Instructions to run MATLAB DSGF code

Lívia M. Corrêa

February 9th, 2024

The only file to be modified is: DSGF_user_inputs.

Step 1: Write a description of your simulation. The description will be available in the results table.

```
%*******************************DESCRIPTION*******************************%

% Short description of system you are modeling (this will be used to name
% the saved files)

description = '2films_SiC_non_uniform_omega_Lx_1um_Ly_1um_t_100nm_d_100nm';
```

Step 2: Select between a ‘sample’ or a ‘user-defined’ simulation. ‘Sample’ is used for simulations with spheres, dipoles, and cubes. ‘User-defined’ is used for simulations with membranes.

Step 3: Define discretization for your simulation. Different parameters need to be modified depending on your selection in Step 2.

In the case of ‘sample’, modify `discretization`, `L_char`, `d`, and `origin`.

```
if strcmp('sample',discretization_type)

    %*********************DISCRETIZATION OF EACH OBJECT********************%
    %
    % Define the discretization for each bulk object. In the sample, each bulk object needs
    % its own discretization. The discretizations can be taken from the
    % pre-made samples or defined by the user.
    %
    % The number at the end of the chosen discretization represents the
    % number of subvolumes in that discretization.
    %
    % Pre-made sample discretization options:
    %     Discretization.sphere_*
    %     Discretization.cube_*
    %
    % Example with two sample discretizations chosen:
    %      discretization = {Discretization.sphere_8, Discretization.sphere_8};
    %

    discretization = {Discretization.sphere_1, Discretization.sphere_1};

    %**************************SCALE EACH OBJECT*************************%

    % Characteristic length for scaling the discretized lattice of each bulk
    % object.
    %
    % If a pre-made sample is chosen, the characteristic length is:
    %     sphere: radius
    %     dipole: radius
    %     cube: side length
    %
    % If a user-defined input is chosen, the characteristic length is the
    % scaling factor of the user-input cubic lattice.
    %

    L_char = [10.e-9, 10.e-9]; % [m] %[50.e-9, 50.e-9]

    %**********************DISTANCE BETWEEN OBJECTS**********************%

    % Distance between the objects
    d =100.e-9; %[m]
```

In the case of ‘user-defined’, modify the `discretization` and `delta_V` with the name of the files of the desired discretizations. These files are located in *Library/Discretizations/User_defined*.

```
elseif strcmp('user_defined',discretization_type)

    %******************DISCRETIZATION OF THE SYSTEM******************%
    %
    % Define the discretization for the system (2 group of objects).
    % The user should modify the discretization and delta_V parameters
    % according to the name of the file with the desired user-defined discretization.
    % These files are generated using matlab scripts.

    discretization = "2_films_Lx1000nm_Ly1000nm_Lz100nm_d100nm_N1600_discretization";
    delta_V = "2_films_Lx1000nm_Ly1000nm_Lz100nm_d100nm_N1600_delta_V_vector";

end
```

Step 4: Select material.

```
%********************************MATERIAL********************************%
% Options:
%     'SiO2'
%     'SiC'
%     'SiN'
%     'user_defined'

material = Material.SiC;
```

Step 5: Define dielectric function of the background reference medium.

```
%*******************DIELECTRIC FUNCTION OF BACKGROUND*******************%

% The dielectric function of the background reference medium must be purely
% real-valued.

epsilon_ref = 1;
```

Step 6: Define the frequency discretization. The option `uniform_lambda` is defined by the wavelength [m] while the options `uniform_omega` and `non_uniform_omega` are defined in angular frequency [rad/s].

```
%************************FREQUENCY DISCRETIZATION************************%

% Vector of angular frequencies at which simulations will be run.
% Vector is of dimension (N_omega x 1)

% Uncomment your selection between uniform_lambda, uniform_omega or
% non_uniform_omega.

%[omega] = uniform_lambda(5e-6, 25e-6, 100); % Wavelength [lambda] limits are provided
[omega] = uniform_omega(1.4e14, 1.9e14, 100); %Frequencies in [rad/s] limits are provided
%[omega] = non_uniform_omega(material); %Frequencies in [rad/s] limits are provided

% Suggestions of wavelength range:
%          SiO2: uniform_lambda(5e-6, 25e-6, 100);
%          SiC: uniform_lambda(9.92e-6, 13.42e-6, 200);
%          SiN: uniform_lambda(8e-6, 90e-6, 300);

% Suggestions of frequency range:
%          SiO2: uniform_omega(7.53e13, 3.76e14, 100);
%          SiC: uniform_omega(1.4e14, 1.9e14, 100);
%          SiN: uniform_omega(2e13, 3e14, 100);
```

Step 7: Define temperatures for each object. These temperatures are used for power dissipation calculations.

```
%************************TEMPERATURE OF EACH OBJECT************************%

T = [400, 300]; % [K]
```

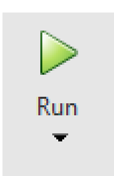
Step 8: Define temperatures for conductance calculations.

```
%*****************TEMPERATURE FOR CONDUCTANCE CALCULATIONS*****************%

% Temperature at which the spectral conductance will be calculated.

T_cond = [200,250,300,350,400]; % [K]
```

Step 9: Select outputs of the simulation.

```
%*****************************DESIRED OUTPUTS*****************************%

% Output the power dissipated in every subvolume?
output.power_dissipated_subvol = true;

% Output the power dissipated in each bulk object?
output.power_dissipated_bulk = true;

% Output the heatmap into slices?
output.heatmap_sliced = false;

% Output the total and spectral conductance for each bulk object?
output.conductance = true;

% Output the transmission coefficient matrix?
output.transmission_coefficient_matrix = false;

% Output the DSGF matrices for every frequency?
output.DSGF_matrix = false;

% Output the heat transfer coefficient?
output.heat_transfer_coefficient = true;

% Save figures?
output.save_fig = true;

% figure format
output.figure_format = FigureFormat.fig;

% Save all Workspace variables in .mat file?
output.save_workspace = true;
```

Step 10: In the MATLAB editor, press the Run button.



The results are stored in a folder named with the time the simulation was launched.