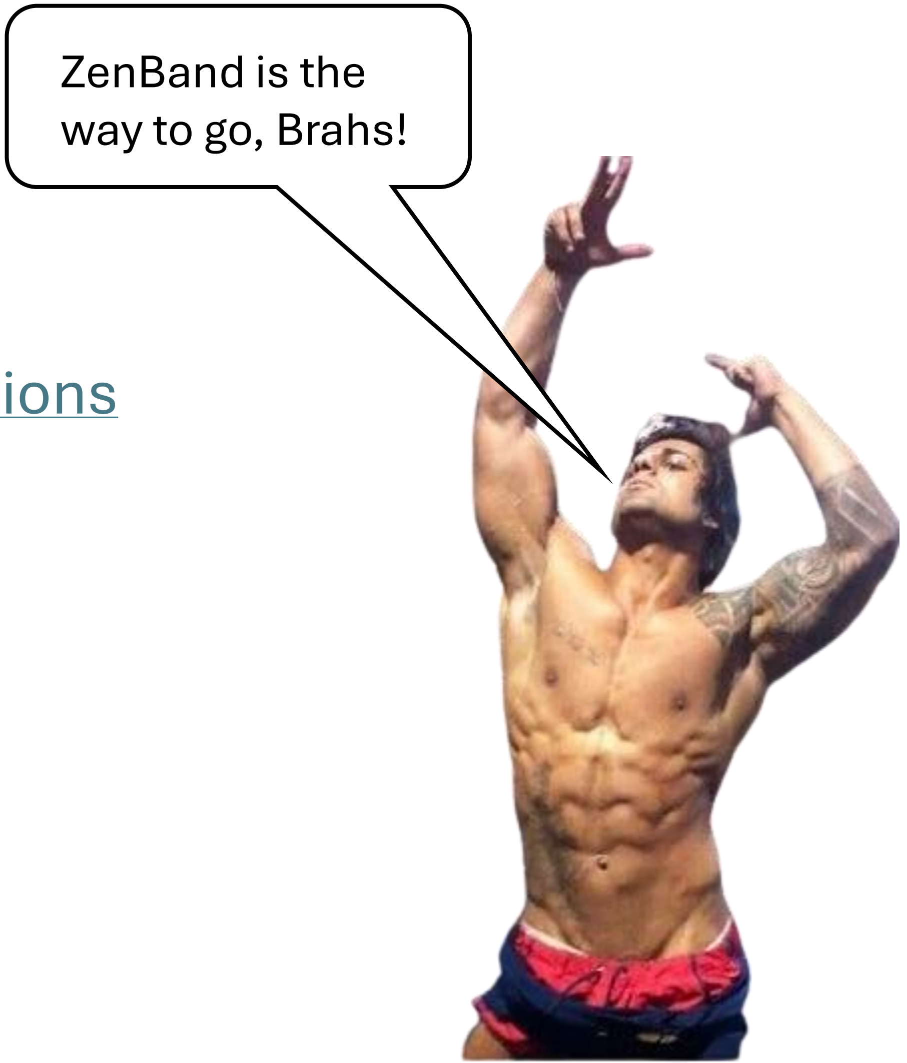
ZenBand

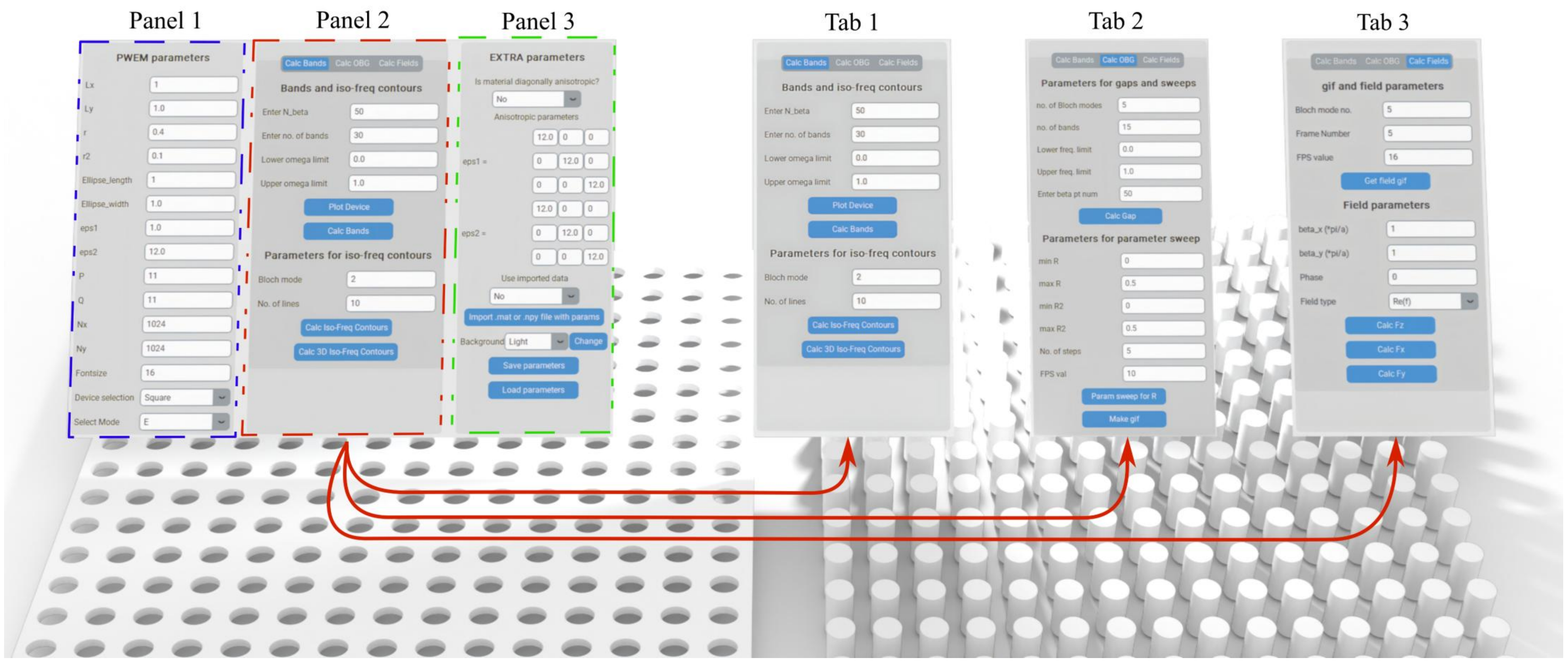
GUI guide for the only 2D PWEM solver you’ll ever need!

Contents

Recommended by legends: Zyzz



Layout



PWEM parameters

Lx	1
Ly	1.0
r	0.4
r2	0.1
Ellipse_length	1
Ellipse_width	1.0
eps1	1.0
eps2	12.0
P	11
Q	11
Nx	1024
Ny	1024
Fontsize	16
Device selection	Square
Select Mode	E

Bands and iso-freq contours

Enter N_beta	50
Enter no. of bands	30
Lower omega limit	0.0
Upper omega limit	1.0

Parameters for iso-freq contours

Bloch mode	2
No. of lines	10

EXTRA parameters

Is material diagonally anisotropic? No

Anisotropic parameters

$$eps1 = \begin{matrix} 12.0 & 0 & 0 \\ 0 & 12.0 & 0 \\ 0 & 0 & 12.0 \end{matrix}$$

$$eps2 = \begin{matrix} 12.0 & 0 & 0 \\ 0 & 12.0 & 0 \\ 0 & 0 & 12.0 \end{matrix}$$

Use imported data: No

Background: Light

Parameters for gaps and sweeps

no. of Bloch modes	5
no. of bands	15
Lower freq. limit	0.0
Upper freq. limit	1.0
Enter beta pt num	50

Parameters for parameter sweep

min R	0
max R	0.5
min R2	0
max R2	0.5
No. of steps	5
FPS val	10

gif and field parameters

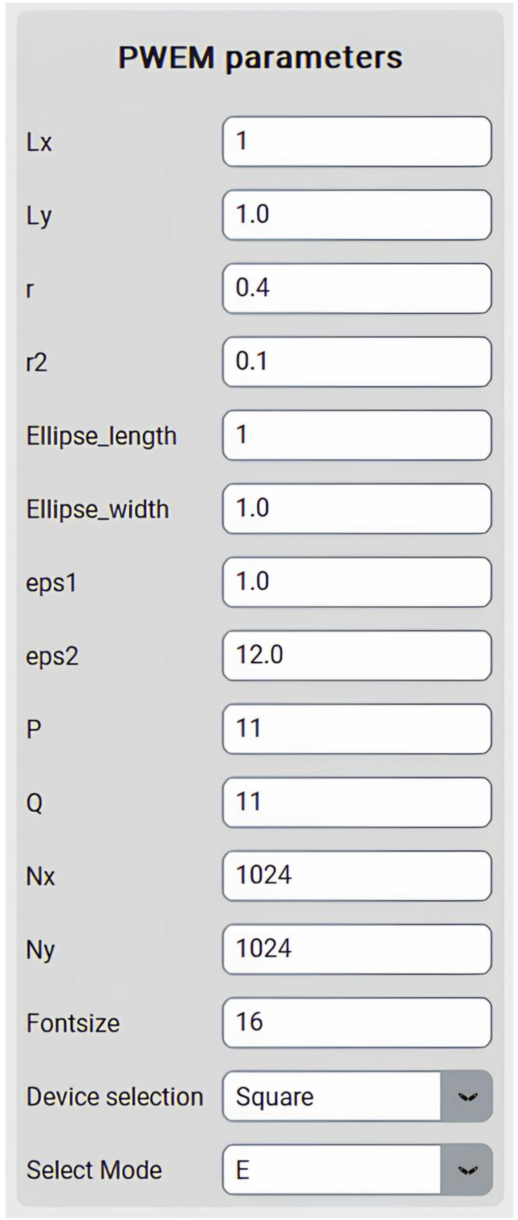
Bloch mode no.	5
Frame Number	5
FPS value	16

Field parameters

beta_x (*pi/a)	1
beta_y (*pi/a)	1
Phase	0
Field type	Re(f)

The app is distinguished into three different panels:
Panel 1 – PWEM Parameters;
Panel 2 – houses functions for PWEM solver, separated into three tabs;
Panel 3 – Extra Parameters.

Panel 1 – PWEM parameters



PWEM parameters

Lx	1
Ly	1.0
r	0.4
r2	0.1
Ellipse_length	1
Ellipse_width	1.0
eps1	1.0
eps2	12.0
P	11
Q	11
Nx	1024
Ny	1024
Fontsize	16
Device selection	Square
Select Mode	E

Here you can find general parameters that describe the device and computational performance.

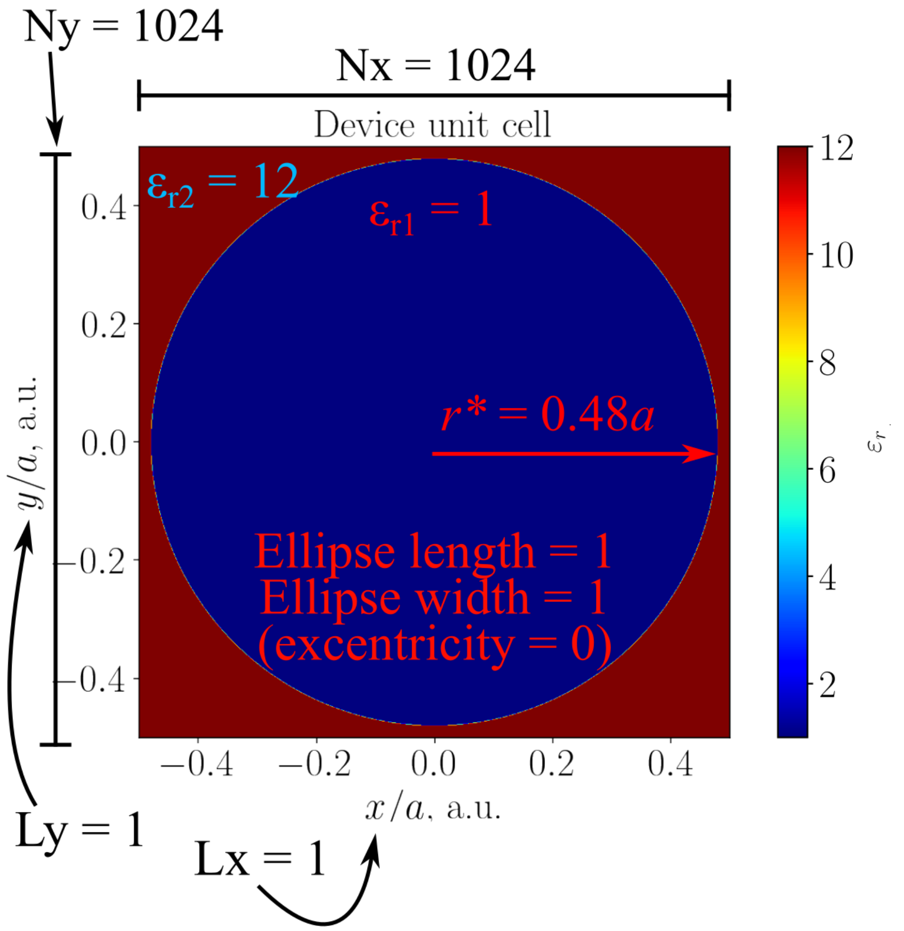
Device's unit cell is described by these variables:
Lx and *Ly* – number of periods in the given direction;
r – size of the cylinder (normalized to lattice constant a);
r2 – size of another cylinder (only works for the smaller *Ring* cylinder or second *Honeycomb* cylinder);
Ellipse_width/length – controls the eccentricity of the cylinder;
eps1 – dielectric permittivity of the cylinder;
eps2 – dielectric permittivity of the slab;
Device selection – let's the user choose between *Square, Frame, Ring, Hex* and *Honeycomb* devices.

Computational performance can be adjusted with these parameters:
P, Q – number of spatial harmonics used for plane wave expansion. Run time increases exponentially. **The number must be odd!**
Nx, Ny – grid size for the unit cell.

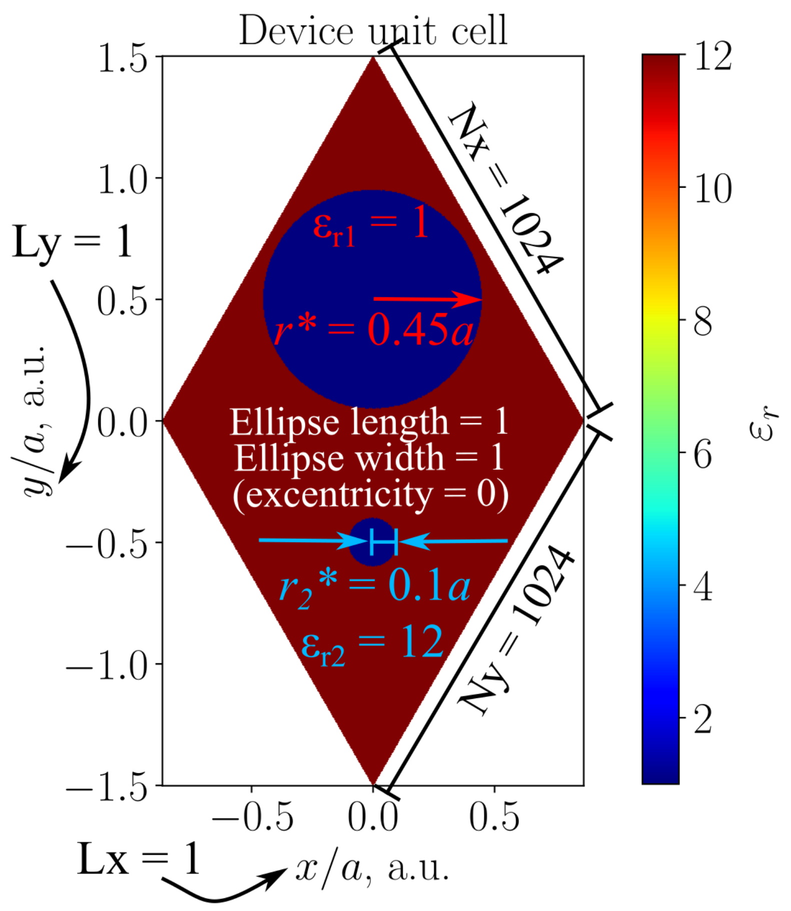
Fontsize – changes the fontsize of the matplotlib plots.
Select mode – selects the mode for iso-frequency contours and field distributions.

Example – device parameters

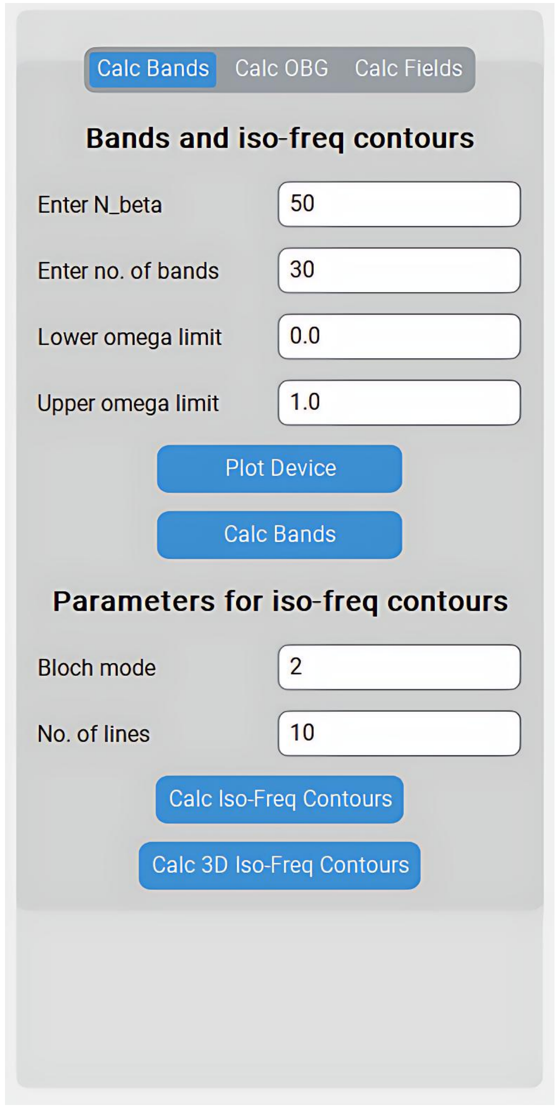
Device selection: Square



Device selection: Honeycomb



Panel 2: Tab1 – Band Diagram Calculations



Bands and iso-freq contours

Enter N_beta	50
Enter no. of bands	30
Lower omega limit	0.0
Upper omega limit	1.0

Parameters for iso-freq contours

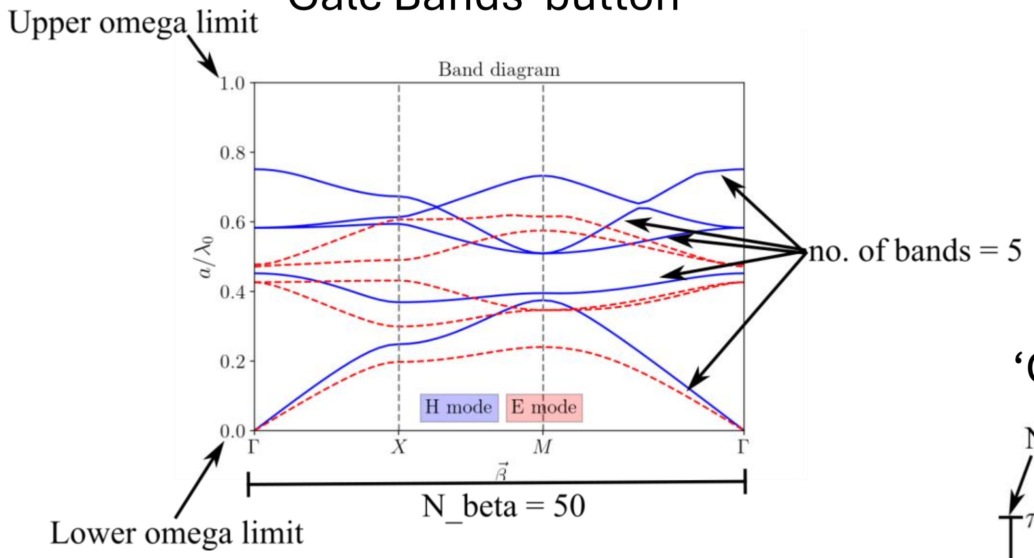
Bloch mode	2
No. of lines	10

'Bands and iso-freq contours' parameters:
Enter N_beta – the number of beta vector points on the band diagrams and iso-frequency contours (for contours, it shows the number of lines and columns, e.g. 50x50 = 2500).
Enter no. of bands – the number of photonic bands that are displayed on band diagram.
Lower omega limit – changes the lower limit of *y* axis in band diagrams.
Upper omega limit – changes the upper limit of *y* axis in band diagrams.

'Parameters for iso-freq contours' :
Bloch mode – selects which Bloch mode is displayed in iso-frequency contours.
No. of lines – selects the number of iso-frequency lines on the plot.

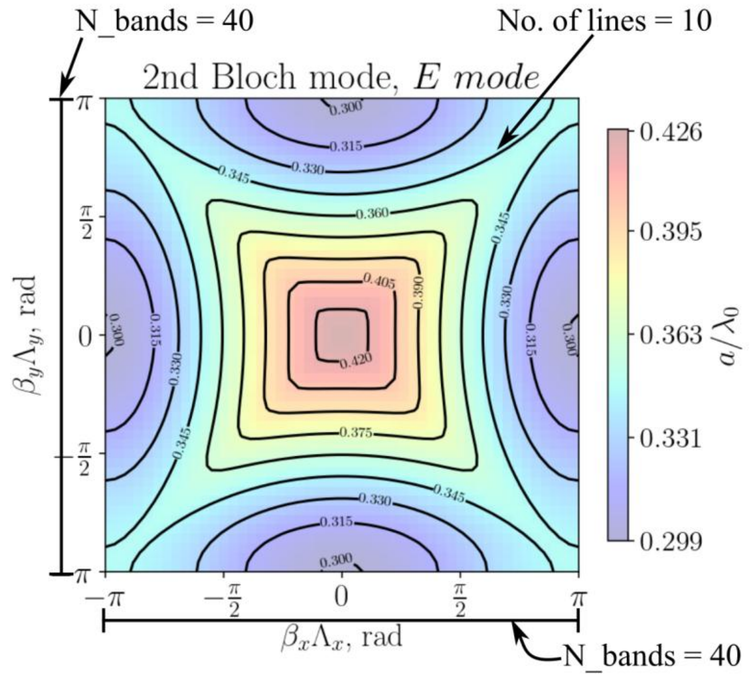
Example – Calc Bands

'Calc Bands' button

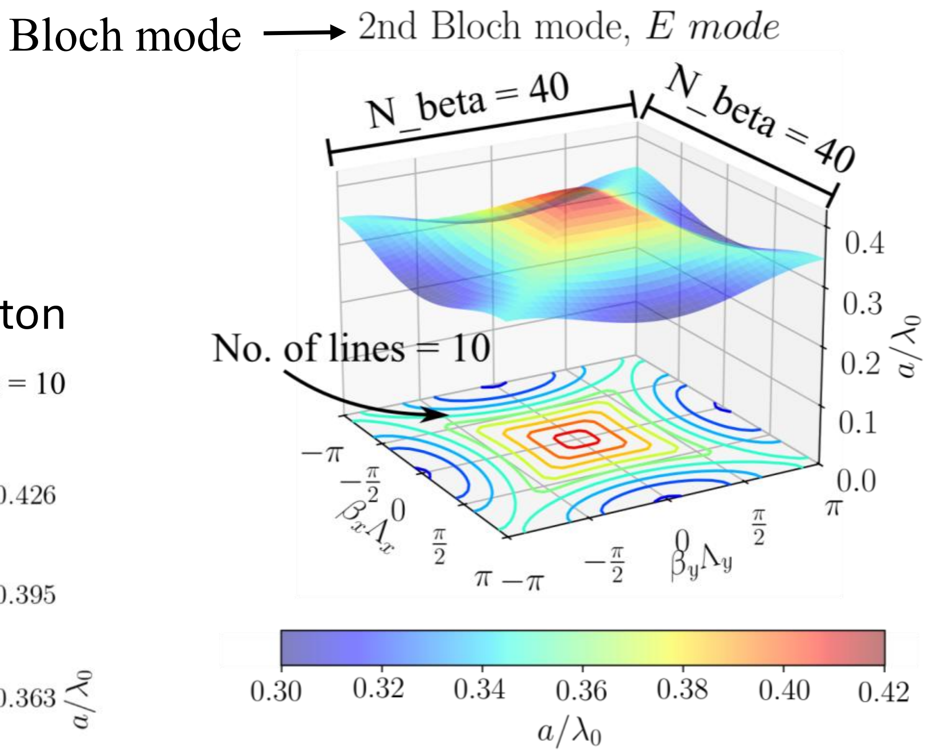


The number of bands can not exceed P*Q!

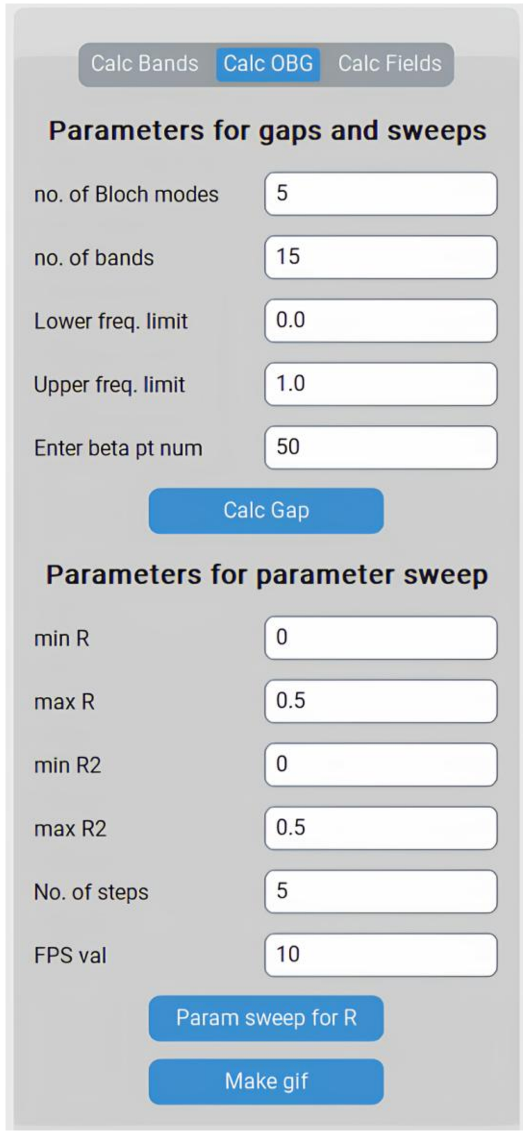
'Calc Iso-Freq Contours' button



'Calc 3D Iso-Freq Contours' button



Panel 2: Tab2 – Omnidirectional Band Gap Calculations



Parameters for gaps and sweeps

no. of Bloch modes	5
no. of bands	15
Lower freq. limit	0.0
Upper freq. limit	1.0
Enter beta pt num	50

Parameters for parameter sweep

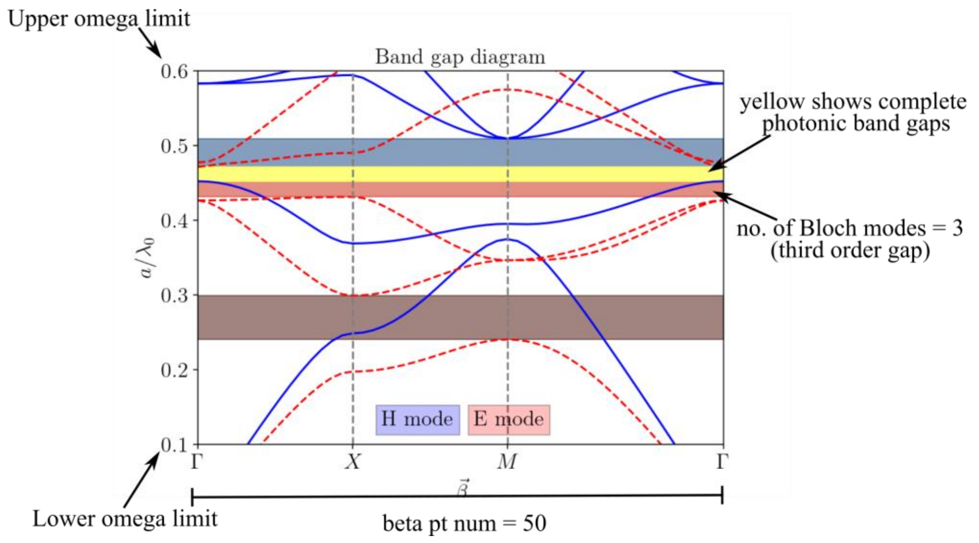
min R	0
max R	0.5
min R2	0
max R2	0.5
No. of steps	5
FPS val	10

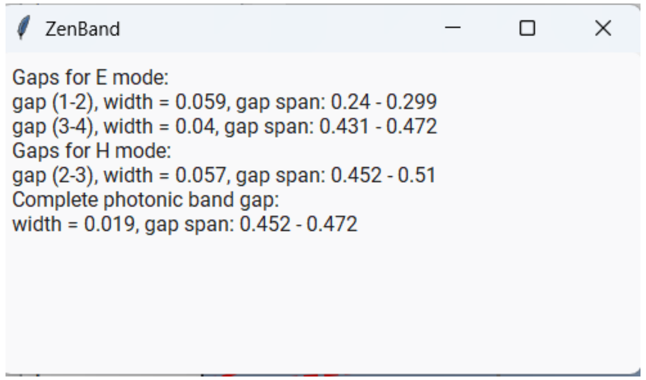
'Parameters for gaps and sweeps':
no. of Bloch modes – choose the highest order band gap to search for;
no. of bands – the number of photonic bands that are displayed on band diagram;
Lower/Upper freq. limit – changes the lower/upper limit of *y* axis in band diagrams;
Enter beta pt num - the number of beta vector points on the band diagrams .

'Parameters for parameter sweep':
min R – start value of cylinder radius;
max R – end value for cylinder radius;
min/max R2 – start/end value for second cylinder radius (works for *Ring* and *Honeycomb* lattices);
No. of steps – select number of intermediate radius values;
Fps val – frames per second for .gif file.

Example – Calc OBG

‘Calc Gap’ button

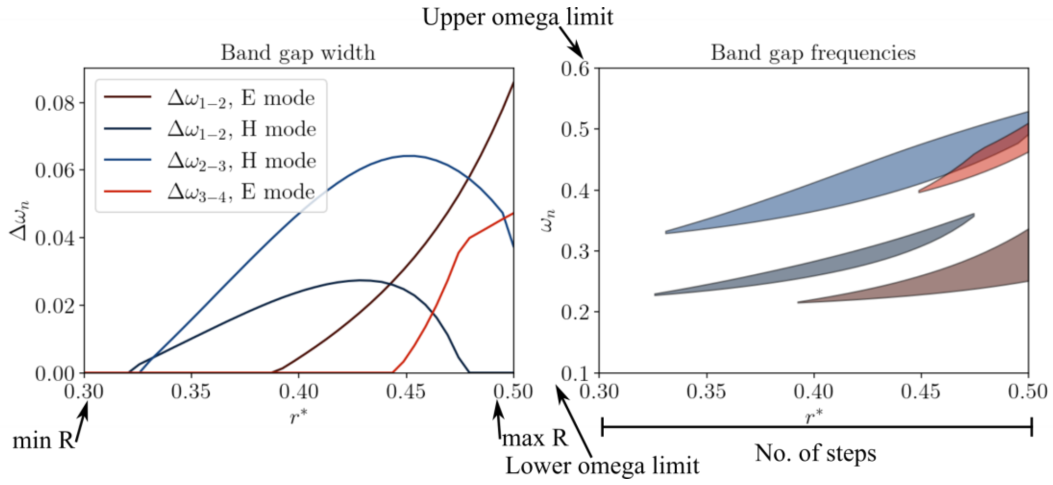




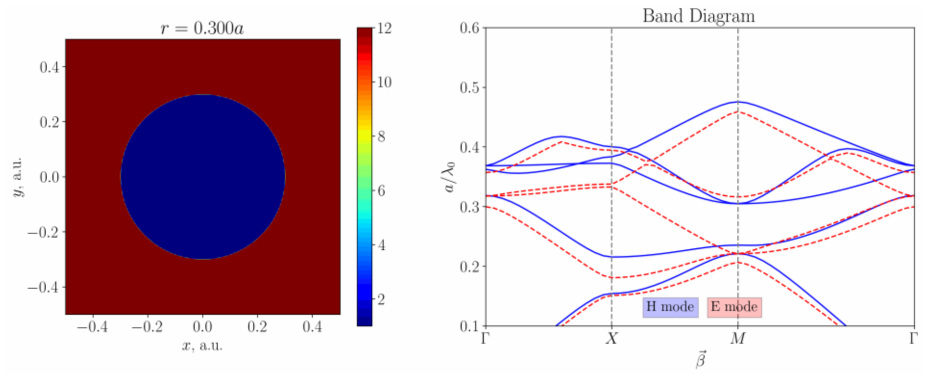
```
Gaps for E mode:
gap (1-2), width = 0.059, gap span: 0.24 - 0.299
gap (3-4), width = 0.04, gap span: 0.431 - 0.472
Gaps for H mode:
gap (2-3), width = 0.057, gap span: 0.452 - 0.51
Complete photonic band gap:
width = 0.019, gap span: 0.452 - 0.472
```

A top window is created that gives a detailed overview of the gaps.

‘Param sweep for R’ button

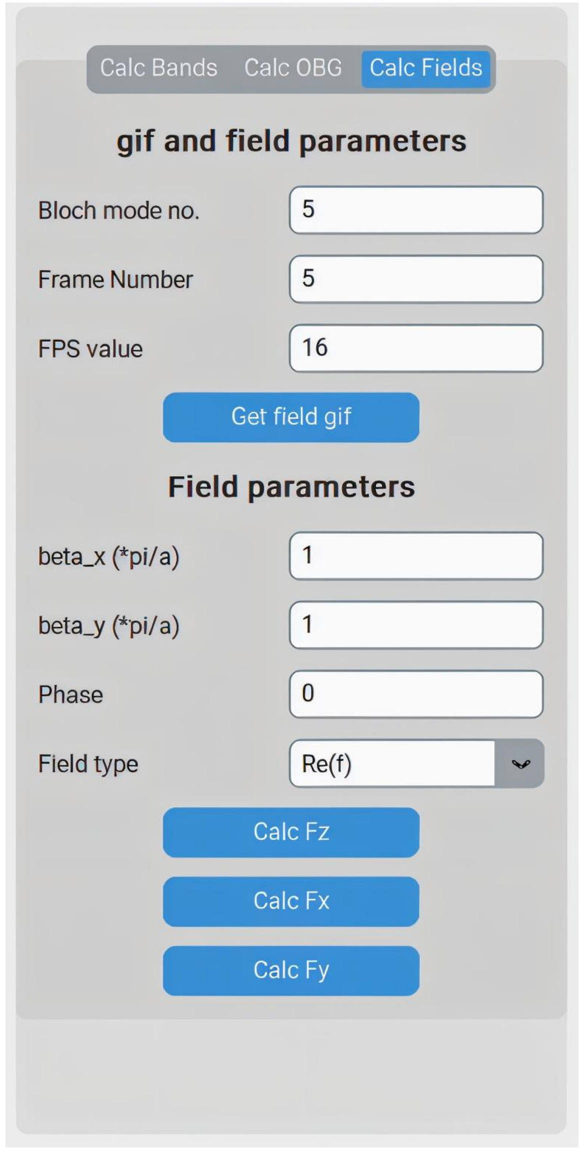


‘Make gif’ button



min R = 0.3;
max R = 0.5;
No. of steps = 40;
Fps = 10.

Panel 2: Tab3 – Field Calculations



gif and field parameters

Bloch mode no.	5
Frame Number	5
FPS value	16

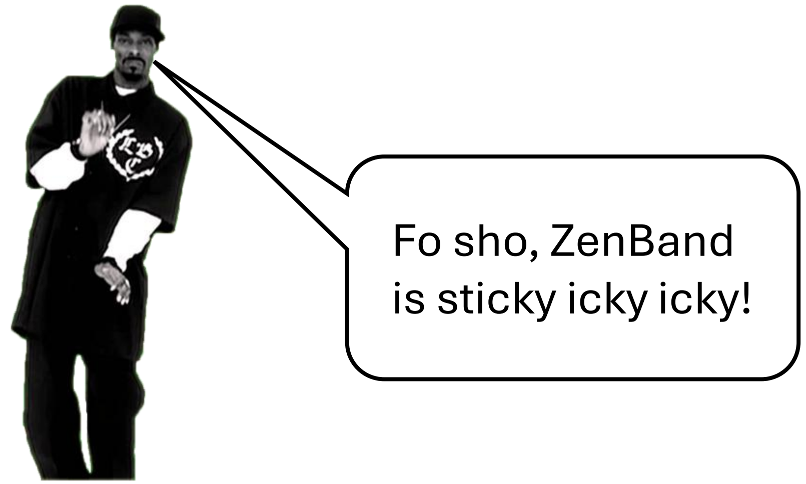
Field parameters

beta_x (*pi/a)	1
beta_y (*pi/a)	1
Phase	0
Field type	Re(f)

‘gif and field parameters’:
Bloch mode no. – choose the Bloch mode;
Frame Number – the number of frames for .gif file;
FPS value – frames per second of the .gif file.

‘Field parameters’:
*beta_x(*pi/a)* – point in the x direction in the reciprocal lattice of the field;
*beta_y(*pi/a)* – point in the y direction in the reciprocal lattice of the field;
Phase – shift the phase of the real field;
Field type – choose between real part of the field and intensity distributions for the graph.

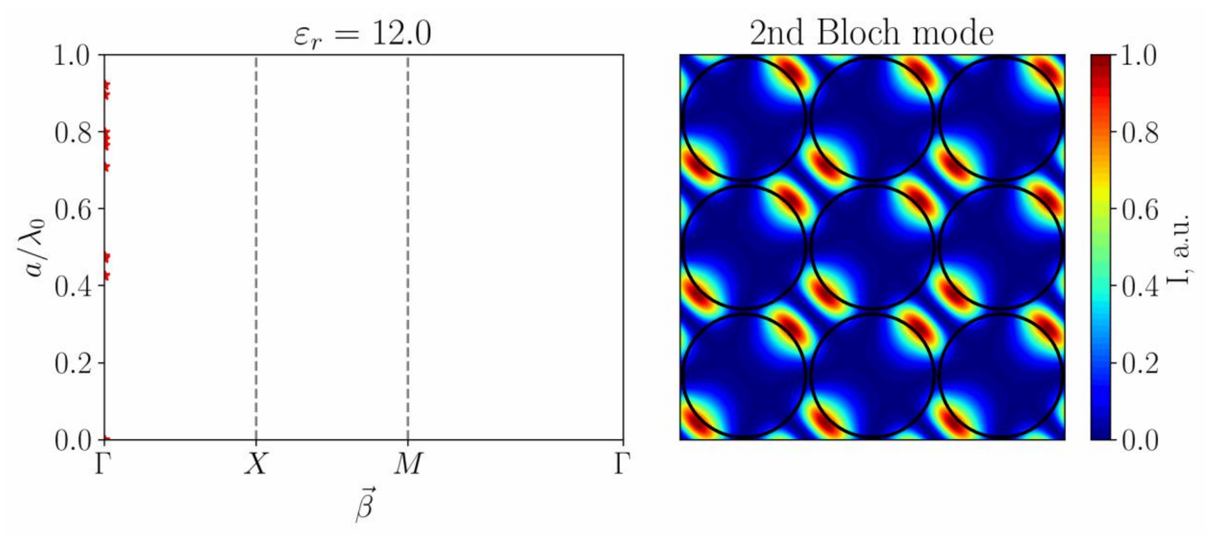
Recommended by legends: Snoop Dogg



Example – Calc Fields

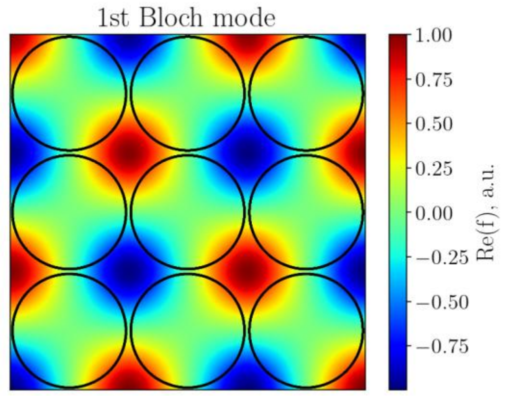
M point (*beta_x(*pi/a) = 1, beta_y(*pi/a) = 1);*
Phase = 0;
Field type = Re(f).

‘Get field gif’ button

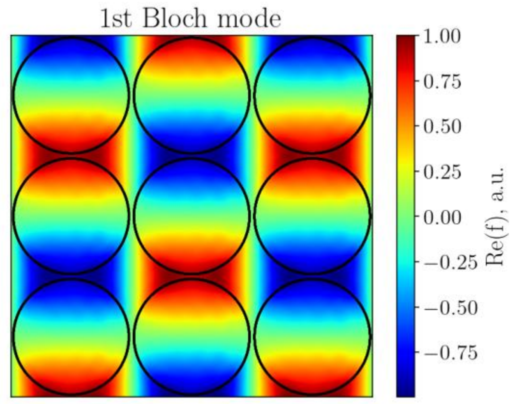


Frame number = 40;
Fps value = 10;
Mode = E.

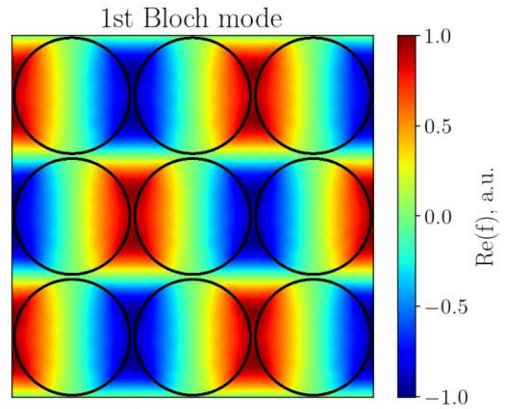
‘Calc Fz’ button



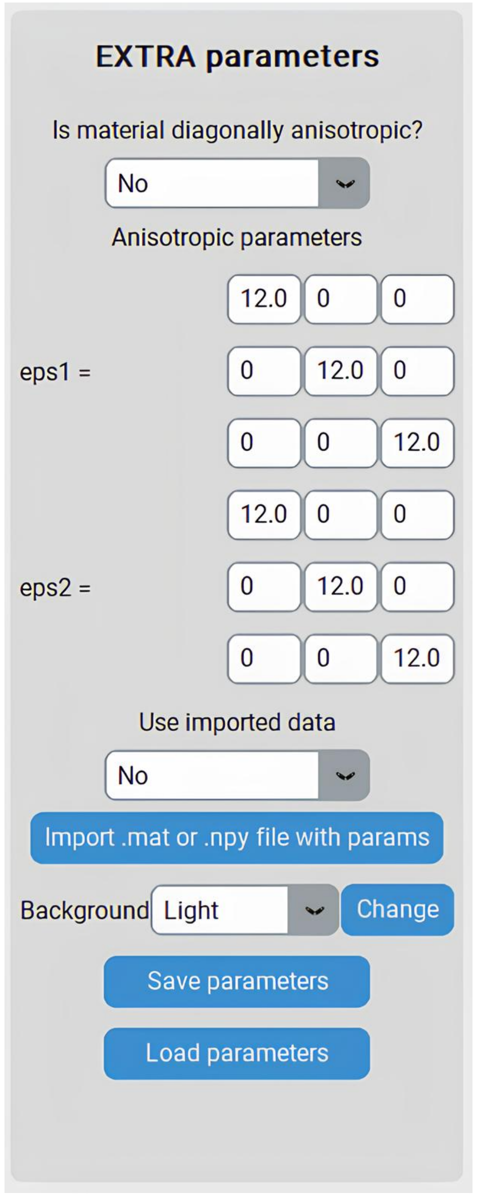
‘Calc Fx’ button



‘Calc Fy’ button



Panel 3 – Extra Parameters



EXTRA parameters

Is material diagonally anisotropic?

No

Anisotropic parameters

eps1 =

12.0	0	0
0	12.0	0
0	0	12.0

eps2 =

12.0	0	0
0	12.0	0
0	0	12.0

Use imported data

No

Import .mat or .npy file with params

Background: Light — Change

Save parameters

Load parameters

This panel is dedicated to extra, more advanced options that a user might want.

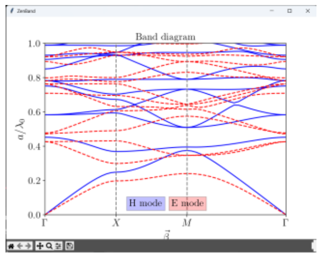
The first option is to define a diagonally anisotropic material. This is a separate option in order to save a fraction of computational time and for user convenience.

An option to import a custom device is also available. The program supports .mat and .npy files. More information on how to build custom structures will be presented later.

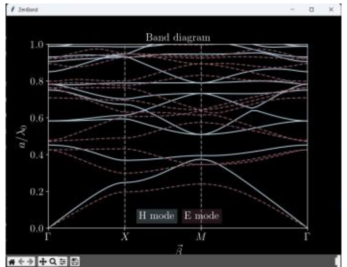
A user can choose between light and dark themes of the app to make the experience more pleasant in darker environments or to better match the figure color theme.

Since there are number of parameters to adjust, an option to save and load the parameters was added. Once every entry is filled, press the ‘Save parameters’ button. Whenever you want to go back to the parameters that were saved, press ‘Load parameters’ button. The feature also works after the app was closed.

Light mode



Dark mode



Importing data

You can import data using .npy and .mat files. If band diagram or field distributions are wanted, it is very important that these parameters are defined:

Unit cell grid, direct lattice vectors, reciprocal lattice vectors, beta vector array, key points of symmetry, names of the key points of symmetry.

To calculate iso-frequency contour diagrams, key points of symmetry are not needed.

It is also important that the arrays are named correctly in such a manor (Python example):

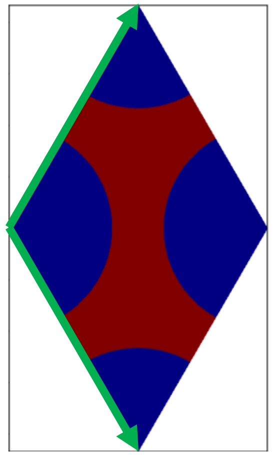
data = {'er':ER, 'beta':beta, 't1':t1, 't2':t2, 'T1':T1, 'T2':T2, 'KP':KP, 'KT':KT} # for bands and fields;

data = {'er':ER, 'beta':beta, 't1':t1, 't2':t2, 'T1':T1, 'T2':T2} # for contours.

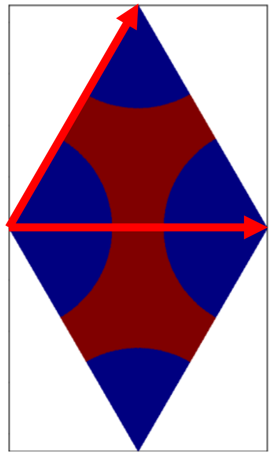
Make sure that primittive lattice vectors define the perimeter of the unit cell, because they are used for *plt.pcolor* plot coordinates!

‘Parameter sweep for R’ and ‘Make gif’ buttons do not work when parameters are imported!

Correct way to define vectors



Incorrect way



Importing data: Python example

Data for bands

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

##############################################
# initialize parameters
er1 = 1
er2 = 12
r = 0.45
N_Points = 100

x = np.linspace(-0.5, 0.5, 1024)
y = np.linspace(-0.5, 0.5, 1024)
X,Y = np.meshgrid(x, y)
###############################################

ER = np.zeros((len(x), len(y)))

# define the shape
ER = ER + (X**2 + Y**2 > r**2)
ER  = er1 + (er2 - er1) * ER
# plt.pcolor(ER)
################## Bands #####################

# define beta vectors
beta = np.zeros((2, N_Points))
beta[0, 0:29] = np.linspace(0, np.pi, 29); beta[0, 29:59] = np.pi;            beta[0, 59:100] = np.linspace(np.pi, 0, 41)
beta[1, 0:29] = 0;            beta[1, 29:59] = np.linspace(0, np.pi, 30); beta[1, 59:100] = np.linspace(np.pi, 0, 41)

# define names of key points of symmetry
KP = ['$\Gamma$', '$X$', '$M$', '$\Gamma$']

# define key points of symmetry
KT = [0, 29, 58, 99]

# define direct lattice vectors
t1    = np.array([[1],[0]])
t2    = np.array([[0],[1]])

# define reciprocal lattice vectors
T1    = 2*np.pi * np.array([[1],[0]])
T2    = 2*np.pi * np.array([[0],[1]])

data = {'er':ER, 'beta':beta, 't1':t1, 't2':t2, 'T1':T1, 'T2':T2, 'KP':KP, 'KT':KT}
np.save('import_sample.npy', data)
```

Data for contours

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

############################################
# initialize parameters
er1 = 1
er2 = 12
r = 0.45
N_Points = 100

############################################

ER = np.zeros((len(x), len(y)))

# define the shape
ER = ER + (X**2 + Y**2 > r**2)
ER  = er1 + (er2 - er1) * ER
# plt.pcolor(ER)
################## Contours ##################

# define beta vectors
bx    = np.linspace(-np.pi,  np.pi, N_Points)
by    = np.linspace( np.pi, -np.pi, N_Points)
beta  = np.zeros((2, N_Points**2))

idx = 0

for nx in range(0, N_Points):
   for ny in range(0, N_Points):
      beta[:,idx] = np.array(([bx[nx], by[ny]]))
      idx = idx + 1

# define direct lattice vectors
t1    = np.array([[1],[0]])
t2    = np.array([[0],[1]])

# define reciprocal lattice vectors
T1    = 2*np.pi * np.array([[1],[0]])
T2    = 2*np.pi * np.array([[0],[1]])

data = {'er':ER, 'beta':beta, 't1':t1, 't2':t2, 'T1':T1, 'T2':T2}
np.save('import_sample.npy', data)
```

Recommended by legends: Big Chungus