Vilnius University

Ignas Lukošiūnas

ZenScat User Guide

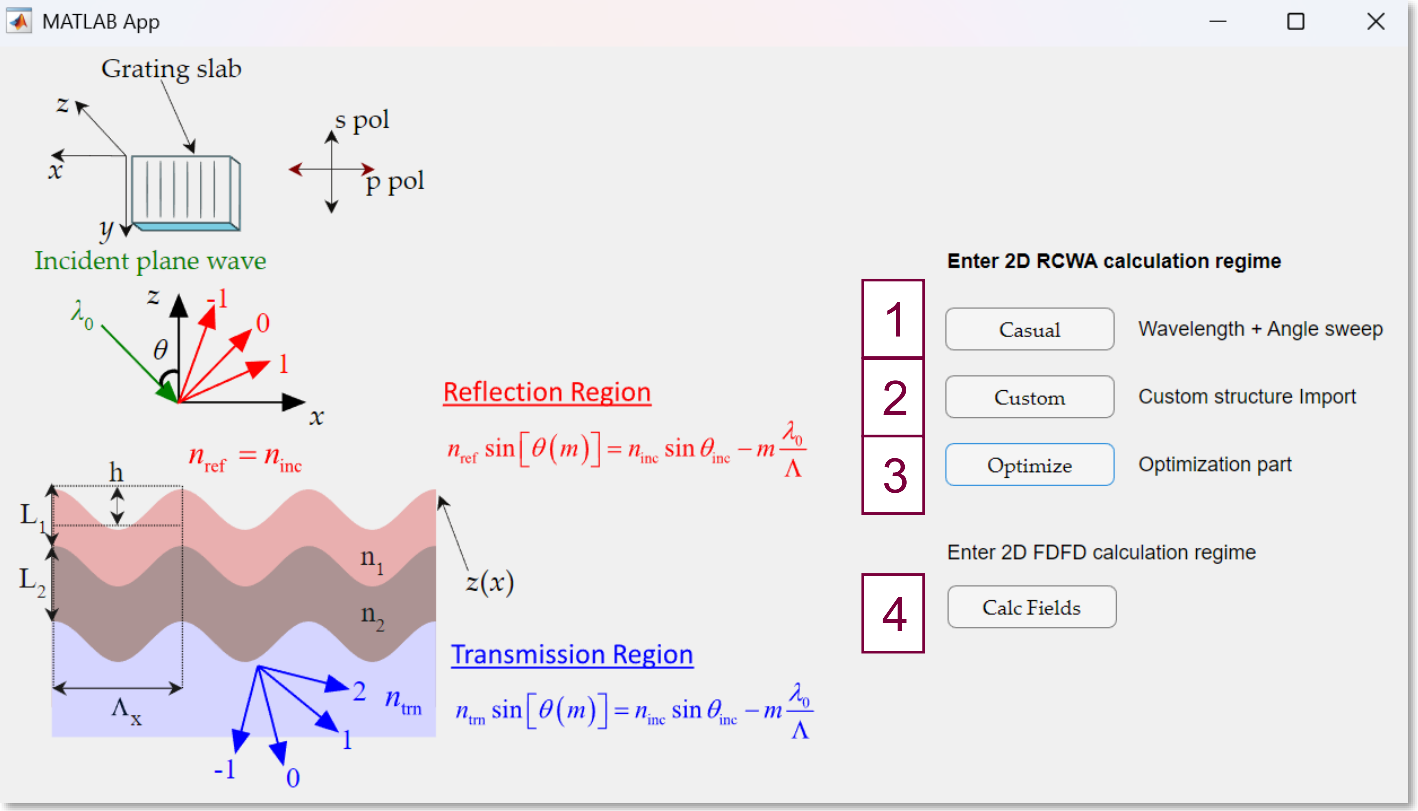
Vilnius University

Content

1. App start (RCWA)
2. Casual app (RCWA)
3. Casual import app (RCWA)
4. Optimization app (RCWA)
5. Field App (FDFD)
6. Code

App start (RCWA)

Vilnius University



1. Parameter sweep application for implemented geometries – Guided Mode Resonance Filters (GMRFs) and Photonic Crystals (PhCs).

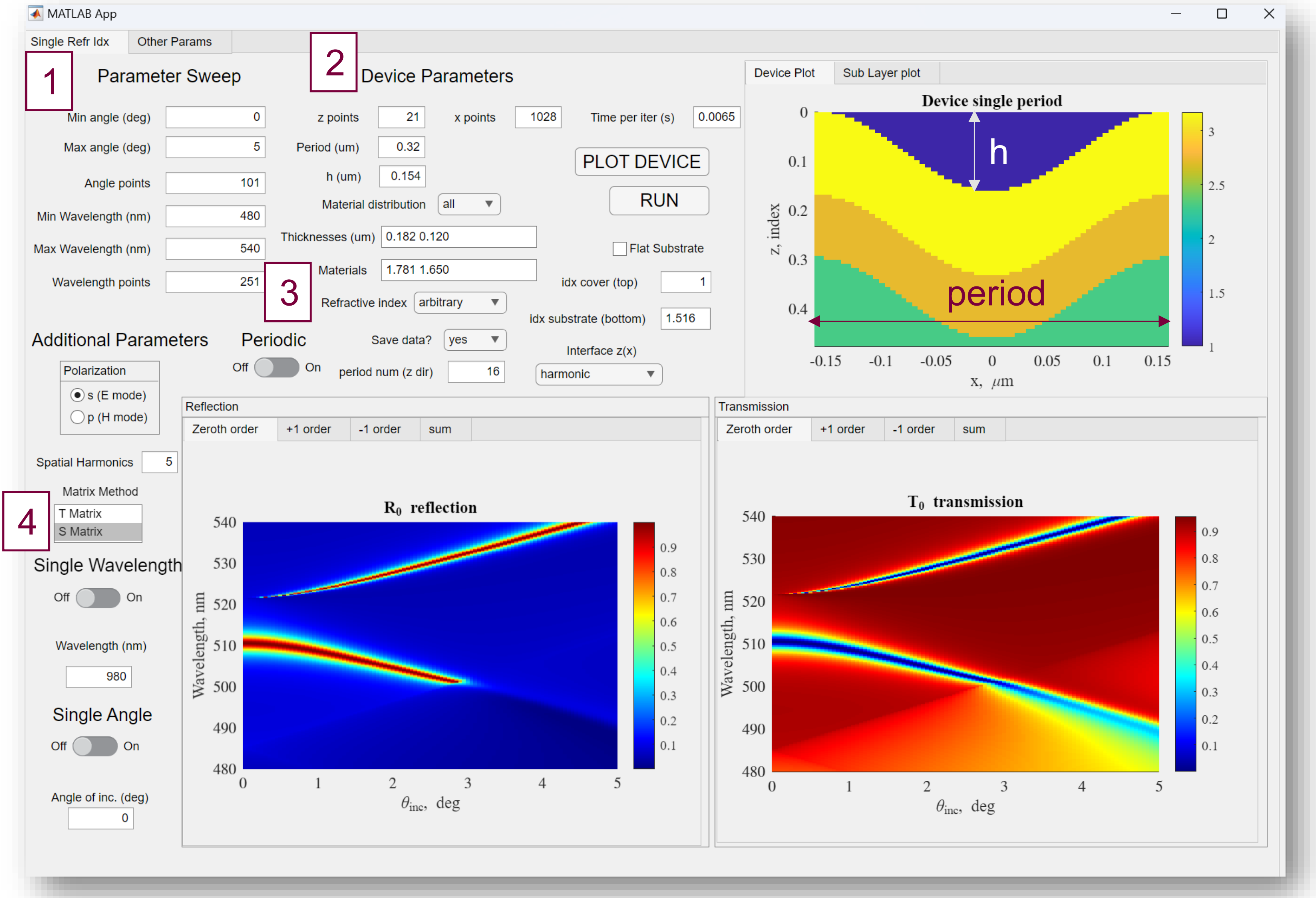
2. Parameter sweep application for custom geometries, created as external .mat files.

3. Genetic Optimization of GMRFs of diffraction efficiency and absorption/gain maximization.

4. Field calculation by 2D FDFD algorithm.

Casual app (RCWA) – GMRF example #1

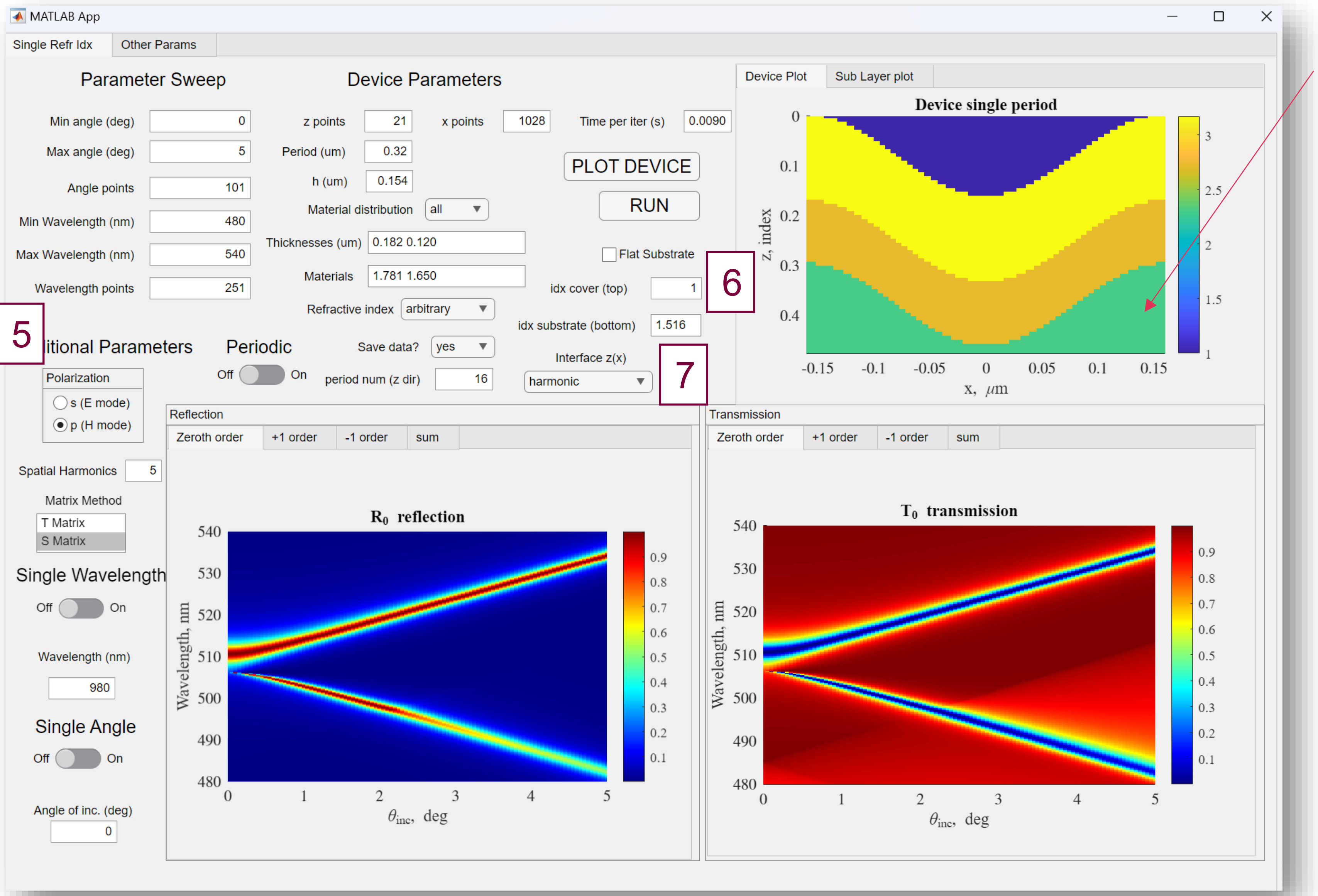
Vilnius University



1. Source parameters
2. Device parameters (period, h, z points per one h).
3. Thickness vector and material value (permittivity) vector.
4. T or S matrix methods, Single wavelength regime used of angular sweep for one wavelength value. The same thing applies for single angle as well.

Casual app (RCWA) – GMRF example #2

Vilnius University



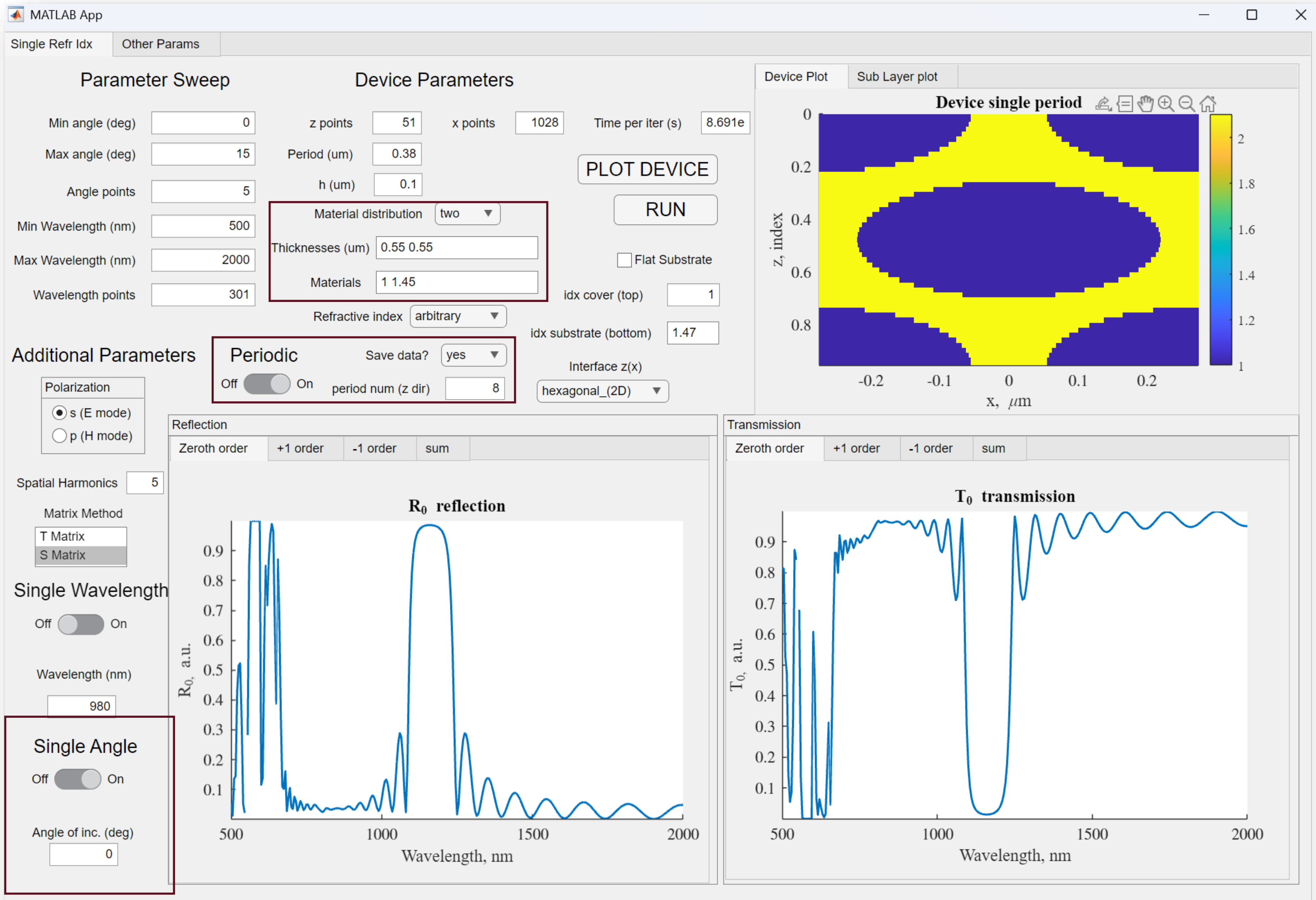
5. Same device, changed polarization from “s” to “p”.

6. Top (superstrate) and Bottom (substrate) material permittivity values.

7. Interface analytical function

Casual app (RCWA) – PhC example #1

Vilnius University



Hexagonal photonic crystal:

2 material thicknesses acquired

Material distribution is set to ‘two’

The fixed geometry’s parameters are in “Other Params” tab

Turn “PERIODIC” slide to “On”.

Single Angle “On”,

Casual import app (RCWA) #1

Vilnius University

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

#------------------- rectangle function ---------------------------------------
def rec(x):
    return abs(x) <= 0.5

#-------------- params that are saved to RCWA ------------------------------
Nx      = int(1028)
Lx      = 0.696
h       = 0.210
x       = np.linspace(-Lx/2, Lx/2, Nx)

#------------ refractive index of a GMR filter -------------------------------
n_rec = 2.52 # TiO2

#---------------------- create a device ---------------------------------------
n1 = rec(x/(0.33*Lx))
n1 = 1 - n1
n1 = n1 * (1 - n_rec) + n_rec


################# Construct device layers ####################################
ER       = np.ones((2, Nx))
ER[0,:] = n1**2
ER[1,:] = n_rec**2
sub_L    = [0.210,0.578] # sublayer thicknesses

################ Vizualize relative permittivity #############################
plt.imshow(ER, aspect = 'auto')
plt.colorbar()

#---------------------- Save data into .mat ---------------------------------

data = {    'x': x,
        'sub_L': sub_L,
           'ER': ER,
           'Lx': Lx}

scipy.io.savemat('RCWA_DATA.mat', data)
np.save('RCWA_DATA.py',data)
```

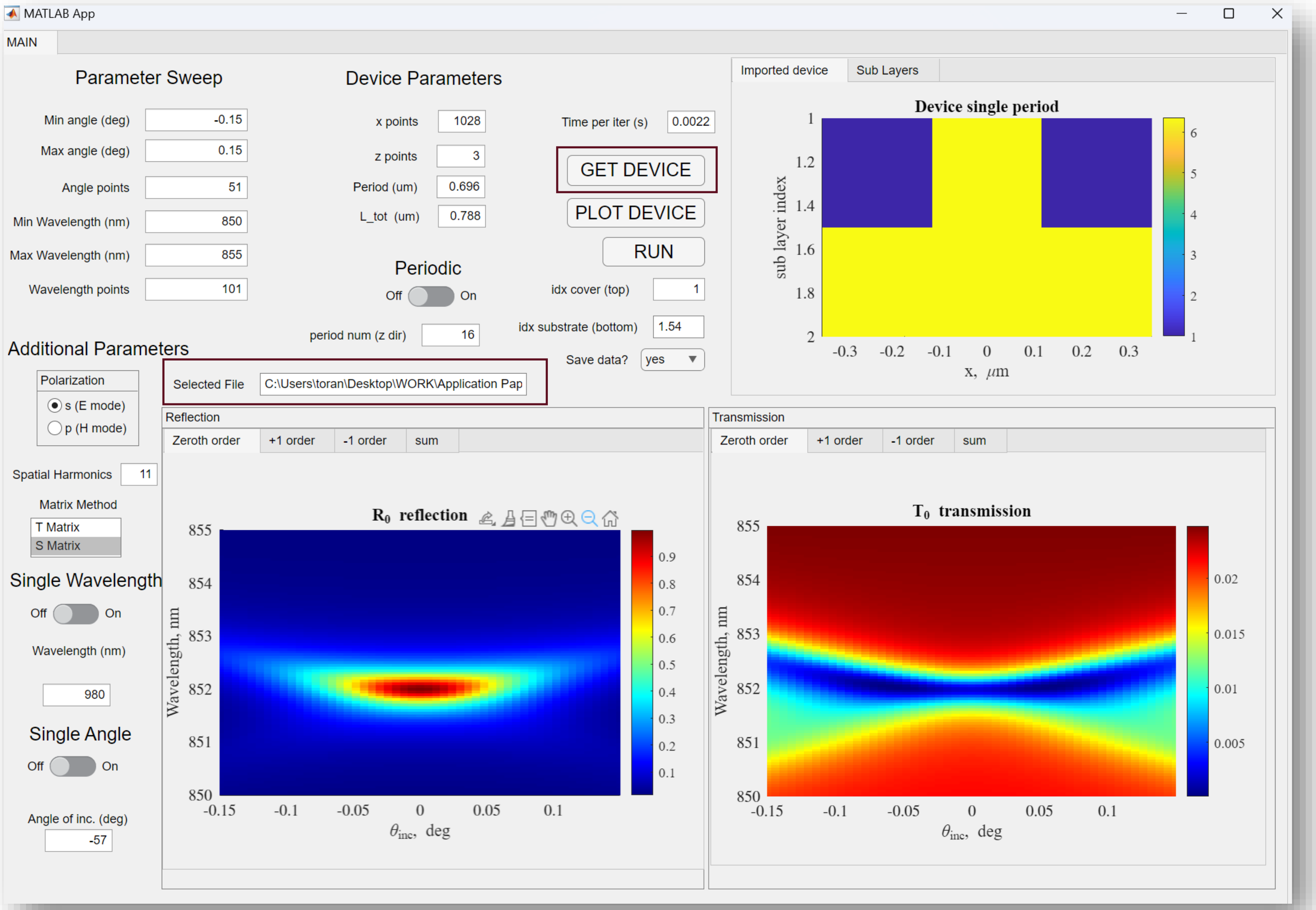
Check the .py file in folder "Spatial_Filter_Magnusson".

The output file is "RCWA_DATA.mat", which is a struct.

The application only accepts such abbreviations as written in the dict data.

Casual import app (RCWA) #2

Vilnius University



Press GET DEVICE and select the RCWA_DATA.mat file.

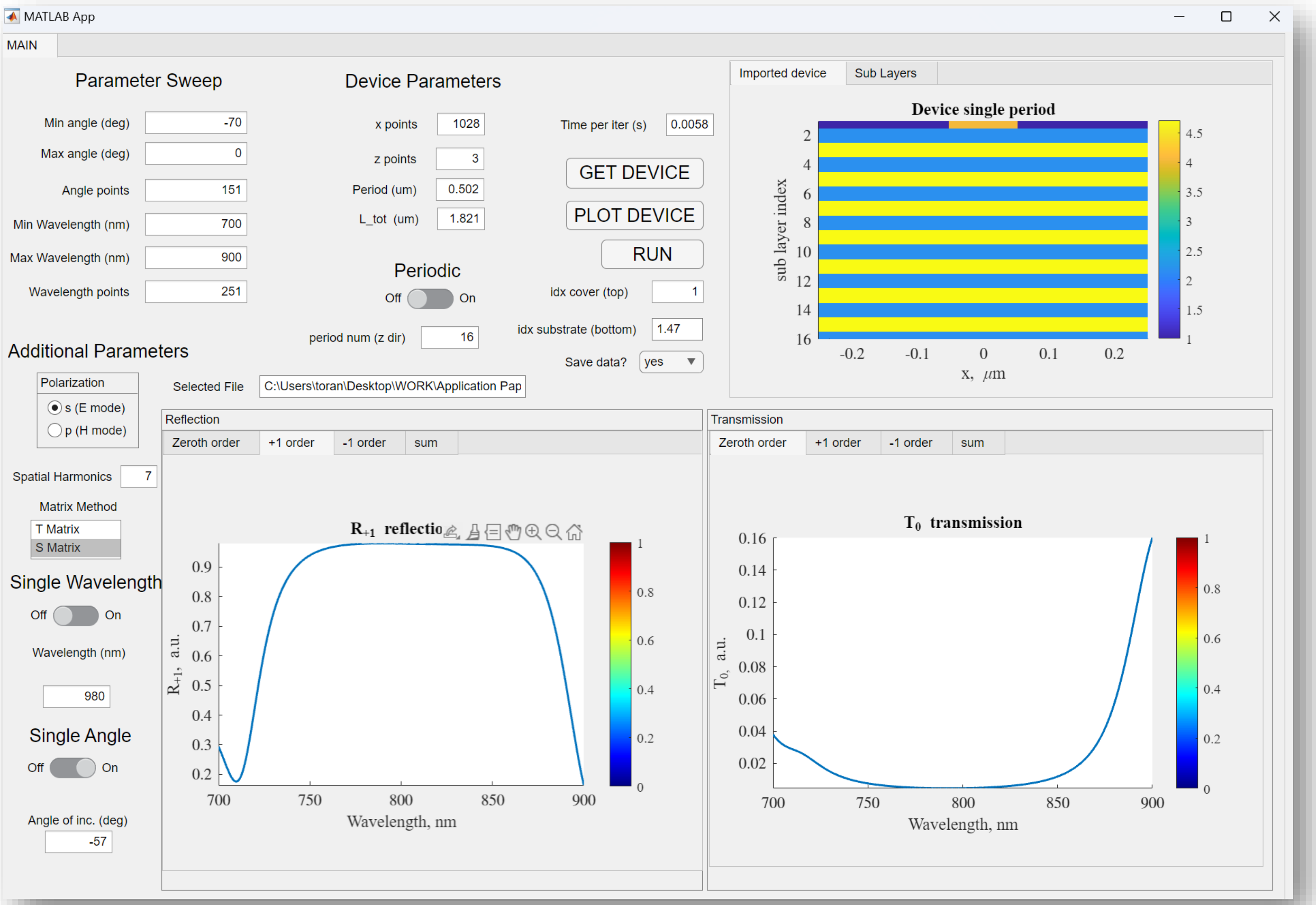
```python
import numpy as np
import matplotlib.pyplot as plt
import scipy

#------------------- rectangle function ------------------------------------
def rec(x):
    return abs(x) <= 0.5

#-------------- params list from paper ---------------------------------------
Params = [2.01,15,0.092,0.136,0.105]

#-------------- params that are saved to RCWA --------------------------------
Nx     = int(1028)
Lx     = 0.502
h      = 0.185
x      = np.linspace(-Lx/2, Lx/2, Nx)

#--------------------- params list extraction --------------------------------
n_rec     = Params[0] #HfO2
layer_num = int(Params[1])
L_n2      = Params[2]
L_n3      = Params[3]
w_rec     = Params[4]

#--------------------- relative permittivities -------------------------------
er1 = 2.17**2
er2 = 1.47**2

n1 = rec(x/w_rec)
n1 = 1 - n1
n1 = n1 * (1 - n_rec) + n_rec
n2 = er1 * np.ones(Nx)
n3 = er2 * np.ones(Nx)

########################### Construct a device ##############################
ER      = np.ones((layer_num + 1, Nx))
ER[0,:] = n1**2

sub_L = np.zeros(layer_num + 1)

#------------------------- arrange layers ----------------------------------
for i in range(1,layer_num + 1):
    if i%2 == 0:
        sub_L[i] = L_n3
        ER[i,:]  = er1
    else:
        sub_L[i] = L_n2
        ER[i,:]  = er2

sub_L[0] = h
sub_L[1] = 0.04

################ Vizualize relative permittivity ###########################
plt.imshow(ER, aspect = 'auto')

#---------------------- Save data into .mat --------------------------------
data = {    'x': x,
        'sub_L': sub_L,
           'ER': ER,
           'Lx': Lx}

scipy.io.savemat('RCWA_DATA.mat', data)
np.save('RCWA_DATA.py',data)
```


Casual import app (RCWA) #3

This import is performed for the Bragg coupler with grating on top.

See folder “LIDT_coupler”

Casual import app (RCWA) #4



Vilnius University

This import is performed for the Bragg coupler with grating on top.

See See folder "LIDT_coupler"

And import "RCWA_data.mat"

Optimization app (RCWA) #1

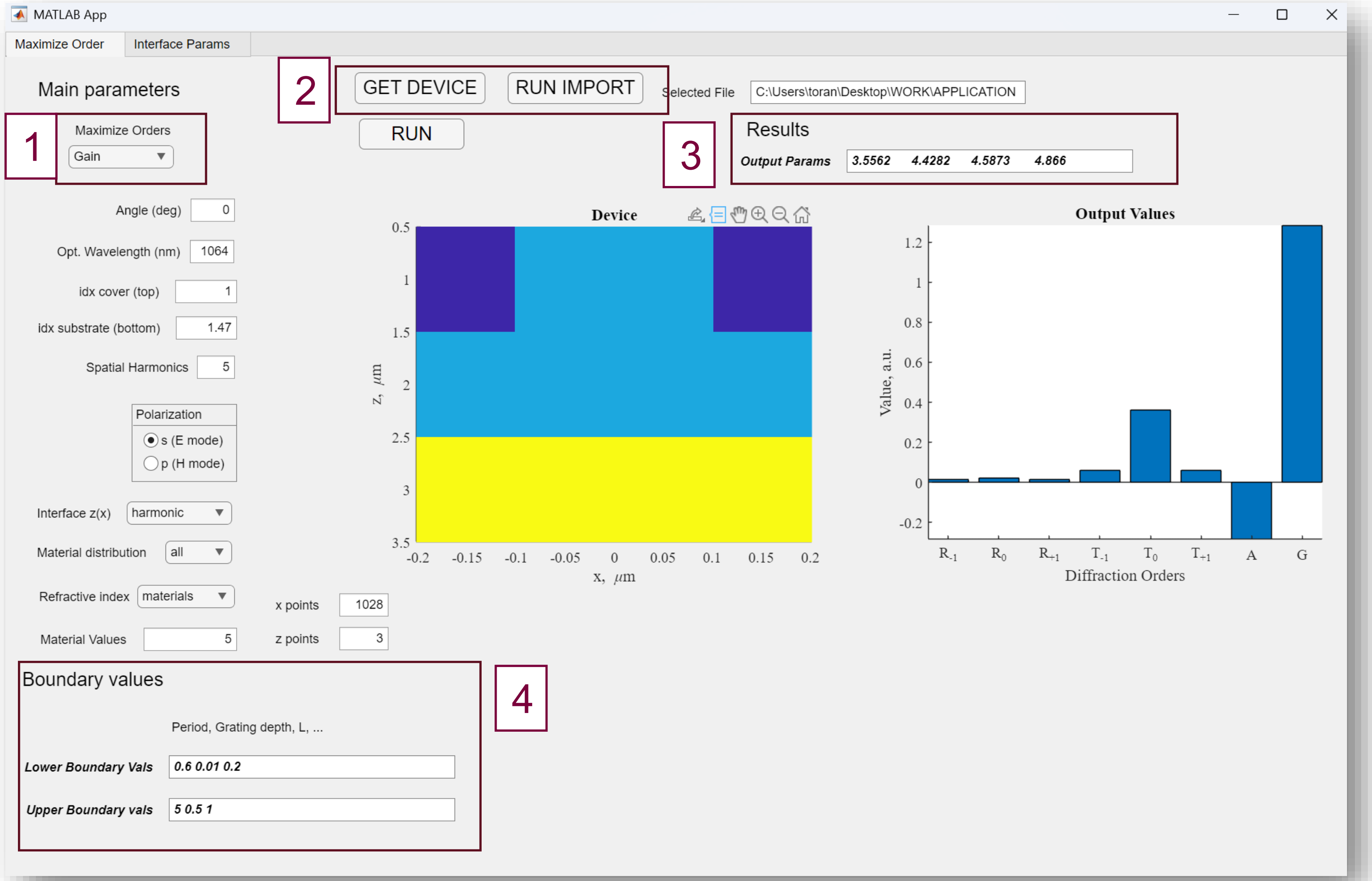
Vilnius University

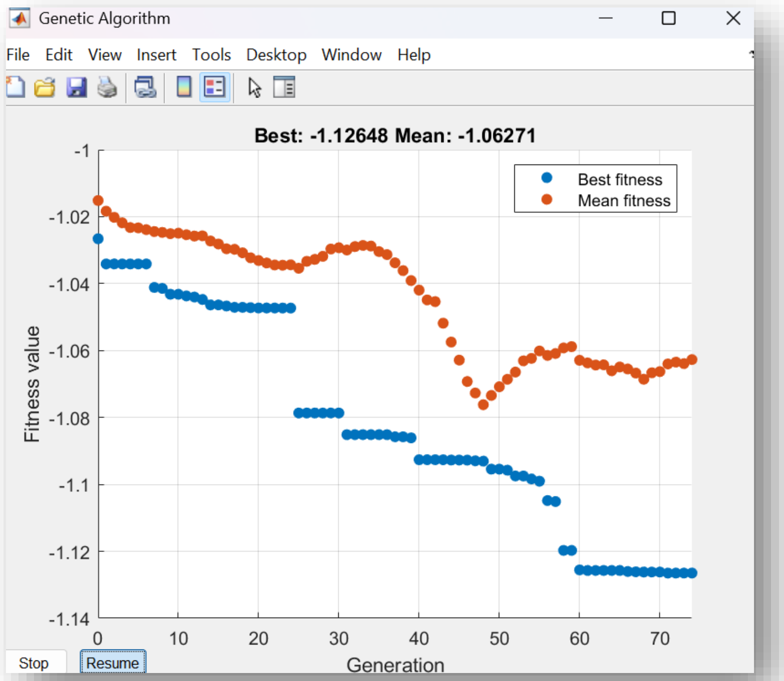
```
import numpy as np
import matplotlib.pyplot as plt
import scipy

#------------- Params list for DFB ----------------------------------------
def rec(x):
    return abs(x) <= 0.5

#------------------------------------------------------------------------

params = [1,0,0,0] # Init params

Lx     = params[0]
sub_L  = params[1:]

#------------- Params that are saved to RCWA ------------------------------
Nx     = int(1028)
# Lx     = 0.35
x      = np.linspace(-Lx/2, Lx/2, Nx)

#-------------------- Homogeneous layer count -----------------------------
layer_num = 3

#-------------------- Relative permittivities ------------------------------
# er1 = 2.00**2      # grating permitttivity (Si) or Metal
er1 = 1.5**2
# er2 = (3.4 - 1e-3j)**2  # Si permittivity
er2 = (2.04 + 1e-3j)**2 # DFB Perovskite
# er2 = -23.062 -0.39381j  # Metal's (Ag) permittivity (Frequency Domain) ε1 = -23.062

########################## Construct a device ##############################
ER    = np.ones((layer_num, Nx), dtype = 'complex')

#---------------------- create a device -----------------------------------
er_rec = rec(x/(0.5*Lx))
er_rec = 1 - er_rec
er_rec = er_rec * (1 - er1) + er1

# sub_L = np.zeros(layer_num)

#------------------------- Arrange layers --------------------------------
ER[0] = er_rec
ER[1] = er1 * np.ones((Nx))
ER[2] = er2 * np.ones((Nx))

# ############### Vizualize relative permittivity ############################
plt.close()
plt.imshow(np.real(ER), aspect = 'auto')
plt.colorbar()

#---------------------- Save data into .mat -------------------------------
data = {    'x': x,
        'sub_L': sub_L,
            'ER': ER,
            'Lx': Lx}

scipy.io.savemat('RCWA_DATA.mat', data)
np.save('RCWA_DATA.py',data)
```

Different permittivity values can be uncommented for optimization.

Period Lx here is for initialization, but it will be varied in merit function.

Optimization app (RCWA) #2





Select “GET DEVICE”

Then “RUN IMPORT”

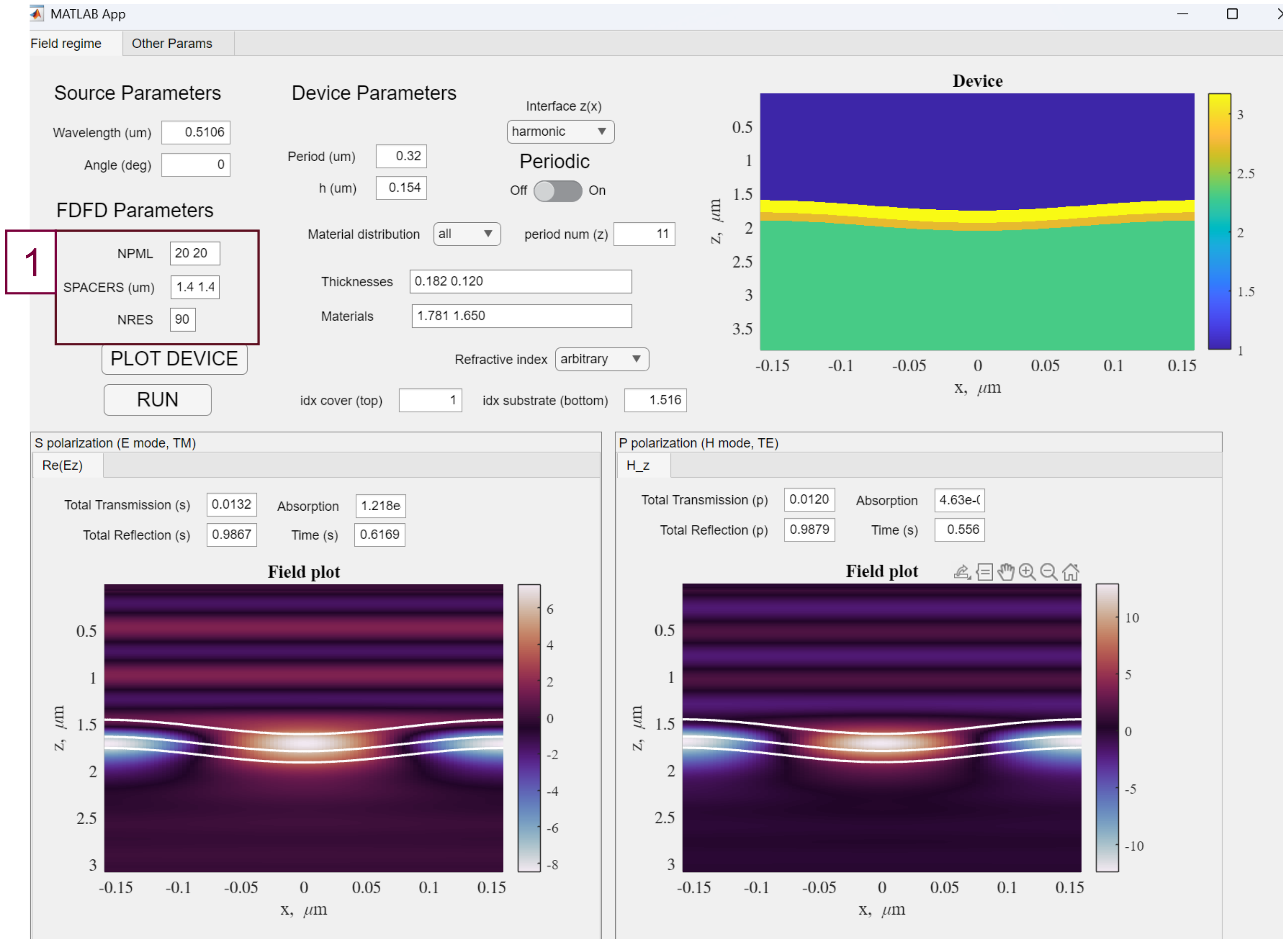
Fixed geometries are in Interface z(x)

1. Merit function = - Diffraction order.
2. Custom device import.
3. Output geometrical parameters (after Genetic Algorithm).

4. Boundary values for optimization

Field App (FDFD)

Vilnius University



1. NPML(top,bottom) – index count for Perfectly Matched Layer
2. SPACERS (expressed in micrometers) – top and bottom.
3. NRES -> points per minimal wavelength.
4. This version is applicable only for fixed geometries.

Code (Casual regime)

Vilnius University

```
%% ---------------------------------------------------------------
if strcmp(interface,'PhC_hex_columns') | strcmp(interface,'PhC_rec_circ') | ....
    strcmp(interface, 'PhC_rec_square') | strcmp(interface,'PhC_hex')

    device = Device_3(NH, grid, interface, Work, interface_stuff);
else
    if app.FlatSubstrateCheckBox.Value
        is_flat = 1;
        device = Device(NH, grid, interface, Work, interface_stuff, is_flat);
    else
        device = Device(NH, grid, interface, Work, interface_stuff);
    end
end
%-----------------------------------------------------------------
```

"Device3" used for periodic in z direction (2D PhCs)

Otherwise, "Device" is used.

```
%% Calculate diffraction efficiencies
tic
if strcmp(interface,'PhC_hex_columns') | strcmp(interface,'PhC_rec_circ') | ....
    strcmp(interface, 'PhC_rec_square') | strcmp(interface,'PhC_hex')
    [TRN,REF] = Launch_RCWA_S_PhC(P.period_num, NH,grid,device,Mode, false, app);
else
    d = uiprogressdlg(app.UIFigure, 'Title', 'Simulation in Progress', ...
        'Message', 'Processing...');
    if strcmp(app.MatrixMethodListBox.Value,'S Matrix')
        [TRN,REF] = Launch_RCWA_S_mex(NH, grid, device, Mode, false); % add _mex but compile it first
    else
        [TRN,REF] = Launch_RCWA_T_mex(NH, grid, device, Mode, false);
    end
end
b     = toc;
b     = b / length(grid.Theta) / length(grid.Lam0);
%-----------------------------------------------------------------
app.TimeperitersEditField.Value = b; % TIME OUTPUT
%-----------------------------------------------------------------
Theta = grid.Theta * 180/pi; Lam0 = grid.Lam0 * 1e9; Size = 15;
%-----------------------------------------------------------------
```

mex is used but it is compiled in local environment. Compiled application uses it globally.

It should be relaunched when running Casual application Class.

Code (Optimization) #1

Vilnius University

```
function Fitness = Merit_Function3(Work, P, NH,X, Params, Mode, ...
    interface, Obj , DispersionCoeffs, interface_stuff, refractive_idx)
%% Getting this fitness on bois!
    warning('off')
    P.Lx      = X(1);
    P.h       = X(2);
    Params(1) = X(3);
%% ----------------------------------------------------------------------
    grid      = Grid(Params, NH, interface, DispersionCoeffs, P, Work, refractive_idx);
    device    = Device(NH, grid, interface, Work, interface_stuff);
    [TRN, REF] = Launch_RCWA_S(NH, grid, device, Mode, false);
    if strcmp(Obj, 'R(-1)')
        Calc = REF.minus_1;
    elseif strcmp(Obj, 'R(0)')
        Calc = REF.REF0;
    elseif strcmp(Obj, 'R(+1)')
        Calc = REF.plus_1;
    elseif strcmp(Obj, 'T(-1)')
        Calc = TRN.minus_1;
    elseif strcmp(Obj, 'T(0)')
        Calc = TRN.TRN0;
    elseif strcmp(Obj, 'T(+1)')
        Calc = TRN.plus_1;
    elseif strcmp(Obj, 'Absorption')
        Calc = -1 * (TRN.sum + REF.sum);
    elseif strcmp(Obj, '')
        Calc = TRN.sum + REF.sum;
    end
    Fitness = -sum(Calc);
end
```

The steps are done in such a manner:

1. Params struct
2. Grid struct (recalculated params).
3. Device (ER (permittivity), ERC (Toeplitz matrix), sub_L)

Code (Optimization) #2

Vilnius University

```
function Fitness = Merit_Function_Import(NH,X, Mode, Obj, grid, device)
%% Use grid and device structcs' params for tuning
% warning('off')
    grid.Lx          = X(1);
    device.sub_L(1) = X(2)* 1e-6;
    device.sub_L(2) = X(3)* 1e-6;
    device.sub_L(3) = X(4)* 1e-6;
%% ----------------------------------------------------------------------
    [TRN, REF] = Launch_RCWA_S(NH, grid, device, Mode, false);
    if strcmp(Obj, 'R(-1)')
        Calc = REF.minus_1;
    elseif strcmp(Obj, 'R(0)')
        Calc = REF.REF0;
    elseif strcmp(Obj, 'R(+1)')
        Calc = REF.plus_1;
    elseif strcmp(Obj, 'T(-1)')
        Calc = TRN.minus_1;
    elseif strcmp(Obj, 'T(0)')
        Calc = TRN.TRN0;
    elseif strcmp(Obj, 'T(+1)')
        Calc = TRN.plus_1;
    elseif strcmp(Obj, 'Absorption')
        Calc = (1 - (TRN.sum + REF.sum));
    elseif strcmp(Obj, 'Gain')
        Calc = TRN.sum + REF.sum;
    end
    Fitness = -sum(abs(Calc));
end
```

1. Calculations are performed for micrometers.
2. Less code needed for imported structures.

Code (Optimization) #3

Vilnius University

```
function [vec_var, vec_var_output, fval_output, eflag_output] = Genetic(fun, vec_var, lb, ub)
    %% Read params
    Params = load("Optimization_Params.mat");
    Params = Params.Value;
    b      = Params(1); % summation of all layer lengths!
    Time   = Params(2); % time hours seconds
    Gen    = 4*Params(3); % generation number
    %% Genetic Algorithm used for discrete variable optimization
    L = length(vec_var);
    A = ones(1,L); A(floor(L/2)+1:end) = 0;

    Aeq = []; beq = []; nonlcon = [];
    options = optimoptions('ga','PlotFcn','gaplotbestf');
    options = optimoptions(options,'UseParallel',true);
    options = optimoptions(options,'Display','diagnose');
    options = optimoptions(options, 'Display','iter');
    options = optimoptions(options, 'MaxTime', Time*3600);
    options = optimoptions(options, 'MaxGenerations', Gen);
    options = optimoptions(options, 'PopulationSize', 150);
    rng default
    tic

    [vec_var_output, fval_output, eflag_output, ~] = ga(fun,L,A,b,Aeq,beq,...
        lb,ub,nonlcon,[],options);
    sprintf('Time passed (hours): ')
    t = toc/3600;
    disp(t)
    fprintf('Fitness function has reached the value: \n')
    disp(num2str(fval_output))
    fprintf('Algorithm outputs the given Length parameters: \n')
    disp(num2str(vec_var_output))
end
```

The genetic algorithm is a “Black Box” from Matlab.

Optimization options are single handedly selected in “optimoptions”.

Then, “ga” is run.

Code (FDFD)

Vilnius University

```
%% --------------------------------------------------------------------
[G]               = Grid_FDFD(Params,P, DispersionCoeffs, refractive_idx);
device            = Device_FDFD(P.Params, P,G, interface, interface_stuff);
%% Plot options
Size = 15;
for Mode = ['E','H']
    tic
    [TRN,REF,f] = FDFD_2D(G, device, Mode);
    b = toc;
```

Similar principles are applied for FDFD code.

Disclaimer

Vilnius University

2D FDFD is adapted from R. C. Rumpf, *Electromagnetic and Photonic Simulation for the Beginner: Finite Difference Frequency-Domain in MATLAB*, Artech House (2022).

Transmittance Matrix Method is adapted: M. G. Moharam, Drew A. Pommet, Eric B. Grann, and T. K. Gaylord, "Stable implementation of the rigorous coupled-wave analysis for surface-relief gratings: enhanced transmittance matrix approach," J. Opt. Soc. Am. A **12**, 1077-1086 (1995).

Scattering Matrix Method is adapted: Rumpf, R. C. Improved formulation of scattering matrices for semi-analytical methods that is consistent with convention. Progress In Electromagnetics Research B, **35**, 241-261 (2011).

https://github.com/ChrisFadden/RCWA