Electronic Supplementary Material (ESI) for Physical Chemistry Chemical Physics.

Supplementary Information

In pursuit of accurate interlayer potentials for twisted bilayer graphynes

Ajay Melekamburath, Anto James, Megha Rajeevan, Chris John and Rotti Srinivasamurthy Swathi*

School of Chemistry, Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM), Vithura, Thiruvananthapuram, 695551, India.

E-mail: swathi@iisertvm.ac.in

The Effect of Peripheral Hydrogen Atoms in the Molecular Models on Interlayer Interaction Energies

In this section, we analyze the contribution of peripheral hydrogen atoms in the model systems towards interlayer interaction energies of the bilayer systems. For this, we compare the potential energy surfaces, evaluated with and without the contribution of hydrogen atoms. First, we illustrate the case of coronene and benzene model systems of graphene using Hod's interlayer potential (H-ILP). Here, the parameters are taken from the previous study on bilayer graphene by Hod and co-workers[1]. A comparison of the potential energy profiles computed using H-ILP for the benzene-coronene and coronene-coronene systems with and without including the contribution from the peripheral hydrogen atoms is shown in Figure S1.

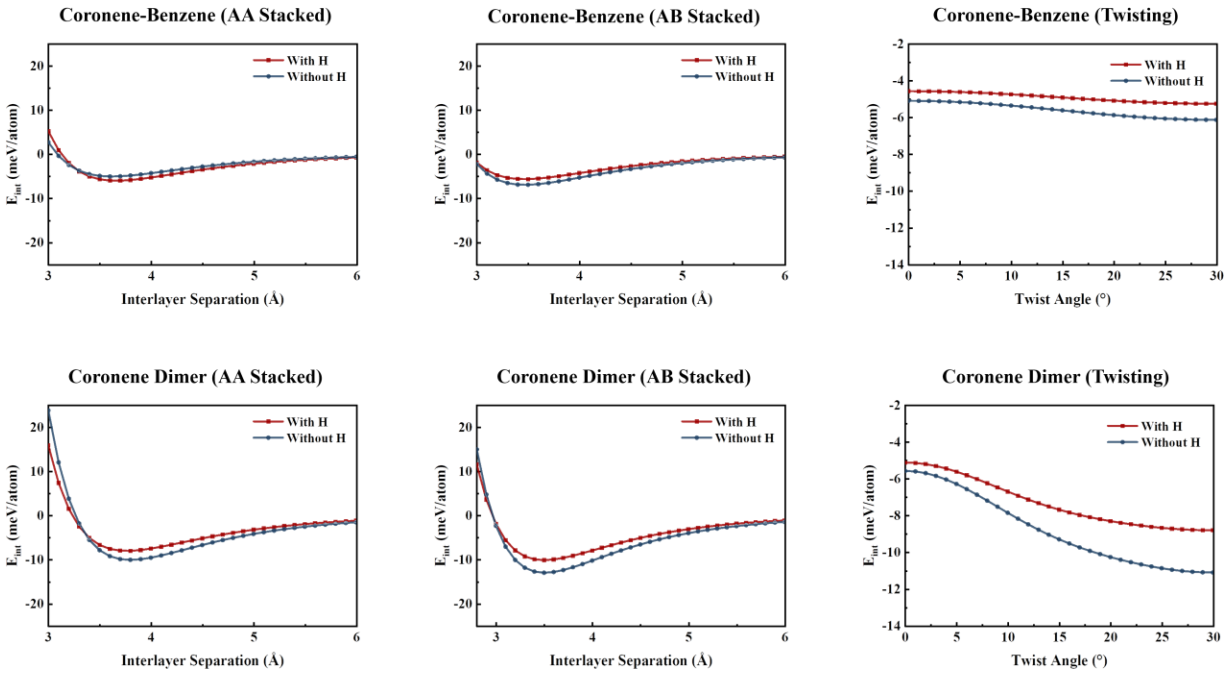

Figure S1: A comparison of the potential energy profiles computed using H-ILP for the benzene-coronene and coronene-coronene systems with and without including the contribution from the peripheral hydrogen atoms.

We have carried out a similar analysis for the improved Lennard-Jones (ILJ) potential using the parameters developed by Bartolomei and co-workers[2], and similiar effect can be seen in these potential energy surfaces too (see Figure S2).

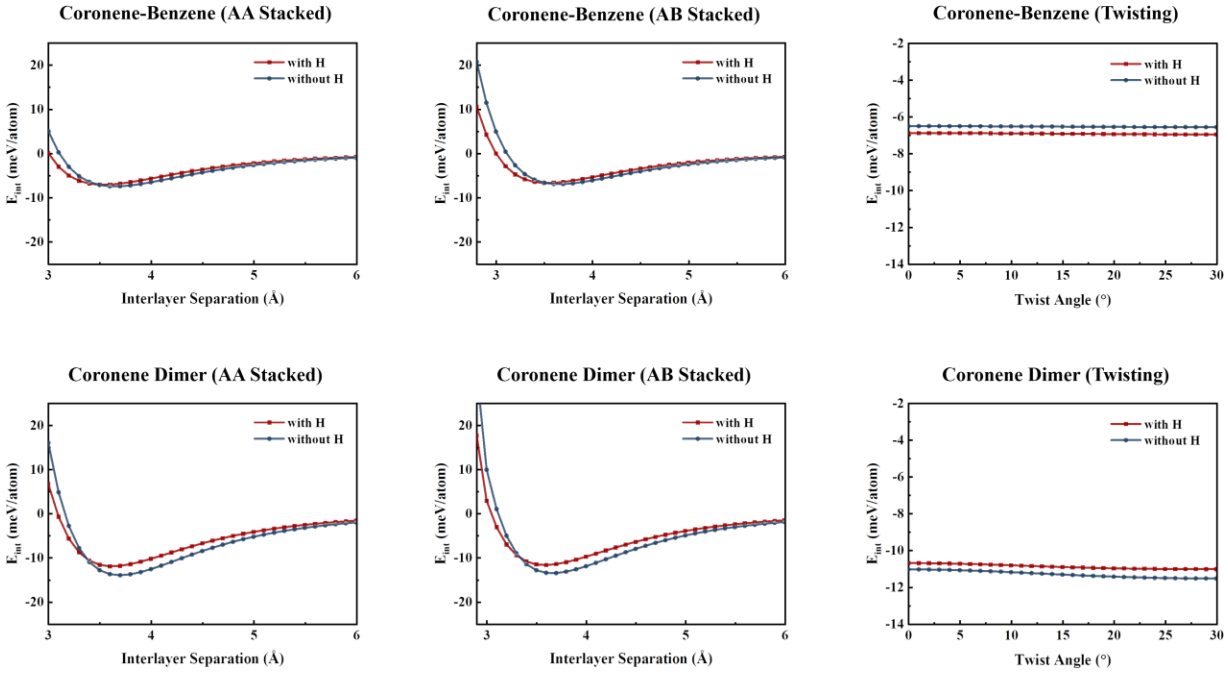

Figure S2: A comparison of the potential energy profiles computed using ILJ potential for the benzene-coronene and coronene-coronene systems with and without including the contribution from the peripheral hydrogen atoms.

As seen in the Figures S1 and S2, the exclusion of hydrogen atoms from the model systems causes only a small shift in the interaction energy values. The root-mean-square (RMS) deviations between these curves are tabulated in Table S1.

Table S1: RMS deviations of potential energy curves of dimers of graphene-based model systems computed using H-ILP and ILJ potential.

System	RMS Deviation (meV/atom)	
	H-ILP	ILJ
Coronene-Benzene (AA Stacked)	0.7922	1.2443
Coronene-Benzene (AB Stacked)	0.7786	2.4888

Coronene-Benzene (Twist)	0.7124	0.3880
Coronene Dimer (AA Stacked)	2.1057	2.4560
Coronene Dimer (AB Stacked)	1.7635	1.9928
Coronene Dimer (Twist)	1.6246	0.4223

Further, we perform a similar analysis with the GY molecular models, by coupling the parameters for C-C pairwise interactions proposed in this study, with C-H and H-H interaction parameters for graphitic hydrogens from literature[1, 2]. Comparisons of the potential energy profiles computed using H-ILP and ILJ potential for bilayers of GYs with and without including the contribution from the peripheral hydrogen atoms are shown in Figures S3 and S4, respectively.

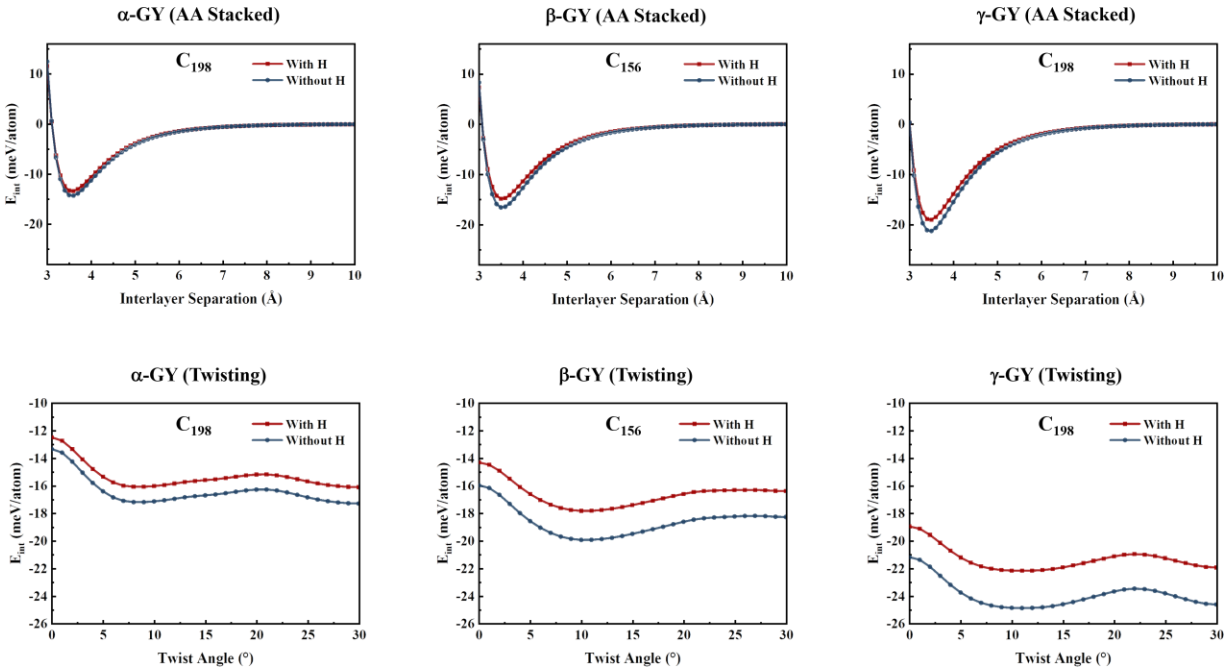

Figure S3: A comparison of the potential energy profiles computed using H-ILP for the stacking and twisting of bilayers of GYs evaluated with and without including the contribution from the peripheral hydrogen atoms.

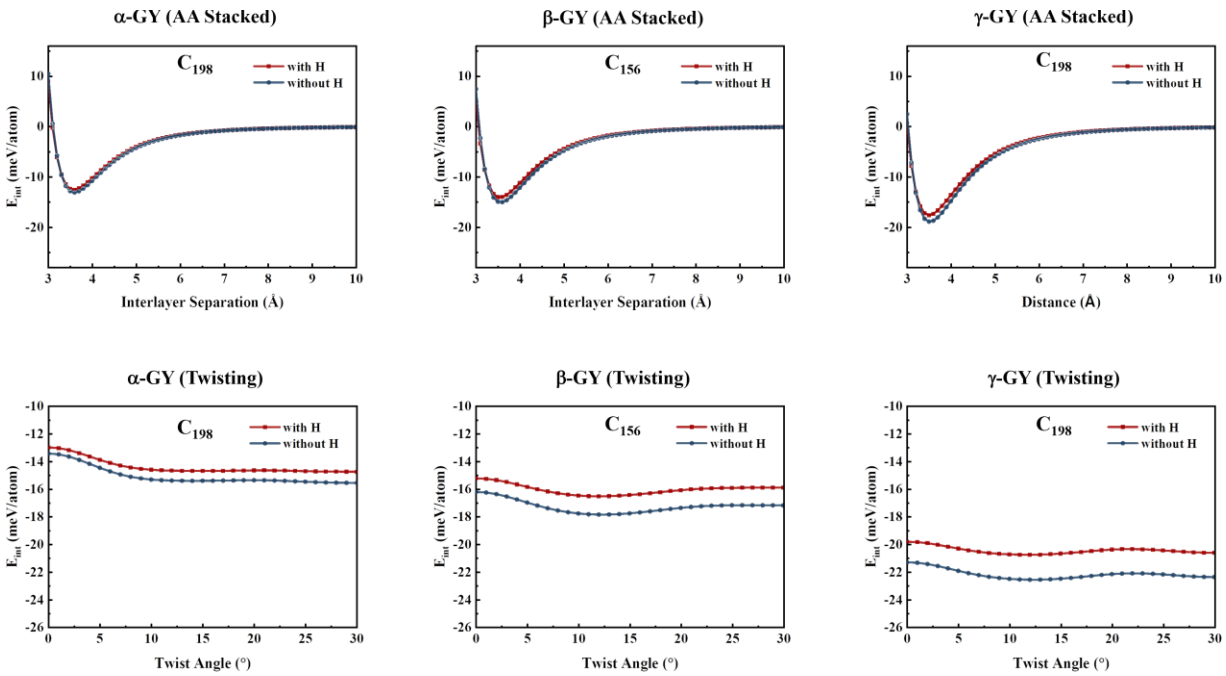

Figure S4: A comparison of the potential energy profiles computed using ILJ potential for the stacking and twisting of bilayers of GYs evaluated with and without including the contribution from the peripheral hydrogen atoms. ILJ-I parameters are used for vertical displacement profiles and ILJ-II is used for twisting profiles.

From the analysis of RMS deviations between the curves for various GY systems (see Table S2), it can be concluded that the hydrogens only scale the interaction energies by a small factor. This difference is due to the repulsion brought in by hydrogen atoms. The exclusion of hydrogen atoms in the model systems does not alter the nature of the potential energy surfaces. While computing the benchmark DFT potential energy curves, the contribution from peripheral hydrogen atoms are considered. However, since we did not consider C-H and H-H pairwise interactions in the parametrization, the set of C-C interaction parameters provided in Table 1 implicitly account for the contribution due to peripheral hydrogen atoms. For explicit analyses including peripheral hydrogens atoms, the pairwise interaction parameters developed for graphitic hydrogens are a good approximation, as they do not bring much deviations. On the contrary, incorporation of the parameters for the hydrogen into the fitting procedure may not be the ideal thing to do because too many parameters can cause overfitting.

Table S2: RMS deviations of potential energy curves of GY model systems computed using H-ILP and ILJ potentials

System	RMS Deviation (meV/atom)	
	H-ILP	**ILJ**
α-GY (AA Stacked)	0.3532	0.3089
α-GY (Twist)	1.0905	0.6870
β-GY (AA Stacked)	0.6442	0.5130
β-GY (Twist)	1.9766	1.2417
γ-GY (AA Stacked)	0.8251	0.5849
γ-GY (Twist)	2.5818	1.7253

Python Codes for Generating Potential Energy Surfaces

Points to note:

- The codes are written in Python 3 and require NumPy library to be installed.
- The codes are designed to deal with planar molecular models confined to the *xy*-plane.
- For vertical displacement profiles, the interlayer separation is varied along the *z*-axis.
- For twisting profiles, the rotation is about the *z*-axis.

1. For evaluating vertical displacement profiles,

```
# The code uses Python 3 and NumPy library should be installed for the execution.

# Note that the coordinates have to be provided in a comma-separated format.

# Article title: In pursuit of accurate interlayer potentials for twisted bilayer graphynes

# Authors: Ajay Melekamburath, Anto James, Megha Rajeevan, Chris John and Rotti Srinivasamurthy Swathi

# Please cite the article while using the code.


input_file=input("File containing sheet coordinates: ")


import numpy as np #Importing relevant libraries

import math


atom=np.loadtxt(input_file,delimiter=',',usecols=(0),dtype=str,unpack=True) #Reading atom symbols

cord=np.loadtxt(input_file,delimiter=',',usecols=(1,2,3),dtype=float,unpack=True) #Reading coordinates

natoms=int(sum(1 for line in open(input_file))) #Counting number of lines/atoms


#ILJ Potential parameters [beta, rm (Å), epsilon (kcal/mol)]

param_CC_ILJ = [7.935, 3.872, 0.087] # Parameters-ILJ-I

#param_CC_ILJ = [6.404, 3.939, 0.095] # Parameters-ILJ-II


r_cut = 16.0 # Defining cut-off distance for H-ILP in Å
```

```
n = [0,0,1] #Normal vector of sheet, taken as unit vector in the z-direction


#H-ILP parameters [alpha_ij, beta_ij (Å), gamma_ij (Å), epsilon_ij (kcal/mol), C_ij (kcal/mol), reff_ij (Å), C_6, ij (kcal Å6/mol)]

param_CC_ILP = [9.95, 2.79, 1.40, 0.178, 2.40, 3.513, 598.99] # Parameters - H-ILP

#d_ij and S_R,ij are fixed at 15.0 and 1.0, respectively.

#----------ILJ energy evaluation function----------#

def ILJ_Eval():

    f = open("ILJ-PESD-%s.txt" %(input_file), "w") #Opening file to write output

    beta = param_CC_ILJ[0] #Defining ILJ parameters

    rm = param_CC_ILJ[1]

    eps = param_CC_ILJ[2]

    for a in range(71): #Varying interlayer separation from 10 Å to 3 Å with a step size of 0.1 Å

        ild = float((100-a)/10)

        VILJ =0.0 ;

        for i in range(natoms):

            X1=np.array((cord[0][i],cord[1][i],cord[2][i]))

            for j in range(natoms):

                X2=np.array((cord[0][j],cord[1][j],cord[2][j]+ild)) #Adding the interlayer separation

                r=np.linalg.norm(np.subtract(X1,X2)) #Calculating interatomic distance

                rred=(r/rm)

                nr=beta+(4*np.square(rred))

                VkILJ=eps*(((6.0/(nr-6.0))*((1.0/rred)**nr))-((nr/(nr-6.0))*((1.0/rred)**6.0)))
*43.36/(2*natoms) #Evaluating pairwise interaction

                VILJ += VkILJ # Adding up pairwise interactions

        print("%10.6f \t %12.9f\n" %(ild,VILJ))

        f.write("%10.6f \t %12.9f\n" %(ild,VILJ))

    f.close()

#----------ILJ energy evaluation function----------#
```

```
#.....H-ILP energy evaluation function.....#
def ILP_fn():
    f = open("ILP-PESD-%s.txt" %(input_file), "w") #Opening file to write output
    for a in range(71):
        ild = float((100-a)/10) #Varying interlayer separation from 10 Å to 3 Å with a step size of 0.1 Å
        V_ILP = 0.0
        f = open("ILP-PESD-%s.txt" %(input_file), "w") #Opening file to write output
        for i in range(natoms):
            X1=np.array((cord[0][i],cord[1][i],cord[2][i]))
            for j in range(natoms):
                X2=np.array((cord[0][j],cord[1][j],cord[2][j]+ild))
                r_vec = np.subtract(X1,X2) #Resultant vector between atom pairs
                r_ij = np.linalg.norm(r_vec) #Calculating interatomic distance


                rho = np.sqrt(r_ij**2 - np.square(np.dot(r_vec, n))) #Lateral distance between interacting atoms
                rd = (r_ij/r_cut) #Reduced distance with respect to cut-off distance
                taper = 20*rd**7 - 70*rd**6 + 84*rd**5 - 35*rd**4 +1 #Taper Function


                V_attr = -(param_CC_ILP[6]/r_ij**6)*(1 + math.exp(-15.0*(r_ij/(param_CC_ILP[5])) -1))**(-1) #Evaluating attractive part of the potential
                V_rep = math.exp(param_CC_ILP[0]*(1 - (r_ij/param_CC_ILP[1]))) * (param_CC_ILP[3] + param_CC_ILP[4]*(math.exp(-(rho/param_CC_ILP[2])**2) + math.exp(-(rho/param_CC_ILP[2])**2))) #Evaluating repulsive part of the potential
                V_ILP += taper*(V_attr + V_rep)*43.36/(2*natoms) #Total energy multiplied with taper function and converted to meV/atom
        f.write("%10.6f \t %12.9f \n" %(ild, V_ILP))
        print("%10.6f \t %12.9f \n" %(ild, V_ILP))
    f.close()
#.....-.....-#
```

```
calc = input("Empirical potential to use? (Choose either ILJ or H-ILP): ")
print('\n')
print("***")
print("This code generates the potential energy surface using the " +calc+ " for bilayer systems.")
print('\n')
print("Starting...")
print("\n")



if calc == "ILJ":
    ILJ_Eval()
    print("\n")
    print("***")
    print("Completed!")
    print("Output is written to ILJ-PESD-%s.txt" %(input_file))
elif calc == "H-ILP":
    ILP_fn()
    print("\n")
    print("***")
    print("Completed!")
    print("Output is written to ILP-PESD-%s.txt" %(input_file))
else:
    print("Choose either ILJ or H-ILP: ")
```

2. For evaluating twisting energy profiles,

```
# The code uses Python 3 and NumPy library should be installed for the execution.

# Note that the coordinates have to be provided in a comma-separated format.

# Article title: In pursuit of accurate interlayer potentials for twisted bilayer graphynes

# Authors: Ajay Melekamburath, Anto James, Megha Rajeevan, Chris John and Rotti Srinivasamurthy Swathi

# Please cite the article while using the code.


input_file=input("File containing sheet coordinates: ")


import numpy as np #Importing relevant libraries

import math


atom=np.loadtxt(input_file,delimiter=',',usecols=(0),dtype=str,unpack=True) #Reading atom symbols

cord=np.loadtxt(input_file,delimiter=',',usecols=(1,2,3),dtype=float,unpack=True) #Reading coordinates

natoms=int(sum(1 for line in open(input_file))) #Counting number of lines/atoms


#ILJ Potential parameters [beta, rm (Å), epsilon (kcal/mol)]

#param_CC_ILJ = [7.935, 3.872, 0.087] # Parameters-ILJ-I

param_CC_ILJ = [6.404, 3.939, 0.095] # Parameters-ILJ-II


r_cut = 16.0 # Defining cut-off distance for H-ILP in Å

n = [0,0,1] #Normal vector of sheet, taken as unit vector in the z-direction


#H-ILP parameters [alpha_ij, beta_ij (Å), gamma_ij (Å), epsilon_ij (kcal/mol), C_ij (kcal/mol), reff_ij (Å), C_6, ij (kcal Å^6/mol)]

param_CC_ILP = [9.95, 2.79, 1.40, 0.178, 2.40, 3.513, 598.99] # Parameters - H-ILP

#d_ij and S_R,ij are fixed at 15.0 and 1.0, respectively.

#.....ILJ energy evaluation function-.....#

def ILJ_Eval():
```

```
    f = open("ILJ-PESA-%s.txt" %(input_file), "w") #Opening file to write output
    beta = param_CC_ILJ[0] #Defining ILJ parameters
    rm = param_CC_ILJ[1]
    eps = param_CC_ILJ[2]
    for a in range(0, 31): #0° to 30° twist angles with a step size of 1°
        angle = a*np.pi/180 #Converting degrees to radian
        ild = 3.4 #Fixed interlayer separation at 3.4 Å
        VILJ =0.0 ;
        for i in range(natoms):
            X1=np.array((cord[0][i], cord[1][i], cord[2][i]))
           for j in range(natoms):
              X2=np.array((cord[0][j]*np.cos(angle)-                cord[1][j]*np.sin(angle),
cord[1][j]*np.cos(angle)+cord[0][j]*np.sin(angle), cord[2][j]+ild)) #Multiplying with rotation matrix
              r=np.linalg.norm(np.subtract(X1,X2)) #Calculating interatomic distance


              rred=(r/rm)
              nr=beta+(4*np.square(rred))
              VkILJ=eps*(((6.0/(nr-6.0))*((1.0/rred)**nr))-((nr/(nr-
6.0))*((1.0/rred)**6.0)))*43.36/(2*natoms) #Evaluating pairwise interaction
              VILJ += VkILJ #Adding up pairwise interactions
        print("%10.6f \t %12.9f\n" %(a ,VILJ))
        f.write("%10.6f \t %12.9f\n" %(a ,VILJ))
    f.close()
#                                    -          -          -#


#                    H-ILP energy evaluation function                    #
def ILP_fn():
    f = open("ILP-PESA-%s.txt" %(input_file), "w") #Opening file to write output
    for a in range(0, 31): #0° to 30° twist angles with a step size of 1°
```

```
    ild = 3.4 #Fixed interlayer separation
    V_ILP = 0.0
    angle = (a*np.pi/180) #Converting degrees to radians
    for i in range(natoms):
        X1=np.array((cord[0][i],cord[1][i],cord[2][i]+ild))
        for j in range(natoms):
            X2=np.array((cord[0][j]*np.cos(angle) - cord[1][j]*np.sin(angle), cord[1][j]*np.cos(angle) + cord[0][j]*np.sin(angle), cord[2][j])) #Multiplying with rotation matrix
            r_vec = np.subtract(X1,X2) #Resultant vector between atom pairs
            r_ij = np.linalg.norm(r_vec) #Calculating interatomic distance

            rho = np.sqrt(r_ij**2 - np.square(np.dot(r_vec, n))) #Lateral distance between atom pairs
            rd = (r_ij/r_cut) #Reduced distance with respect to cut-off distance
            taper = 20*rd**7 - 70*rd**6 + 84*rd**5 - 35*rd**4 +1 #Taper Function

            V_attr = -(param_CC_ILP[6]/r_ij**6)*(1 + math.exp(-15.0*(r_ij/(param_CC_ILP[5])) -1))**(-1) #Evaluating attractive part of the potential
            V_rep = math.exp(param_CC_ILP[0]*(1 - (r_ij/param_CC_ILP[1]))) * (param_CC_ILP[3] + param_CC_ILP[4]*(math.exp(-(rho/param_CC_ILP[2])**2) + math.exp(-(rho/param_CC_ILP[2])**2))) #Evaluating repulsive part of the potential
            V_ILP += taper*(V_attr + V_rep)*43.36/(2*natoms) #Total energy multiplied with taper function and converted to meV/atom
        f.write("%10.6f \t %12.9f \n" %(a, V_ILP))
        print("%10.6f \t %12.9f \n" %(a, V_ILP))
    f.close()
#-----------------------------------------------------------------------------#


calc = input("Empirical potential to use? (Choose either ILJ or H-ILP): ")
print('\n')
```

```
print("***")
print("This code generates the potential energy surface using the " +calc+ " for bilayer systems.")
print('\n')
print("Starting...")
print("\n")


if calc == "ILJ":
    ILJ_Eval()
    print("\n")
    print("***")
    print("Completed!")
    print("Output is written to ILJ-PESA-%s.txt" %(input_file))
elif calc == "H-ILP":
    ILP_fn()
    print("\n")
    print("***")
    print("Completed!")
    print("Output is written to ILP-PESA-%s.txt" %(input_file))
else:
    print("Choose either ILJ or H-ILP: ")
```

References

1. T. Maaravi, I. Leven, I. Azuri, L. Kronik and O. Hod, *J. Phys. Chem. C*, 2017, **121**, 22826-22835.
2. M. Bartolomei, F. Pirani and J. M. C. Marques, *J. Phys. Chem. C*, 2017, **121**, 14330-14338.