PythTB for (topological) tight-binding models

Jennifer Cano

Stony Brook University and

Flatiron Institute for Computational Quantum Physics

PythTB is based at: http://physics.rutgers.edu/pythtb/

What is PythTB?

- PythTB is a software package providing a Python implementation for tight-binding models.

- Developed by Sinisa Coh and David Vanderbilt

PythTB is based at: http://physics.rutgers.edu/pythtb/

Making tight-binding models is easy. Why should I use PythTB?

- Only work in real space.
- Easily compute band structure and get eigenvectors.
- Easily create slab, cube, or other finite boundary conditions.
- Easily compute Berry phase or plot Wilson loop eigenvalues.

PythTB is based at: http://physics.rutgers.edu/pythtb/

But I don’t know Python….

- Learning Python is a good time investment.
- Let this be your excuse to learn it.
- Truth: the code is so easy, you don’t even need to know Python to use it.

PythTB is based at: http://physics.rutgers.edu/pythtb/

Goals

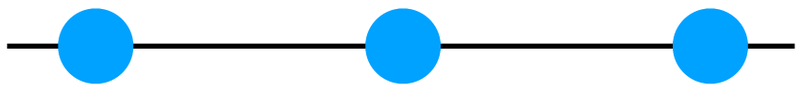
- Today: plot band structures for models from the previous lecture.
- Tomorrow: implement topological tight-binding models in PythTB to visualize topological surface states and Berry phase.

PythTB is based at: http://physics.rutgers.edu/pythtb/

Example 1: atoms in 1d

$$H(\mathbf{R} = \pm\hat{x}) = -t$$

$$H(\mathbf{R} \neq \pm\hat{x}) = 0$$



Lattice vectors (we only have one)

Orbitals in units of lattice vets (we only have one)

Define model: (dim k space, dim real space, lattice vecs, orbital vecs)

Hopping term: (amplitude, iα, jβ, **R**)

k path in units of reciprocal lattice vecs

Labels of k-points on path

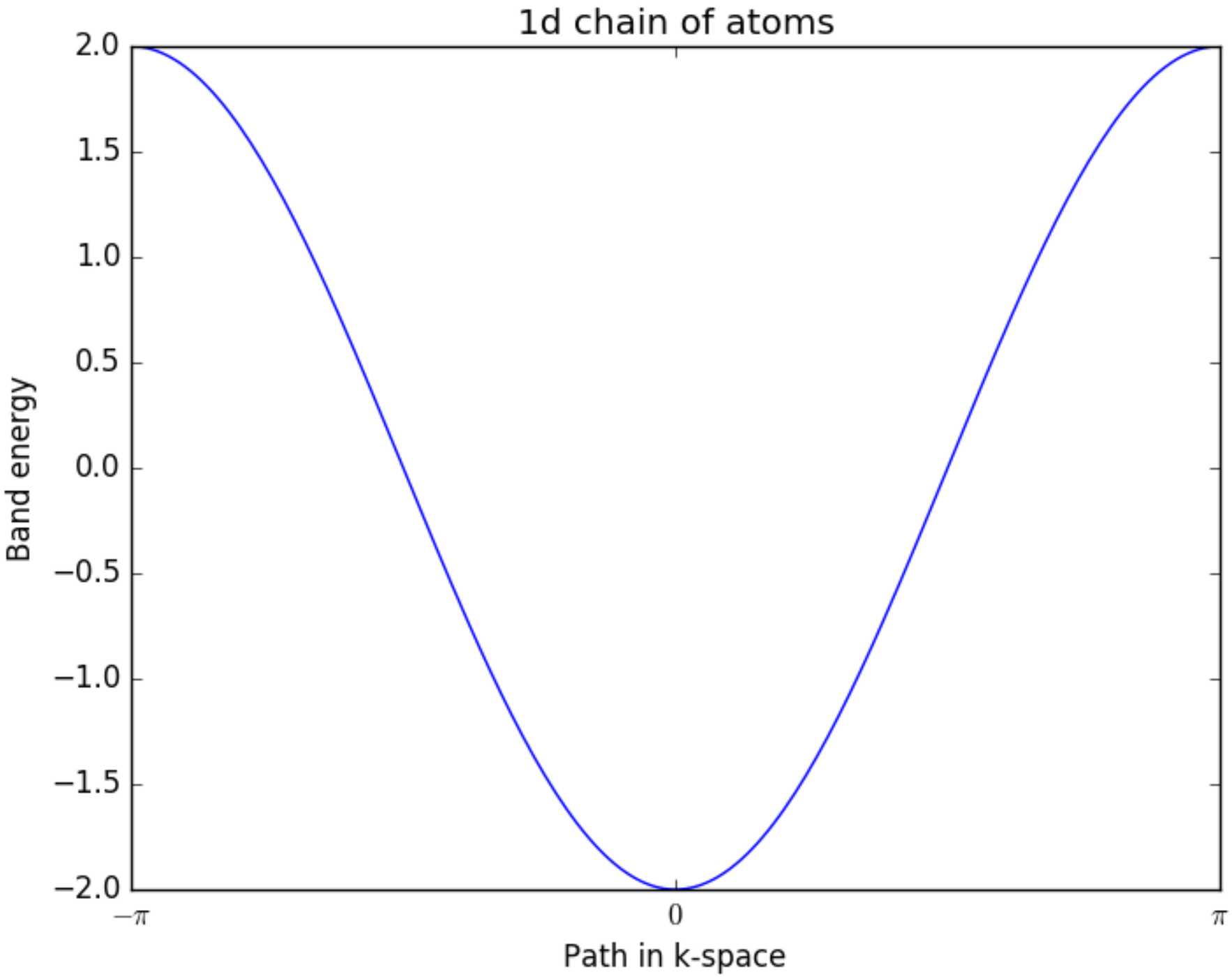
Repeat this line for more bands, up to evals[n]

Vector of ticks based on path and numsteps

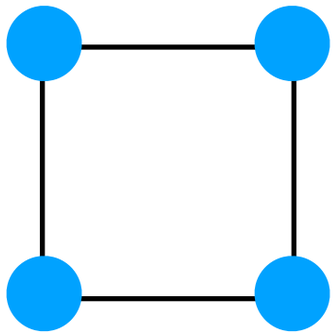
```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 24 21:23:16 2018

@author: jennifercano
"""

# adaptation from "simple example"
# at http://physics.rutgers.edu/pythtb/examples.html

from pythtb import * # import TB model class
import numpy as np
import matplotlib.pyplot as plt

# specify model
# lattice vectors
lat=[[1.0]]
# positions of orbitals
orb=[[0.0]]

# define the model
my_model=tb_model(1,1,lat,orb)
# assign hopping terms
my_model.set_hop(-1., 0, 0, [1])

# define a path in k-space to plot
path=[[-.5],[0],[.5]]
# label k points
label=(r'$-\pi$',r'$0$',r'$\pi $')
# number of steps between points
numsteps=100
kpts=k_path(path,numsteps)

# solve model
evals=my_model.solve_all(kpts)

# make a figure object
fig=pl.figure()
# plot bands
pl.plot(evals[0])
# put title on top
pl.title("1d chain of atoms")
pl.xlabel("Path in k-space")
pl.ylabel("Band energy")
pl.xticks([0,100,200],label)
```

PythTB is based at: http://physics.rutgers.edu/pythtb/

Example 1: atoms in 1d



PythTB is based at: http://physics.rutgers.edu/pythtb/



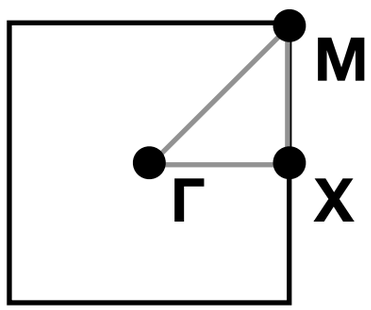
Example 2: square lattice

$$H(\mathbf{R}) = \begin{cases} -t & \text{if } \mathbf{R} = \pm\hat{x}, \pm\hat{y} \\ 0 & \text{else} \end{cases}$$

Two two-component lattice vecs

Still one orbital; now two components

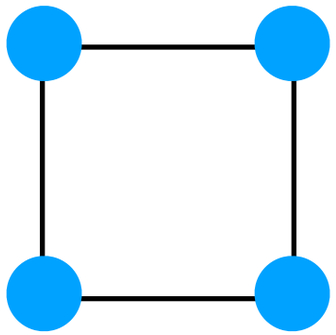
```
11 from pythtb import * # import TB model class
12 import numpy as np
13 import matplotlib.pyplot as plt
14
15 # specify model
16 # lattice vectors
17 lat=[[1.0,0.0],[0.0,1.0]]
18 # positions of orbitals
19 orb=[[0.0,0.0]]
20
21 # define the model
22 my_model=tb_model(2,2,lat,orb)
23 # assign hopping terms
24 # x-hopping
25 my_model.set_hop(-1., 0, 0, [1.0,0])
26 # y-hopping
27 my_model.set_hop(-1., 0, 0, [0,1.0])
28
29 # define a path in k-space to plot
30 path=[[0.0,0.0],[.5,0],[.5,.5],[0.0,0.0]]
31 # label k points
32 label=(r'$\Gamma$',r'$X$',r'$M$',r'$\Gamma$')
33 # number of steps between points
34 numsteps=100
35 kpts=k_path(path,numsteps)
36
37 # solve model
38 evals=my_model.solve_all(kpts)
39
40 # make a figure object
41 fig=pl.figure()
42 # plot bands
43 pl.plot(evals[0])
44 # put title on top
45 pl.title("2d square lattice")
46 pl.xlabel("Path in k-space")
47 pl.ylabel("Band energy")
48 pl.xticks([0,100,200,300],label)
```



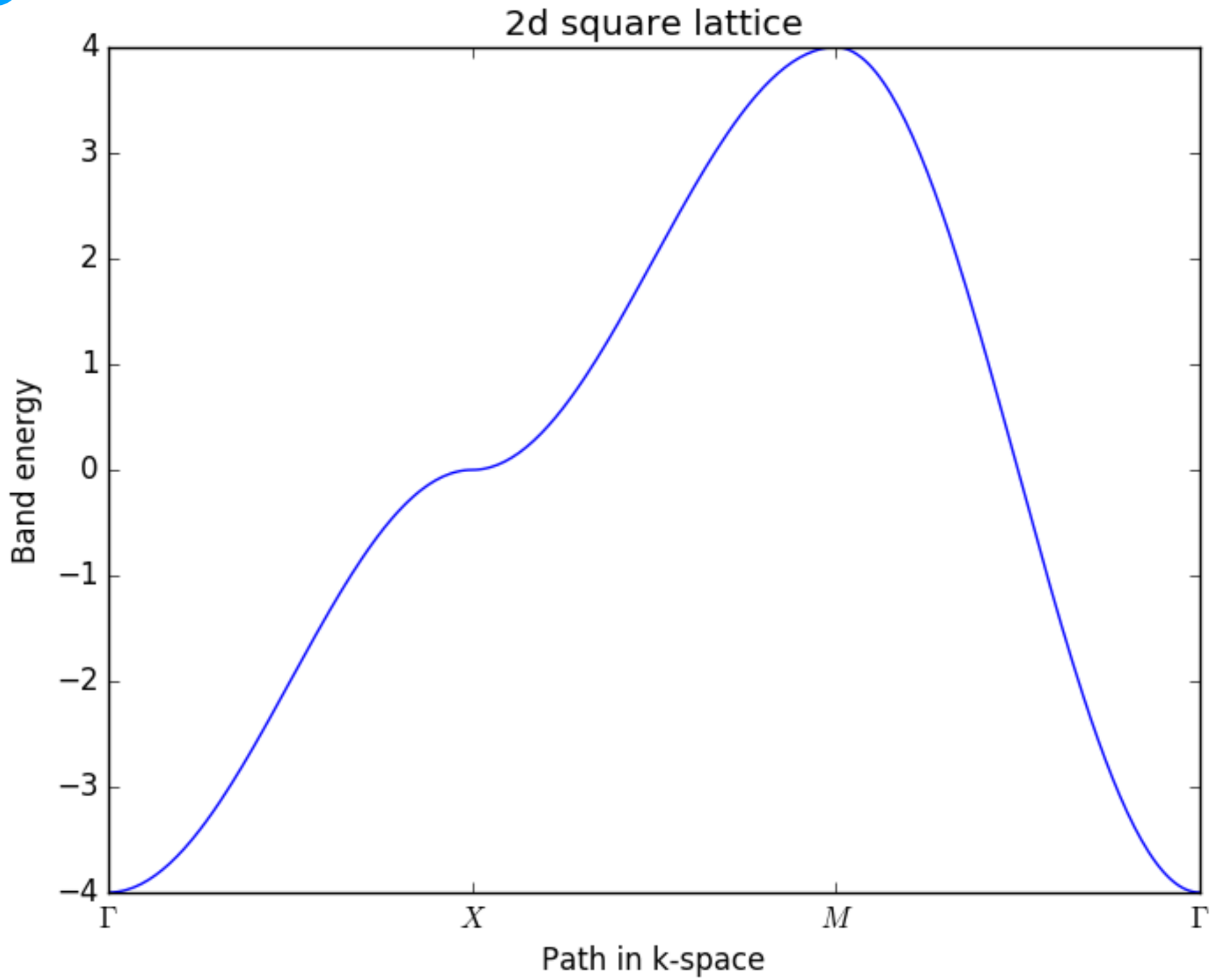
k path: Γ, X, M, Γ

Added another tick because longer k path

PythTB is based at: http://physics.rutgers.edu/pythtb/

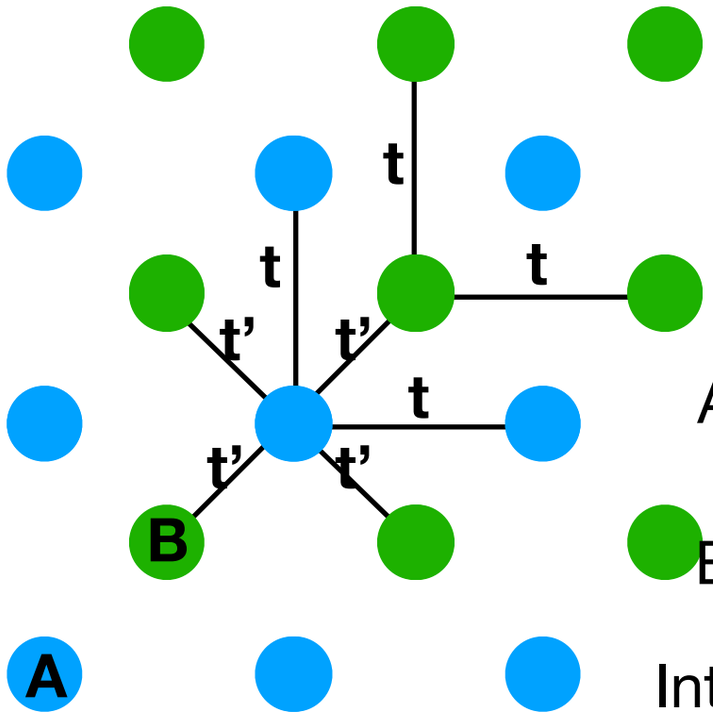


Example 2: square lattice



PythTB is based at: http://physics.rutgers.edu/pythtb/

Example 3: interpenetrating square lattices



Two orbitals!

Onsite energy for each orbital

A sublattice hopping

B sublattice hopping

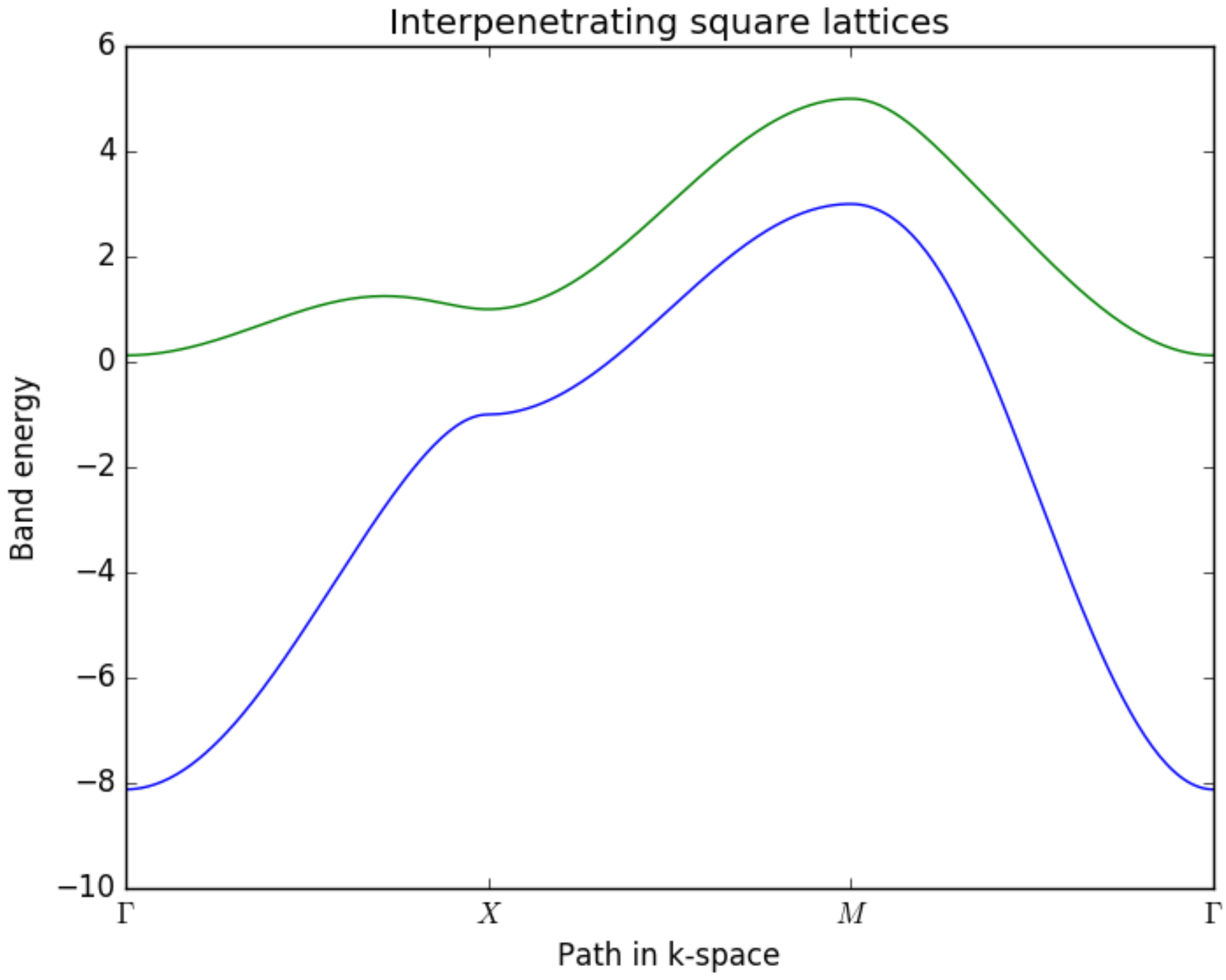
Inter-sublattice hopping (last term is R)

Now plotting two bands

PythTB is based at: http://physi

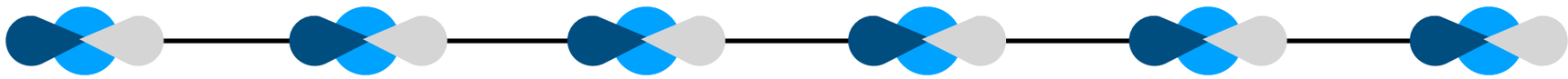
```
# specify model
# lattice vectors
lat=[[1.0,0.0],[0.0,1.0]]
# positions of orbitals
orb=[[0.0,0.0],[.5,.5]]

# define the model
my_model=tb_model(2,2,lat,orb)

# assign onsite energy
my_model.set_onsite([1.0,-1.0])

# assign hopping terms
t=1.0
t2=1.0
# x-hopping within sublattice of orbital "0"
my_model.set_hop(-t, 0, 0, [1.0,0])
# y-hopping within sublattice of orbital "0"
my_model.set_hop(-t, 0, 0, [0,1.0])
# x-hopping within sublattice of orbital "1"
my_model.set_hop(-t, 1, 1, [1.0,0])
# y-hopping within sublattice of orbital "1"
my_model.set_hop(-t, 1, 1, [0,1.0])
# four inter-sublattice hopping terms, from "0" to "1"
my_model.set_hop(-t2, 0, 1, [0.0,0.0])
my_model.set_hop(-t2, 0, 1, [-1.0,0.0])
my_model.set_hop(-t2, 0, 1, [-1.0,-1.0])
my_model.set_hop(-t2, 0, 1, [0.0,-1.0])

# define a path in k-space to plot
path=[[0.0,0.0],[.5,0],[.5,.5],[0.0,0.0]]
# label k points
label=(r'$\Gamma$',r'$X$',r'$M$',r'$\Gamma$')
# number of steps between points
numsteps=100
kpts=k_path(path,numsteps)

# solve model
evals=my_model.solve_all(kpts)

# make a figure object
fig=pl.figure()
# plot bands
pl.plot(evals[0])
pl.plot(evals[1])
# put title on top
pl.title("Interpenetrating square lattices")
pl.xlabel("Path in k-space")
pl.ylabel("Band energy")
pl.xticks([0,100,200,300],label)
```

Example 3: interpenetrating square lattices



PythTB is based at: http://physics.rutgers.edu/pythtb/

Exercise 1: implement 1d chain with inversion, s and p orbitals



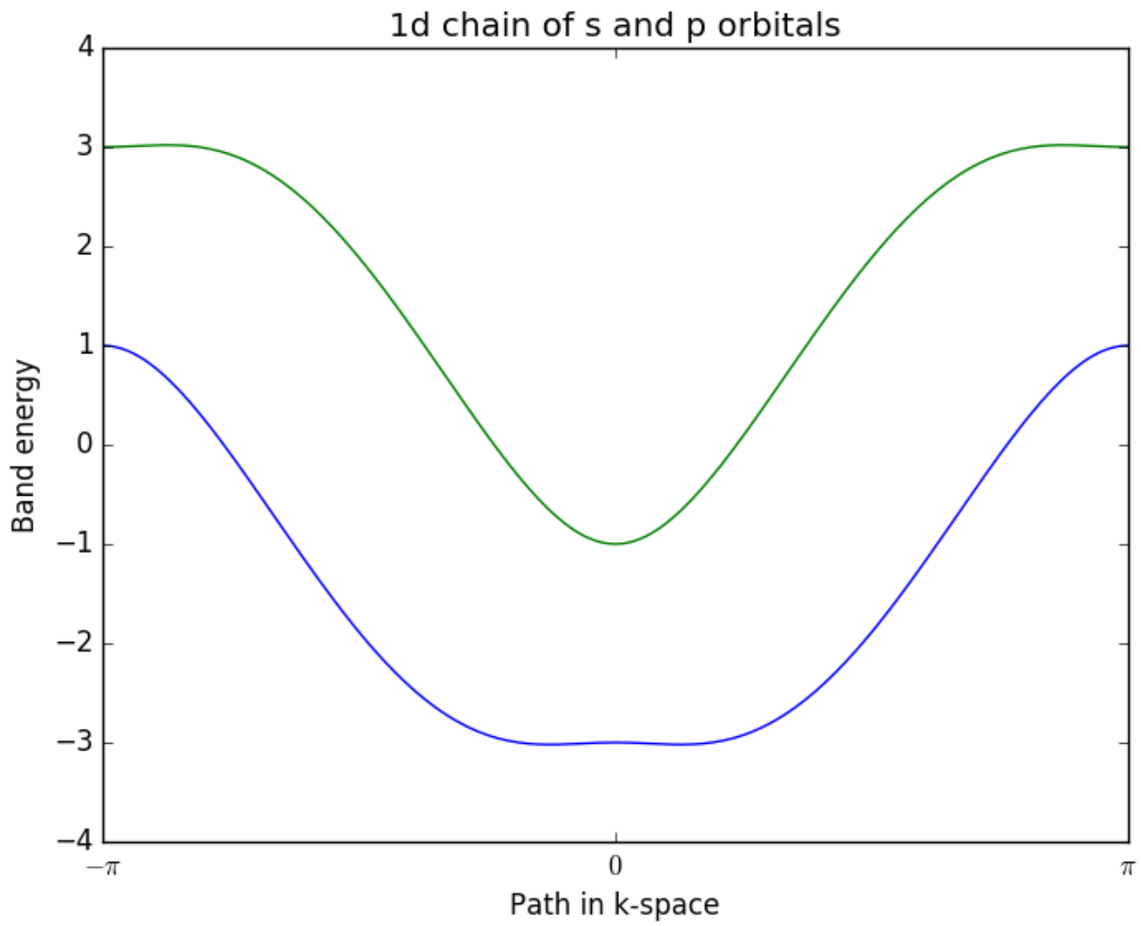
$$U = \sigma_z \qquad H^k = \begin{pmatrix} \mu_s & 0 \\ 0 & \mu_p \end{pmatrix} + \begin{pmatrix} -t_s & 0 \\ 0 & -t_p \end{pmatrix} \cos k + \begin{pmatrix} 0 & -it_{sp} \\ it_{sp} & 0 \end{pmatrix} \sin k$$

onsite — nearest neighbor, same orbital — nearest neighbor, different orbital orbital

$$H_{ss(pp)}(\hat{x}) = H_{ss(pp)}(-\hat{x})$$

$$H_{sp}(\hat{x}) = -H_{sp}(-\hat{x})$$

Sample spectrum:

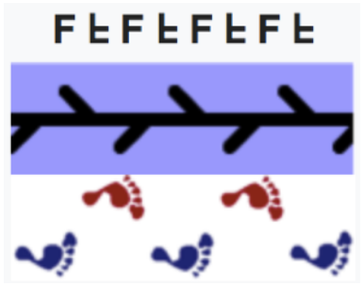


PythTB is based at: http://physics.rutgers.edu/pythtb/

Exercise 2: implement pg Hamiltonian and plot band structure

(Non-symmorphic) group generated by {m_y | ½ 0}

$$\mathbf{r}_1 = 0, \mathbf{r}_2 = 1/2$$



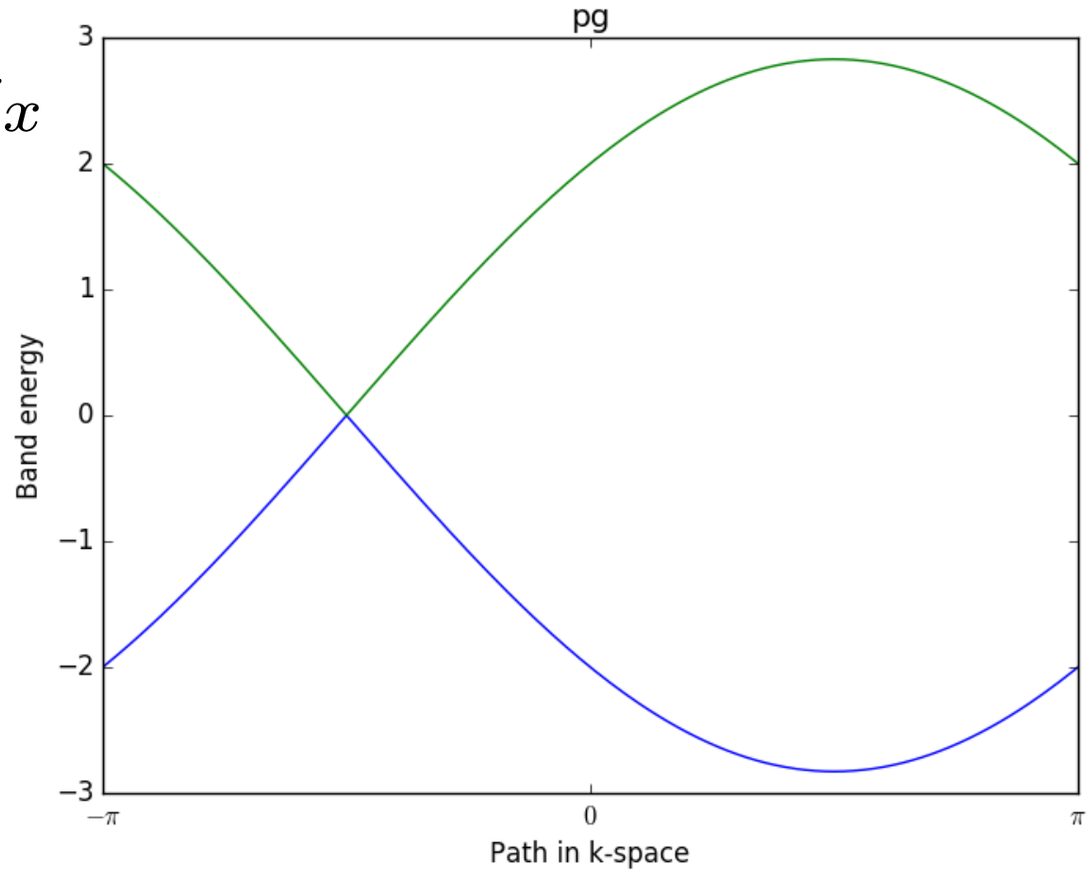
"Frieze" group

glide exchanges orbitals $U_g = \sigma_x$ $\sigma_x H^k \sigma_x = H^k$

$$H^k = \mu\mathbb{I} + (t\cos\frac{k}{2} + t'\sin\frac{k}{2})\sigma_x$$

- both sites have same chemical potential
- two types of nearest neighbor hopping

Sample spectrum: t = t' = 1



PythTB is based at: http://physics.rutgers.edu/pythtb/