Appendix A
The PythTB package

Introduction
Numerous examples and exercises in this book make use of the PythTB software package. This package, written in the Python programming language, is designed to allow the user to construct and solve tight binding (TB) models of the electronic structure of finite clusters and of systems that display periodicity in one or more dimensions, such as polymer chains, ribbons, slabs, and 3D crystals. It is also designed to provide convenient features for computing geometric or topological properties such as Berry phases, Berry curvatures, and Chern numbers.

The examples and exercises used in this book assume the use of Version 1.6.2 of PythTB. At the time of this writing this is available at http://www.physics.rutgers.edu/pythtb. Instructions for downloading and installing the package can be found there. The example files contained in this Appendix can be downloaded from http://www.physics.rutgers.edu/grad/682/pythtb.

PythTB extensions module
The examples that follow often make use of a few auxiliary functions for printing and plotting results, contained in the module file ptbe.py (“PythTB extensions”) reproduced below. To use these functions, this file should be present in the working directory (or centrally installed), and the calling program should contain the line import ptbe.

ptbe.py

# ptbe.py
***
A collection of functions that serve to extend PythTB capabilities
def print_eig_real(eval, evec):
    """
    Prints eigenvalues and real parts of eigenvectors, one to a line.
    Should not be used if eigenvectors are complex. Also, there is
    no line wrapping, so this should not be used for long eigenvectors.
    """
    n = len(eval)
    evecr = evec.real
    print " n  eigval  eigvec"
    for i in range(n):
        print " %2i %7.3f" % (i, eval[i]),
        print " (%s" % (", ".join("%6.2f" % x for x in evecr[i,:]) + " )")

Example PythTB programs

h2o.py

#!/usr/bin/env python

# import the pythtb module
from pythtb import *

# geometry: bond length and half bond-angle
b = 1.0; angle = 54.0 * np.pi / 180

# site energies [O(s), O(p), H(s)]
eos = -1.5; eop = -1.2; eh = -1.0

# hoppings [O(s)-H(s), O(p)-H(s)]
ts = -0.4; tp = -0.3

# define frame for defining vectors: 3D Cartesian
lat = [[1.0,0.0,0.0], [0.0,1.0,0.0], [0.0,0.0,1.0]]

# define coordinates of orbitals: O(s,px,py,pz) ; H(s) ; H(s)
orb = [ [0.,0.,0.], [0.,0.,0.], [0.,0.,0.], [0.,0.,0.],
     [b*np.cos(angle), b*np.sin(angle),0.],
     [b*np.cos(angle),-b*np.sin(angle),0.]]

my_model = tbmodel(0, 3, lat, orb)
my_model.set_onsite([eos, eop, eop, eop, eh, eh])
my_model.set_hop(ts, 0, 4)
my_model.set_hop(ts, 0, 5)
my_model.set_hop(tp * np.cos(angle), 1, 4)
my_model.set_hop(tp * np.cos(angle), 1, 5)
Example PythTB programs

my_model.set_hop(tp*np.sin(angle),2,4)
my_model.set_hop(-tp*np.sin(angle),2,5)
my_model.display()

(eval,evec)=my_model.solve_all(eig_vectors=True)
#
# # signs of evec's are regularized by the following rule
# (this is arbitrary and not very important)
for i in range(len(eval)):
    if sum(evec.real[i,1:4]) < 0:
        evec[i,:]=-evec[i,:]
ptbe.print_eig_real(eval,evec)

benzene.py

#!/usr/bin/env python
#
# import the pythtb module
from pythtb import *
#
# define frame for defining vectors: 2D Cartesian
lat=[[1.0,0.0],[0.0,1.0]]

# define coordinates of orbitals:
orb=np.zeros((6,2),dtype=float)
for i in range(6):
    angle=i*np.pi/3.0
    orb[i,:]= [r*np.cos(angle), r*np.sin(angle)]
my_model=tbmodel(0,2,lat,orb)
my_model.set_onsite([ep,ep,ep,ep,ep,ep])
my_model.set_hop(t,0,1)
my_model.set_hop(t,1,2)
my_model.set_hop(t,2,3)
my_model.set_hop(t,3,4)
my_model.set_hop(t,4,5)
my_model.set_hop(t,5,0)
my_model.display()
(eval, evec) = my_model.solve_all(eig_vectors=True)
ptbe.print_eig_real(eval, evec)