Supplementary Information: Moiré patterns generated by stacked 2D lattices: a general algorithm to identify primitive coincidence cells

V. Carnevali[1,†], S. Marcantoni[1,2,‡], M. Peressi[1]

April 7, 2021

1. Department of Physics, University of Trieste, via A. Valerio 2, 34127 Trieste, Italy

2. National Institute for Nuclear Physics (INFN), Trieste Section, via A. Valerio 2, 34127 Trieste, Italy

† Presently at: Department of Physics, Central Michigan University, Mt. Pleasant, MI 48859, US

‡ Presently at: School of Physics & Astronomy, University of Nottingham, Nottingham NG7 2RD, UK

*corresponding author: Virginia Carnevali (carne1v@cmich.edu)

Hexagonal/square stacked lattices: example of serial code

For the sake of readability and simplicity, we report the simplest serial Python version of the code, specifically written for the case study of hexagonal/square stacked lattices.

The code can be easily parallelized on the (i, j, k, l, m, n, q, r) indexes. This operation will help the user in speeding up the code considerably. To do that, we suggest to translate the code in C++ or Fortran90 and use the OpenMP library.

README

```
# Three values from command line
# 1st --> misorientation angle between overlayer and substrate Bravais lattices (degree)
# 2nd --> R: semi-amplitude of the range of variation for (i,j,k,l,m,n,q,r)
# 3rd --> t: tolerance

# HOW TO RUN THE CODE IN SERIAL
python MoireEq_serial.py 48 4 0.04
```

Functions

```
from math import tan, sqrt, cos, sin
import csv
import numpy as np

# Check if the area of the supercell is larger then the areas of the
# primitive cells of the stacked Bravais lattices - this excludes zero area case
# check if a,b,c,d computed through the integer values (i,j,k,l,m,n,q,r) are close
# to the exact values within the tolerance
def validity_check(t,As,Ao,a,b,c,d,M):
    am=[a-t,a+t]
    bm=[b-t,b+t]
    cm=[c-t,c+t]
    dm=[d-t,d+t]
    if((M[4] > As) and (M[4] >= Ao*abs(M[5]*M[8]-M[6]*M[7]))):
        if(am[0] < M[0] < am[1] and bm[0] < M[1] < bm[1] and cm[0] < M[2] < cm[1] and dm[0] < M[3] < dm[1]):
            return 'True'
    else:
        return 'False'

```

```
# compute the parameters a,b,c,d and the cell area given a set of (i,j,k,l,m,n,q,r)
def area_cell(As,i,j,k,l,m,n,q,r):
    dd = (i*l-j*k)
    if dd!=0:
        a = float(l*m-j*q)/float(dd)
        b = float(l*n-j*r)/float(dd)
        c = float(-k*m+i*q)/float(dd)
        d = float(-k*n+i*r)/float(dd)
        Acell = abs(m*r-n*q)*As
        matr = [a,b,c,d,Acell,i,j,k,l,m,n,q,r]
    else:
        matr=[0.0]*13
    return matr

# write a list on a .csv file
def writers_csv(filename,list):
    with open(filename, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerows(list)
```

Main program

```
import itertools
import numpy as np
from math import tan, sqrt, cos, sin
import sys
import csv
import MoireEq_functions

#============= SETTING THE BRAVAIS LATTICES ================

# angle: input in deg, conversion in rad
phi = float(sys.argv[1])*0.0174533
# range indexes: semi-amplitude
mm = int(sys.argv[2])
# tolerance
t = float(sys.argv[3])

# 1st primitive vector of the substrate square Bravais lattice: Angstrom
as1 = 2.49
# 2nd primitive vector of the substrate square Bravais lattice: Angstrom
as2 = 2.49
# 1st primitive vector of the overlayer Hexagonal Bravais
```

```
   lattice: Angstrom
ao1 = 2.46
# 2nd primitive vector of the overlayer Hexagonal Bravais
   lattice: Angstrom
ao2 = 2.46
# rescaling parameter between the 1st bases vectors
p1=ao1/as1
# rescaling parameter between the 2nd bases vectors
p2=ao2/as2
# area of the primitive cell of the square Bravais lattice
As = as1*as2
# area of the primitive cell of the hexagonal Bravais lattice
Ao = sqrt(3.0)/2.0*ao1*ao2
# initialization for the solutions array
M = []

# calculation of the exact a,b,c,d
a=p1*cos(phi)
b=p1*sin(phi)
c=-sqrt(3.0)/2.0*b-1/2.0*a
d=sqrt(3.0)/2.0*a-1/2.0*b

#======================== MAIN ========================

# setting the range for (i,j,k,l,m,n,q,r)
ranges = [np.arange(-mm,mm+1,1)]*8

for i,j,k,l,m,n,q,r in itertools.product(*ranges):

    # compute the parameters a,b,c,d and the cell area given
   a set of (i,j,k,l,m,n,q,r)
    matr = MoireEq_functions.area_cell(As,i,j,k,l,m,n,q,r)

    # check if the volume of the supercell is larger then the
    volumes of the
    # primitive cells of the stacked Bravais lattices
    check = MoireEq_functions.validity_check(t,As,Ao,a,b,c,d,
   matr)
    if (check=='True'):
        M.append(matr)

#ordering the solutions according to the area
M.sort(key=lambda row: row[4])

# save the found solutions in a .csv file
```

```
62 # format [a,b,c,d,Acell,i,k,j,l,m,n,q,r]
63 if(len(M)!=0):
64     MoireEq_functions.writers_csv('solutions_'+str(mm)+'_'+
   str(t)+'_serial.csv',M)
```