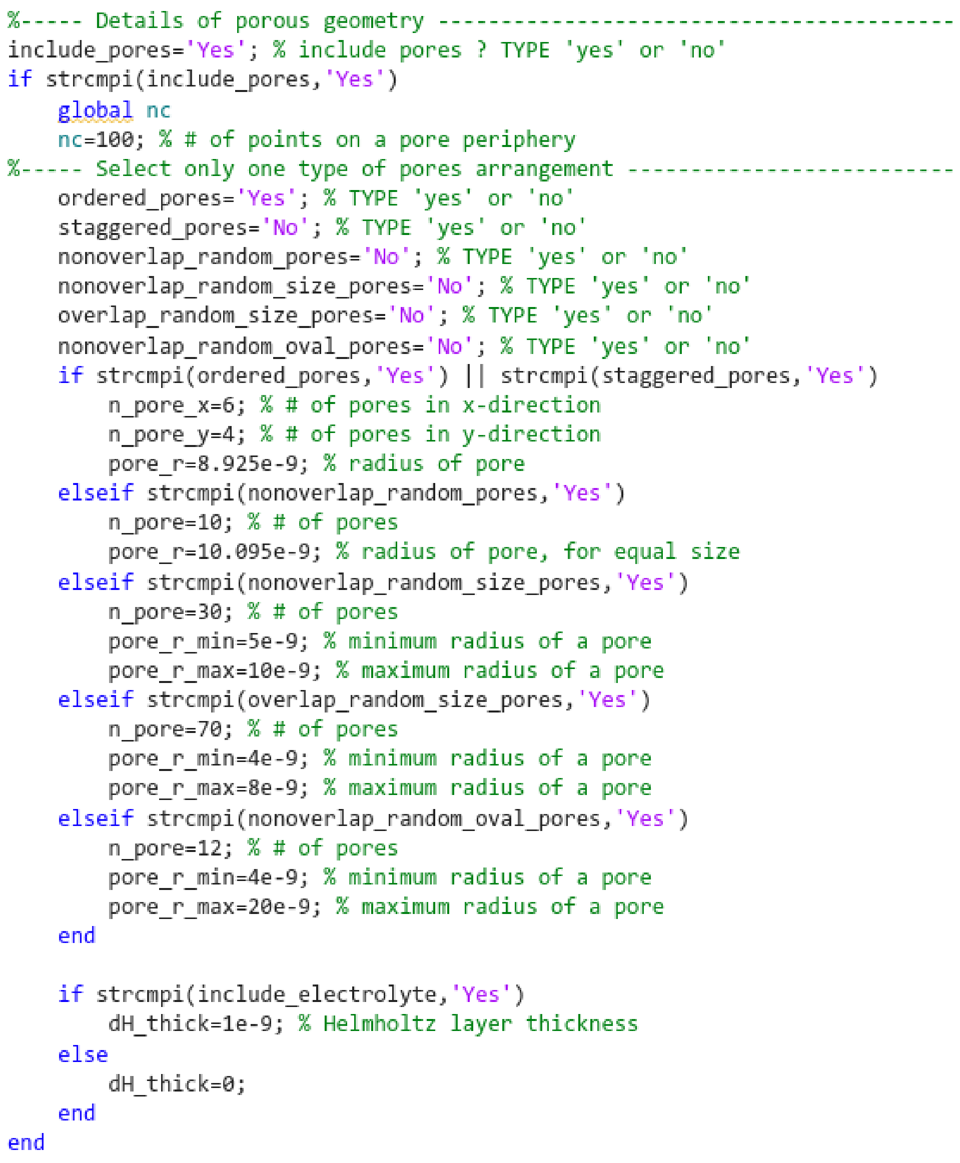


```
%----- Domain geometry --------------------------------------------------
Lx=200e-9; % length of domain (meter)
Ly=100e-9; % widthth of domain (meter)
dLx=1e-9; % step size in x-direction (meter)
dLy=1e-9; % step size in y-direction (meter)
```

Figure 3.5: Section illustrate the *input_geometry.m* file

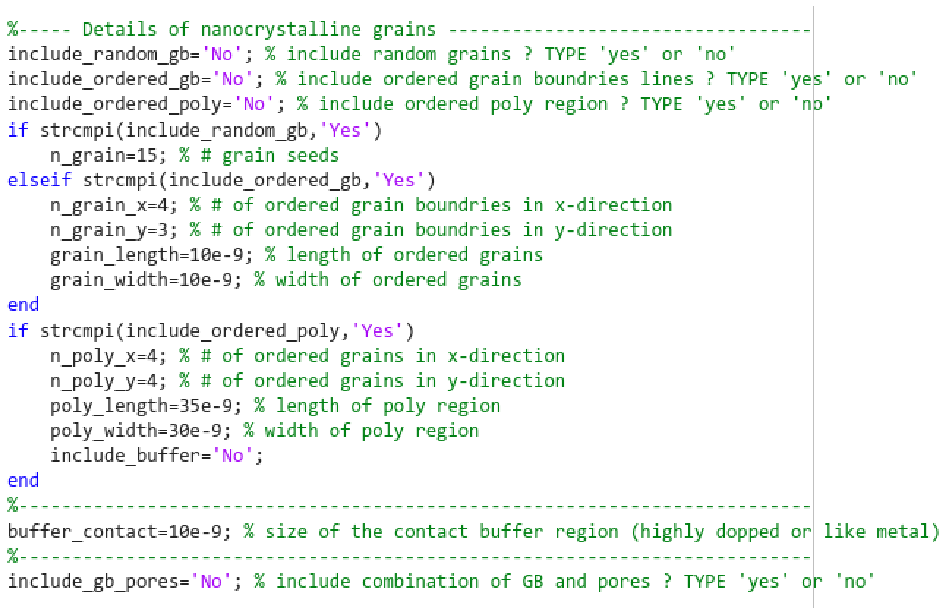
This section of the input geometry file allows for defining various porous structures, ranging from arranged to staggered, random, and overlapping configurations, as in Fig. 3.6. You can select the pore shape as either circular or oval (which has been included in this version). In future versions, we plan to support arbitrarily shaped pore structures.

The subsequent section defines the grain boundary structure, which can be configured in either an ordered or randomized manner, as clearly indicated by the comments in the code (shown in Fig. 3.7). Typical examples of a nanocrystalline geometry, a porous geometry and a porous with crystalline geometry are shown in Fig. 3.8:



```
%----- Details of porous geometry -----------------------------------------
include_pores='Yes'; % include pores ? TYPE 'yes' or 'no'
if strcmpi(include_pores,'Yes')
    global nc
    nc=100; % # of points on a pore periphery
%----- Select only one type of pores arrangement --------------------------
    ordered_pores='Yes'; % TYPE 'yes' or 'no'
    staggered_pores='No'; % TYPE 'yes' or 'no'
    nonoverlap_random_pores='No'; % TYPE 'yes' or 'no'
    nonoverlap_random_size_pores='No'; % TYPE 'yes' or 'no'
    overlap_random_size_pores='No'; % TYPE 'yes' or 'no'
    nonoverlap_random_oval_pores='No'; % TYPE 'yes' or 'no'
    if strcmpi(ordered_pores,'Yes') || strcmpi(staggered_pores,'Yes')
        n_pore_x=6; % # of pores in x-direction
        n_pore_y=4; % # of pores in y-direction
        pore_r=8.925e-9; % radius of pore
    elseif strcmpi(nonoverlap_random_pores,'Yes')
        n_pore=10; % # of pores
        pore_r=10.095e-9; % radius of pore, for equal size
    elseif strcmpi(nonoverlap_random_size_pores,'Yes')
        n_pore=30; % # of pores
        pore_r_min=5e-9; % minimum radius of a pore
        pore_r_max=10e-9; % maximum radius of a pore
    elseif strcmpi(overlap_random_size_pores,'Yes')
        n_pore=70; % # of pores
        pore_r_min=4e-9; % minimum radius of a pore
        pore_r_max=8e-9; % maximum radius of a pore
    elseif strcmpi(nonoverlap_random_oval_pores,'Yes')
        n_pore=12; % # of pores
        pore_r_min=4e-9; % minimum radius of a pore
        pore_r_max=20e-9; % maximum radius of a pore
    end

    if strcmpi(include_electrolyte,'Yes')
        dH_thick=1e-9; % Helmholtz layer thickness
    else
        dH_thick=0;
    end
end
```

Figure 3.6: Section illustrate the *input_geometry.m* file



```
%----- Details of nanocrystalline grains ----------------------------------
include_random_gb='No'; % include random grains ? TYPE 'yes' or 'no'
include_ordered_gb='No'; % include ordered grain boundries lines ? TYPE 'yes' or 'no'
include_ordered_poly='No'; % include ordered poly region ? TYPE 'yes' or 'no'
if strcmpi(include_random_gb,'Yes')
    n_grain=15; % # grain seeds
elseif strcmpi(include_ordered_gb,'Yes')
    n_grain_x=4; % # of ordered grain boundries in x-direction
    n_grain_y=3; % # of ordered grain boundries in y-direction
    grain_length=10e-9; % length of ordered grains
    grain_width=10e-9; % width of ordered grains
end
if strcmpi(include_ordered_poly,'Yes')
    n_poly_x=4; % # of ordered grains in x-direction
    n_poly_y=4; % # of ordered grains in y-direction
    poly_length=35e-9; % length of poly region
    poly_width=30e-9; % width of poly region
    include_buffer='No';
end
%-------------------------------------------------------------------------
buffer_contact=10e-9; % size of the contact buffer region (highly dopped or like metal)
%-------------------------------------------------------------------------
include_gb_pores='No'; % include combination of GB and pores ? TYPE 'yes' or 'no'
```

Figure 3.7: Section illustrate the *input_geometry.m* file

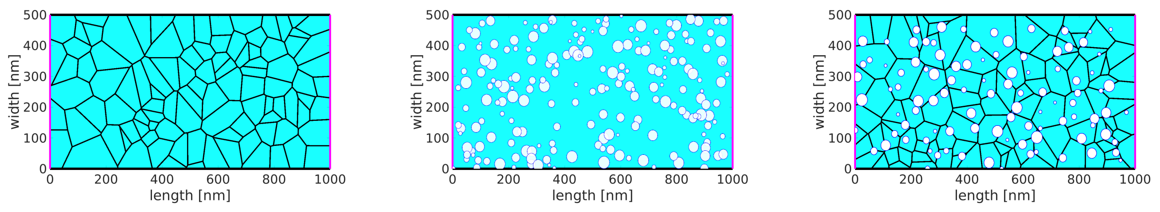


Figure 3.8: Schematic of complex nanostructured material in 2-dimensional domain populated with (a) grain boundary,(b) mesopores, and (c) combination of pores & grain boundary.

4. Post-processing

After completion of simulation, the data or matrix file is saved with a unique name in the specified folder, allowing you to use it later for post-processing and plotting to analyze the thermoelectric transport coefficients. Four files are provided for plotting: (a) the domain, (b) the potential and charge profile in the 2D domain from the Poisson solver, (c) the flux and transport distribution function (TDF), and (d) the transport coefficients such as electrical conductivity (σ), Seebeck coefficients (S), power factor, and mobility. You simply need to open the relevant file and run it according to your desired plots. Comments in the file explain which options to choose for each type of plot you need. The file names are:-

- plot_domain_meshgrid.m
- plot_Poisson_results_SC_Elect.m
- plot_flux_TDF.m
- plot_TE_parameters.m

Using the *plot_domain_meshgrid.m* file, you can generate the domain featuring nanostructures in real space, as shown in Fig. 3.8. Similarly, with the commands in the *plot_Poisson_results_SC_Elect.m* file, you can plot and analyze the Poisson potential, conduction band profile, and the resulting charge density profile within the domain. An example output for a typical input settings, considering electrolyte inside the pores, is depicted below:

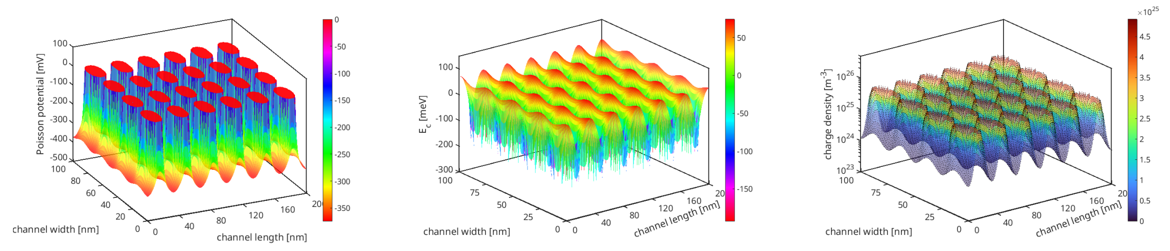


Figure 4.1: The two-dimensional profiles in the domain: (a) Poisson potential, (b) conduction band profile E_c, and (c) charge density profile obtained after the self-consistent Poisson solver.

Typical outputs are shown below (in Fig. 4.2) for the electrical conductivity σ, Seebeck coefficient S, and power factor (PF) plotted versus the charge density n. The plot includes the following cases: the pristine structure (blue solid line, representing only phonons: elastic), the pristine structure with phonons plus IIS (blue dashed-dotted line), and the solid pink line representing a hollow pore structure with the 30% porosity level.

The marked plots correspond to pores filled with electrolytes at different redox levels: $E_{\text{redox}} = 0.2$, 0.5, 1.0, 2.0, and 3.0 eV, respectively. For the redox level of $E_{\text{redox}} = 0.5$ eV, the potential profile is plotted in Fig. 4.1.

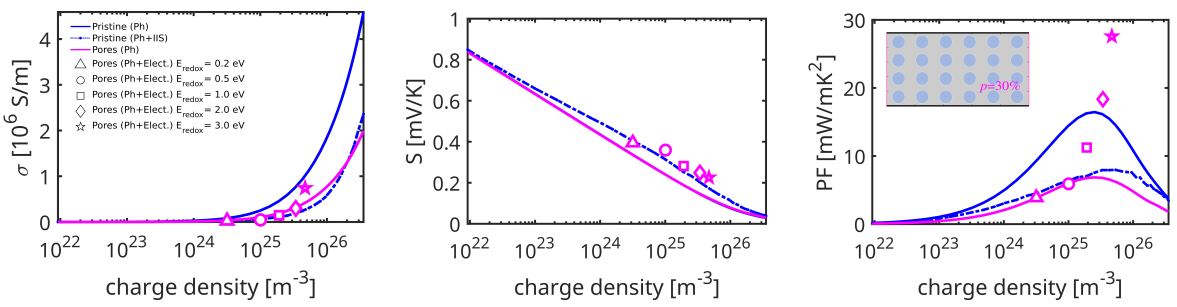


Figure 4.2: The calculated electrical conductivity (a) σ, (b) Seebeck coefficient S, and (c) thermoelectric power factor PF for a regularly arranged pore structure with 30% porosity (illustrated in the inset plot) as a function of charge density.