analysis8

March 6, 2023

1 Potential of Monolayer Charge

1.1 Analysis of DFT results

1.2 Import packages

[1]:
```
! pip install ase
! pip install matplotlib --upgrade

from ase import Atoms
from ase.db import connect
from ase.io import write, read
from ase.data import atomic_numbers
from os import listdir
import os
from os.path import isfile, join
from sympy import *

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
import re
from sklearn.linear_model import LinearRegression
from matplotlib.ticker import MultipleLocator
from collections import defaultdict
```

```
Requirement already satisfied: ase in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (3.22.1)
Requirement already satisfied: scipy>=1.1.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from ase) (1.7.3)
Requirement already satisfied: matplotlib>=3.1.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from ase) (3.5.1)
Requirement already satisfied: numpy>=1.15.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from ase) (1.21.4)
Requirement already satisfied: fonttools>=4.22.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (4.38.0)
Requirement already satisfied: packaging>=20.0 in
```

```
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (21.3)
Requirement already satisfied: cycler>=0.10 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from
matplotlib>=3.1.0->ase) (3.0.9)
Requirement already satisfied: six>=1.5 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from python-
dateutil>=2.7->matplotlib>=3.1.0->ase) (1.16.0)
Requirement already satisfied: matplotlib in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (3.5.1)
Collecting matplotlib
  Downloading
matplotlib-3.7.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6
MB)
                            11.6/11.6 MB
3.5 MB/s eta 0:00:00 0:01 0:01
Collecting contourpy>=1.0.1
  Using cached
contourpy-1.0.7-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (299
kB)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(1.4.4)
Requirement already satisfied: cycler>=0.10 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(4.38.0)
Requirement already satisfied: packaging>=20.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(21.3)
Requirement already satisfied: pillow>=6.2.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(9.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
```

```
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: numpy>=1.20 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(1.21.4)
Requirement already satisfied: importlib-resources>=3.2.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(5.10.0)
Requirement already satisfied: python-dateutil>=2.7 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: zipp>=3.1.0 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from importlib-
resources>=3.2.0->matplotlib) (3.10.0)
Requirement already satisfied: six>=1.5 in
/home/vlad/anaconda3/envs/dft/lib/python3.9/site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
Installing collected packages: contourpy, matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.5.1
    Uninstalling matplotlib-3.5.1:
      Successfully uninstalled matplotlib-3.5.1
Successfully installed contourpy-1.0.7 matplotlib-3.7.1
```

1.3 Constants

[2]:
```
k = 1.9872041E-3  # kcal/mol/K
rho = 0.03332819694513222
el_chg = 1.60217662e-19 #coulomb
Na = 6.02214129e23
epsilon0 =  8.854187817e-12  #F/m
eps0 = 0.0055263494 # el/V/A
kcal_eV = 2.611448e22  # kcal/(eV)
bohr2a = 0.529177 #A/bohr
D2Cm =  3.33564*10**-30 #C*m
au2D = 2.5417 #D
debye2eAngs = 0.2081943# e*Angs
A2nm=0.1
hartree2V = 27.211386245988
au2ev = 27.211384500
```

1.4 Functions

[3]:
```
def getSurfaceCoverage(area):
  """
  Turn molecule area in Angs2 to surface coverage mol/cm2
  """
```

```
    return (1/(area*10**-16))/Na
```

[4]:
```
def potential_at_a(a, z_array, rho_z):
    """Returns poteintial at point a, given a 1D-chg distribution (rho_z)
    and points at which it was evaluated (z_array) """
    to_integrate = z_array < a
    dz = z_array[1] - z_array[0]
    integrand = 0
    for z_i, rho_zi in zip(z_array[to_integrate], rho_z[to_integrate]):
        integrand += (a - z_i)*rho_zi
    return -1/eps0*integrand*dz
```

[5]:
```
"""
Class to read in data of one surface that corresponds to some optimal␣
 ↪distribution of particles on the electrode surface
"""
class SurfaceED:
    def __init__(self, dataRow, calculationType="PW", shift=0):

        self.calculationType = calculationType
        #Additions to select right columns
        if calculationType == "LCAO":
          _calcType=["lcao_","lcao_"] #First is how separate components were␣
 ↪calculated, second is how the interface was calculated
        elif calculationType == "cDFT":
          _calcType = ["lcao_","cdft_"]
        else:
          _calcType = ["",""]

        self.sysName = str(dataRow["electrode"])+"_"+str(dataRow["ads_name"])
        shift = shift #Angs shifting charge density profiles to not have␣
 ↪discontinuity

        #First get substrate+electrode data
        self.electrodeArea = float(dataRow["area"])

        self.z_Ed=np.array(dataRow["sys_"+_calcType[1]+"den_zax"][1:]).
 ↪astype(float)
        self.dz_Ed = float(self.z_Ed[2]-self.z_Ed[1]) #Angstroms
        shiftIndex = int(shift//self.dz_Ed)
        self.sysEd = np.array(dataRow["sys_"+_calcType[1]+"den_val"][1:]).
 ↪astype(float)
        self.sysEd = np.r_[self.sysEd[-shiftIndex:], self.sysEd[:-shiftIndex]]

        #Get substrate data
```

```
        self.adsEd = np.array(dataRow["ads_"+_calcType[0]+"den_val"][1:]).
↪astype(float)
        self.adsEd = np.r_[self.adsEd[-shiftIndex:], self.adsEd[:-shiftIndex]]

        #Get electrode data
        self.metEd = np.array(dataRow["met_"+_calcType[0]+"den_val"][1:]).
↪astype(float)
        self.metEd = np.r_[self.metEd[-shiftIndex:], self.metEd[:-shiftIndex]]
        #Calculate charge density change
        self.chgChange = -(self.sysEd - self.adsEd - self.metEd)

        #Calculate potential change
        self.potentialProfileEdDiff = [potential_at_a(a, self.z_Ed, self.
↪chgChange) for a in self.z_Ed] #V
        self.potDropEdDiff = self.potentialProfileEdDiff[0]-self.
↪potentialProfileEdDiff[-1]
        self.potentialProfileHartree = np.
↪array(dataRow["sys_"+_calcType[1]+"pot_val"][1:]).astype(float) #V
        self.potDropHartree = -(self.potentialProfileHartree[int(len(self.
↪z_Ed)*0.06)]-self.potentialProfileHartree[int(len(self.z_Ed)*0.95)])
        self.z_Hartree = np.array(dataRow["sys_"+_calcType[1]+"pot_zax"][1:]).
↪astype(float)

        #Get data related to electrode positions and molecule positions
        metalIndexes = np.asarray(dataRow.numbers)==atomic_numbers[dataRow.
↪electrode]
        moleculeIndexes = np.asarray(dataRow.numbers)!=atomic_numbers[dataRow.
↪electrode]
        self.metalSheetPositions = np.unique(dataRow.positions[:
↪,2][metalIndexes].round(decimals=4))
        self.moleculeFirstPosition = np.min(dataRow.positions[:
↪,2][moleculeIndexes])
        self.moleculeMeanPosition = np.mean(dataRow.positions[:
↪,2][moleculeIndexes])

        #Identifying last layer
        roundedMetalSheetPosition = self.metalSheetPositions.round(0)
        lastLayerRounded = np.unique(roundedMetalSheetPosition)[-1]
        positionsOfLastLayer = self.
↪metalSheetPositions[roundedMetalSheetPosition==lastLayerRounded]
        self.metalSheetLastLayerAveragePosition = np.mean(positionsOfLastLayer)
        self.distanceBetweenElectrodeLayers = self.metalSheetPositions[1]-self.
↪metalSheetPositions[0]
        self.distanceBetweenMetalMoleculeGeometric = self.moleculeMeanPosition␣
↪- self.metalSheetLastLayerAveragePosition - self.
↪distanceBetweenElectrodeLayers/2
```

```
        #Estimate different positions from charge density
        self.metalChargeMaximaPosition = self.estimateMetalChargeMaxima()
        self.metalMoleculeBoundary = self.estimateMetalMoleculeBoundary()

        #Estimate the electrode and molecule charge density
        self.metalCharge = np.sum(self.chgChange[self.z_Ed<self.
↪metalMoleculeBoundary])*self.dz_Ed #e/Å^2
        self.metalCharge = self.metalCharge*el_chg*(10**6)/(10**-16) # C/cm²
        self.moleculeCharge = np.sum(self.chgChange[self.z_Ed>self.
↪metalMoleculeBoundary])*self.dz_Ed #e/Å^2
        self.moleculeCharge = self.moleculeCharge*el_chg*(10**6)/(10**-16) # C/
↪cm²

        self.moleculePosition = self.estimateMoleculeLocationFromChargeDensity()
        self.metalChargePlanePosition = self.estimateMetalChargePlane()
        self.distanceBetweenMetalMolecule = self.moleculePosition - self.
↪metalChargePlanePosition



    def estimateMetalChargeMaxima(self):
        #Estimate metal charge maxima plane
        #Limits are the position of last Electrode sheet and half the distance␣
↪to molecule position
        mask = np.nonzero((self.z_Ed>self.metalSheetPositions[-1]) & (self.
↪z_Ed<0.5*(self.moleculeFirstPosition+self.metalSheetPositions[-1])))[0]
        z_axis_fragment = self.z_Ed[mask]
        chgChange_fragment = self.chgChange[mask]
        #Take absolute value
        chgDensMax = np.argmax(abs(chgChange_fragment))
        metalChargeMaximaPosition = z_axis_fragment[chgDensMax]
        return metalChargeMaximaPosition

    def estimateMetalMoleculeBoundary(self):
        #Estimate the position, where the charge is 0 between metal and molecule
        #Limits are the position charge plane and the molecule position
        mask = np.nonzero((self.z_Ed>self.metalChargeMaximaPosition) & (self.
↪z_Ed<self.moleculeFirstPosition))[0]
        z_axis_fragment = self.z_Ed[mask]
        chgChange_fragment = self.chgChange[mask]
        #Take absolute value
        chgDensMax = np.argmin(abs(chgChange_fragment))
        metalMoleculeBoundary = z_axis_fragment[chgDensMax]
        return metalMoleculeBoundary
```

```
    def estimateMoleculeLocationFromChargeDensity(self):
        #Estimate the position of molecule by weighted average of charge density
        mask = self.z_Ed>self.metalMoleculeBoundary
        weights = np.abs(self.chgChange[mask])
        weights = weights/np.sum(weights)
        moleculePosition = np.sum(weights*self.z_Ed[mask])
        return moleculePosition

    def estimateMetalChargePlane(self):
        #Estimate the position of metal charge plane by weighted average of␣
↪charge density
        #First, when summing the charges from right to left, there must be a␣
↪point, where
        #overall charge is 0, this will be the lower limit of Metal Plane␣
↪position

        reverseChgChange = self.chgChange[::-1]
        reverseZ = self.z_Ed[::-1]
        #Check the charge you expect
        isMoleculeNegative = self.moleculeCharge < 0
        #Set molecule side always positive
        if isMoleculeNegative:
          reverseChgChange=-reverseChgChange

        cumulativeCharge = 0
        metalChargePlaneLowerLimit = 0
        for i in range(len(reverseZ)):
          cumulativeCharge+=reverseChgChange[i]
          if(reverseZ[i]<self.metalMoleculeBoundary and cumulativeCharge<0):
            metalChargePlaneLowerLimit = reverseZ[i]
            break

        #Now estimating the molecule charge plane by taking the weighted average
        #of chgDensity between found lower limit and metal molecule boundary
        mask = np.nonzero((self.z_Ed>metalChargePlaneLowerLimit) & (self.
↪z_Ed<self.metalMoleculeBoundary))[0]
        z_axis_fragment = self.z_Ed[mask]
        chgChange_fragment = self.chgChange[mask]
        weights = np.abs(chgChange_fragment)
        weights = weights/np.sum(weights)
        metalPosition = np.sum(weights*z_axis_fragment)
        return metalPosition
```

[6]:
```
"""
Class to read in data of one system
"""
```

```
class SurfaceDDECAnalysis:
    def __init__(self, dataRow, calculationType="PW", shift=0):

        self.calculationType = calculationType
        #Additions to select right columns
        if calculationType == "LCAO":
          _calcType= "lcao_"
        elif calculationType == "cDFT":
          _calcType = "cdft_"
        else:
          _calcType = ""

        self.sysName = str(dataRow.electrode)+"_"+str(dataRow.ads_name)
        shift = shift #Angs shifting charge density profiles to avoid discontinuity

        ## Estimate electrode charge density
        charges = dataRow["sys_"+_calcType+"chg_val"]
        self.electrodeArea = float(dataRow.area)
        self.metalIndexes = np.asarray(dataRow.numbers)==atomic_numbers[dataRow.electrode]
        metCharges = np.array(charges)[self.metalIndexes]
        self.electrodeChargeDensity = np.sum(metCharges)/self.electrodeArea #e/Å
        self.electrodeChargeDensity = self.electrodeChargeDensity*el_chg*(10**6)/(10**-16) # C/cm²

        ## Estimate potential profile
        zCoords = dataRow.positions[:,2]
        self.bin_size=0.01 #A
        ##Order atoms according to their z-coordinate
        snap_z_sort=[]
        snap_c_sort=[]
        snap_k_sort=[]
        indexes=np.argsort(zCoords)
        for index in indexes:
            snap_z_sort+=[zCoords[index]]
            snap_c_sort+=[charges[index]]

        ##Divide charges of atoms into bins to get charge density
        z_bin=np.arange(0, 60, self.bin_size)
        c_b=[0]*len(z_bin)
        for l in range(len(snap_z_sort)):
            c_b[int(snap_z_sort[l]//self.bin_size)]+=snap_c_sort[l]

        self.chgDensityDDEC = np.asarray(c_b)/(self.bin_size*self.electrodeArea) #e/Å³ #Divide charges by bin size to get charge density
        self.z_bin = z_bin
```

```
        self.potentialDDEC = [potential_at_a(a, self.z_bin, self.
↪chgDensityDDEC) for a in self.z_bin] #V
        self.potDropDDEC = self.potentialDDEC[0]-self.potentialDDEC[-1] #V
```

1.5 Matplotlib settings

[7]:
```
pd.options.mode.chained_assignment = None  # default='warn'

plt.rc('font',        size=10)  # controls default text sizes
plt.rc('axes',   titlesize=10)  # fontsize of the axes title
plt.rc('axes',   labelsize=9)   # fontsize of the x and y labels
plt.rc('xtick',  labelsize=8)   # fontsize of the tick labels
plt.rc('ytick',  labelsize=8)   # fontsize of the tick labels
plt.rc('legend', fontsize =8)   # legend fontsize
plt.rc('figure', titlesize=10)  # fontsize of the figure title
plt.rc('figure', facecolor='w') # white background
```

1.5.1 Color and shape data

[8]:
```
colorDict = defaultdict(lambda: "black")
colorDict["N"] = "blue"
colorDict["S"] = "yellow"
colorDict["O"] = "red"
```

1.6 Mount drive

[10]:
```
#from google.colab import drive
#drive.mount('/content/drive')
#path = '/content/drive/MyDrive/TartuPorto/DATA/IONS-PSC-PRL'
path = './'
```

[11]:
```
# Connect an ASE database with all results
db = connect(os.path.join(path,'pmc8.db'))
```

[12]:
```
# Convert the data into a panda dataframe
columnNames = ['version',
               'electrode',
               'ads_name',
               'ads_enrg',
               'ads_disp',
               'ads_chg',
               'ads_dist',
               'area',
               'face',
               'mode',
               'size',
```

```
              'sigma',
              'numbers',
              'positions',
              'sys_chg_val',
              'sys_dip_val',
              'sys_pot_zax',
              'sys_pot_val',
              'sys_den_zax',
              'sys_den_val',
              'ads_den_zax',
              'ads_den_val',
              'met_den_zax',
              'met_den_val',
              'sys_lcao_chg_val',
              'sys_lcao_dip_val',
              'sys_lcao_pot_zax',
              'sys_lcao_pot_val',
              'sys_lcao_den_zax',
              'sys_lcao_den_val',
              'ads_lcao_den_zax',
              'ads_lcao_den_val',
              'met_lcao_den_zax',
              'met_lcao_den_val',
              'sys_cdft_chg_val',
              'sys_cdft_dip_val',
              'sys_cdft_pot_zax',
              'sys_cdft_pot_val',
              'sys_cdft_den_zax',
              'sys_cdft_den_val',
            ]

rdf  = pd.DataFrame(columns=columnNames)

for row in db.select():
    dt = pd.DataFrame(columns=columnNames,
                      data=[[row.version,
                             row.electrode,
                             row.ads_name,
                             row.ads_enrg,
                             row.ads_disp,
                             row.ads_chg,
                             row.ads_dist,
                             row.area,
                             row.face,
                             row.mode,
                             row.size,
                             row.sigma,
```

```
                            row.numbers,
                            row.positions,
                            row.data.sys_chg_val,
                            row.data.sys_dip_val,
                            row.data.sys_pot_zax,
                            row.data.sys_pot_val,
                            row.data.sys_den_zax,
                            row.data.sys_den_val,
                            row.data.ads_den_zax,
                            row.data.ads_den_val,
                            row.data.met_den_zax,
                            row.data.met_den_val,
                            row.data.sys_lcao_chg_val,
                            row.data.sys_lcao_dip_val,
                            row.data.sys_lcao_pot_zax,
                            row.data.sys_lcao_pot_val,
                            row.data.sys_lcao_den_zax,
                            row.data.sys_lcao_den_val,
                            row.data.ads_lcao_den_zax,
                            row.data.ads_lcao_den_val,
                            row.data.met_lcao_den_zax,
                            row.data.met_lcao_den_val,
                            row.data.sys_cdft_chg_val,
                            row.data.sys_cdft_dip_val,
                            row.data.sys_cdft_pot_zax,
                            row.data.sys_cdft_pot_val,
                            row.data.sys_cdft_den_zax,
                            row.data.sys_cdft_den_val,
                           ]])
    rdf = pd.concat([rdf, dt], ignore_index=True)
```

[13]:
```
rdf
```

[13]:

	version	electrode	ads_name	ads_enrg	ads_disp	ads_chg	ads_dist	area	\
0	a	Au	C5N	-1.12	-0.82	0.568	2.9	66.09	
1	a	Au	C19O3v	-0.115	-2.22	0.484	3.14	117.5	
2	a	Au	C19N3v	-0.967	-2.392	0.707	3.08	117.5	
3	a	Au	C19N2Ov	-0.657	-2.34	0.669	3.08	117.5	
4	a	Au	C23N	-0.506	-2.498	0.634	3.18	117.5	
5	a	Au	C53N	-0.07	-5.438	0.914	3.15	264.37	
6	a	Au	C95N	7.46	-9.362	1.118	3.16	359.84	
7	a	Au	C19NO2v	-0.39	-2.257	0.597	3.1	117.5	

	face	mode	…	ads_lcao_den_zax	\
0	fcc111	PW	…	[0.0909999590729343, 0.1819999181458687, 0.272…	
1	fcc111	PW	…	[0.0909999590729343, 0.1819999181458687, 0.272…	
2	fcc111	PW	…	[0.0909999590729343, 0.1819999181458687, 0.272…	

```
3  fcc111   PW  …  [0.0909999590729343, 0.1819999181458687, 0.272…
4  fcc111   PW  …  [0.0909999590729343, 0.1819999181458687, 0.272…
5  fcc111   PW  …  [0.0909999590729343, 0.1819999181458687, 0.272…
6  fcc111   PW  …  [0.0909999590729343, 0.1819999181458687, 0.272…
7  fcc111   PW  …  [0.0909999590729343, 0.1819999181458687, 0.272…

                                      ads_lcao_den_val  \
0  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
1  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
2  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
3  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
4  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
5  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
6  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
7  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …

                                      met_lcao_den_zax  \
0  [0.0909999590729343, 0.1819999181458687, 0.272…
1  [0.0909999590729343, 0.1819999181458687, 0.272…
2  [0.0909999590729343, 0.1819999181458687, 0.272…
3  [0.0909999590729343, 0.1819999181458687, 0.272…
4  [0.0909999590729343, 0.1819999181458687, 0.272…
5  [0.0909999590729343, 0.1819999181458687, 0.272…
6  [0.0909999590729343, 0.1819999181458687, 0.272…
7  [0.0909999590729343, 0.1819999181458687, 0.272…

                                      met_lcao_den_val  \
0  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
1  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
2  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
3  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
4  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
5  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
6  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
7  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …

                                      sys_cdft_chg_val  \
0  [-0.010574, -0.013575, -0.01323, -0.014091, -0…
1  [-0.036335, -0.039491, -0.009884, -0.011074, -…
2  [-0.058961, -0.047015, -0.00791, -0.023353, -0…
3  [-0.035053, -0.024583, -0.009955, -0.024212, -…
4  [-0.011597, -0.012592, -0.011897, -0.01151, -0…
5  [-0.034409, -0.018307, -0.012476, -0.013182, -…
6  [-0.012195, -0.01341, -0.013687, -0.013358, -0…
7  [-0.018601, -0.016589, -0.01077, -0.01653, -0…

                                      sys_cdft_dip_val  \
```

```
0  [-0.119266, -0.115112, -0.115232, -0.114678, -…
1  [-0.164757, -0.162379, -0.11704, -0.116641, -0…
2  [-0.206343, -0.177474, -0.113878, -0.139782, -…
3  [-0.164229, -0.137612, -0.114473, -0.137932, -…
4  [-0.119665, -0.118003, -0.118618, -0.1198, -0…
5  [-0.151464, -0.133229, -0.11765, -0.117465, -0…
6  [-0.119057, -0.118188, -0.118875, -0.11851, -0…
7  [-0.135117, -0.125065, -0.117715, -0.125259, -…

                                    sys_cdft_pot_zax  \
0  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
1  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
2  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
3  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
4  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
5  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
6  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…
7  [0.091, 0.182, 0.273, 0.364, 0.454999999999999…

                                    sys_cdft_pot_val  \
0  [2.7572299264265507, 2.757229480706091, 2.7572…
1  [1.5121774191639776, 1.5121771747126214, 1.512…
2  [1.3352875348058522, 1.3352873189496604, 1.335…
3  [1.3457037466871309, 1.345703529147082, 1.3457…
4  [1.4507972738779498, 1.4507970393489984, 1.450…
5  [0.5935904256941208, 0.5935903297370301, 0.593…
6  [0.3814100014847857, 0.3814099324190053, 0.381…
7  [1.425775457593649, 1.4257752271096216, 1.4257…

                                    sys_cdft_den_zax  \
0  [0.0909999590729343, 0.1819999181458687, 0.272…
1  [0.0909999590729343, 0.1819999181458687, 0.272…
2  [0.0909999590729343, 0.1819999181458687, 0.272…
3  [0.0909999590729343, 0.1819999181458687, 0.272…
4  [0.0909999590729343, 0.1819999181458687, 0.272…
5  [0.0909999590729343, 0.1819999181458687, 0.272…
6  [0.0909999590729343, 0.1819999181458687, 0.272…
7  [0.0909999590729343, 0.1819999181458687, 0.272…

                                    sys_cdft_den_val
0  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
1  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
2  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
3  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
4  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
5  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
6  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
```

```
7  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …

[8 rows x 40 columns]
```

[14]:
```
systems = []
for i in range(len(rdf)):
  for calcType in ["PW", "LCAO", "cDFT"]:
    systems+=[SurfaceED(rdf.iloc[i], calculationType=calcType)]
```

[15]:
```
systemsDDEC = []
for i in range(len(rdf)):
  for calcType in ["PW", "LCAO", "cDFT"]:
    systemsDDEC+=[SurfaceDDECAnalysis(rdf.iloc[i], calculationType=calcType)]
```

[16]:
```
#Estimate potential drops for charge differences and Hartree
df = pd.DataFrame()

potDropsEdDiff = [system.potDropEdDiff for system in systems]
potDropsHartree = [system.potDropHartree for system in systems]
sigmas = [system.metalCharge for system in systems]
distances = [system.distanceBetweenMetalMolecule for system in systems]
distancesGeometric = [system.distanceBetweenMetalMoleculeGeometric for system␣
 ↪in systems]
heteroAtoms = [re.sub("[^NOS]", "", system.sysName) for system in systems]
sysNames = [system.sysName for system in systems]
calcTypes = [system.calculationType for system in systems]
areas = [system.electrodeArea for system in systems]

#Grab also DDEC data
sigmasDDEC =  [system.electrodeChargeDensity for system in systemsDDEC]
potDropsDDEC = [system.potDropDDEC for system in systemsDDEC]

df["sysName"] = sysNames
df["calcType"] = calcTypes
df["potDropHartree"] = potDropsHartree
df["potDropEdDiff"] = potDropsEdDiff
df["potDropDDEC"] = potDropsDDEC
df["sigma"] = sigmas
df["sigma[e]"] = np.array(sigmas)*np.array(areas)/el_chg/(10**6)*(10**-16)
df["sigmaDDEC"] = sigmasDDEC
df["sigmaDDEC[e]"] = np.array(sigmasDDEC)*np.array(areas)/el_chg/
 ↪(10**6)*(10**-16)
df["distance"] = distances
df["distanceGeometric"] = distancesGeometric
df["coverage"] = [getSurfaceCoverage(system.electrodeArea) for system in␣
 ↪systems]
df["heteroAtom"] = heteroAtoms
```

[17]:
```
df
```

[17]:

	sysName	calcType	potDropHartree	potDropEdDiff	potDropDDEC
0	Au_C5N	PW	-2.183221	-2.599209	-3.805396
1	Au_C5N	LCAO	-2.268615	-2.678370	-3.661137
2	Au_C5N	cDFT	-5.514444	-5.924256	-7.678217
3	Au_C19O3v	PW	-1.334801	-1.474399	-2.412207
4	Au_C19O3v	LCAO	-1.338951	-1.469358	-2.231595
5	Au_C19O3v	cDFT	-3.024346	-3.154304	-4.482663
6	Au_C19N3v	PW	-1.642138	-2.030408	-3.286638
7	Au_C19N3v	LCAO	-1.749939	-2.128106	-3.135332
8	Au_C19N3v	cDFT	-2.670567	-3.048774	-4.262455
9	Au_C19N2Ov	PW	-1.565421	-1.898401	-2.989137
10	Au_C19N2Ov	LCAO	-1.620926	-1.945172	-2.794408
11	Au_C19N2Ov	cDFT	-2.691400	-3.015453	-4.166551
12	Au_C23N	PW	-1.620191	-1.806339	-2.896697
13	Au_C23N	LCAO	-1.664282	-1.818813	-2.695815
14	Au_C23N	cDFT	-2.901586	-3.055766	-4.290684
15	Au_C53N	PW	-1.181785	-1.288000	-2.098997
16	Au_C53N	LCAO	-1.172380	-1.264401	-1.934229
17	Au_C53N	cDFT	-1.187177	-1.279180	-1.953677
18	Au_C95N	PW	-1.073124	-1.149213	-1.865382
19	Au_C95N	LCAO	-1.026493	-1.097094	-1.617242
20	Au_C95N	cDFT	-0.762818	-0.833375	-1.285449
21	Au_C19NO2v	PW	-1.472122	-1.714088	-2.646046
22	Au_C19NO2v	LCAO	-1.511017	-1.741721	-2.464043
23	Au_C19NO2v	cDFT	-2.851543	-3.081265	-4.230852

	sigma	sigma[e]	sigmaDDEC	sigmaDDEC[e]	distance	distanceGeometric
0	-11.306423	-0.466391	-13.770355	-0.568029	1.989027	1.836170
1	-11.360587	-0.468626	-13.505605	-0.557108	2.247722	1.836170
2	-24.127794	-0.995275	-26.805852	-1.105745	2.381377	1.836170
3	-7.013839	-0.514379	-6.599277	-0.483976	1.870225	2.031931
4	-6.760272	-0.495783	-6.443545	-0.472555	2.077090	2.031931
5	-13.662882	-1.002005	-13.418918	-0.984113	2.141628	2.031931
6	-9.717940	-0.712692	-9.640590	-0.707019	1.859294	1.986841
7	-9.894313	-0.725626	-9.629613	-0.706214	2.079107	1.986841
8	-13.615178	-0.998506	-13.234988	-0.970624	2.105954	1.986841
9	-9.088740	-0.666548	-9.118458	-0.668727	1.851672	1.993774
10	-8.995430	-0.659704	-8.931964	-0.655050	2.081218	1.993774
11	-13.387282	-0.981793	-13.293089	-0.974885	2.108411	1.993774
12	-8.871772	-0.650636	-8.650390	-0.634400	1.840503	2.066703
13	-8.580836	-0.629299	-8.480191	-0.621918	2.036385	2.066703
14	-13.603396	-0.997642	-13.445684	-0.986076	2.080132	2.066703
15	-6.073197	-1.002119	-5.537318	-0.913695	1.899874	2.063971
16	-5.893600	-0.972484	-5.495477	-0.906791	2.044850	2.063971
17	-5.963457	-0.984011	-5.568377	-0.918820	2.042641	2.063971

```
18  -5.813535 -1.305688  -4.975822     -1.117542  1.856922          2.072029
19  -5.515574 -1.238767  -4.789076     -1.075600  1.929662          2.072029
20  -4.499401 -1.010540  -3.778944     -0.848730  1.925058          2.072029
21  -8.156445 -0.598175  -8.145548     -0.597376  1.855371          2.024265
22  -7.958282 -0.583642  -7.960459     -0.583802  2.097225          2.024265
23 -13.448106 -0.986254 -13.481873     -0.988730  2.137079          2.024265

        coverage heteroAtom
0   2.512542e-10          N
1   2.512542e-10          N
2   2.512542e-10          N
3   1.413225e-10          O
4   1.413225e-10          O
5   1.413225e-10          O
6   1.413225e-10          N
7   1.413225e-10          N
8   1.413225e-10          N
9   1.413225e-10         NO
10  1.413225e-10         NO
11  1.413225e-10         NO
12  1.413225e-10          N
13  1.413225e-10          N
14  1.413225e-10          N
15  6.281117e-11          N
16  6.281117e-11          N
17  6.281117e-11          N
18  4.614659e-11          N
19  4.614659e-11          N
20  4.614659e-11          N
21  1.413225e-10         NO
22  1.413225e-10         NO
23  1.413225e-10         NO
```

[18]:
```
####This table here is for electron density difference values

dfTable = pd.DataFrame()

selectionPW = df[df["calcType"]=="PW"]
selectionCDFT = df[df["calcType"]=="cDFT"]

dfTable["Compound"] = selectionPW.sysName.values
dfTable["varphi"] = selectionPW.potDropEdDiff.values
dfTable["varphi_cDFT"] = selectionCDFT.potDropEdDiff.values
dfTable["sigma"] = selectionPW.sigma.values
dfTable["sigma_cDFT"] = selectionCDFT.sigma.values
dfTable["q_ion[e]"] = -selectionPW["sigma[e]"].values
dfTable["q_ion_cDFT[e]"] = -selectionCDFT["sigma[e]"].values
```

```
dfTable["l"] = selectionPW.distance.values
dfTable["d"] = selectionPW.distanceGeometric.values
dfTable = dfTable.sort_values(by="varphi", ascending=False)
```

[19]:
```
dfTable
```

[19]:

	Compound	varphi	varphi_cDFT	sigma	sigma_cDFT	q_ion[e]	\
6	Au_C95N	-1.149213	-0.833375	-5.813535	-4.499401	1.305688	
5	Au_C53N	-1.288000	-1.279180	-6.073197	-5.963457	1.002119	
1	Au_C19O3v	-1.474399	-3.154304	-7.013839	-13.662882	0.514379	
7	Au_C19NO2v	-1.714088	-3.081265	-8.156445	-13.448106	0.598175	
4	Au_C23N	-1.806339	-3.055766	-8.871772	-13.603396	0.650636	
3	Au_C19N2Ov	-1.898401	-3.015453	-9.088740	-13.387282	0.666548	
2	Au_C19N3v	-2.030408	-3.048774	-9.717940	-13.615178	0.712692	
0	Au_C5N	-2.599209	-5.924256	-11.306423	-24.127794	0.466391	

	q_ion_cDFT[e]	l	d
6	1.010540	1.856922	2.072029
5	0.984011	1.899874	2.063971
1	1.002005	1.870225	2.031931
7	0.986254	1.855371	2.024265
4	0.997642	1.840503	2.066703
3	0.981793	1.851672	1.993774
2	0.998506	1.859294	1.986841
0	0.995275	1.989027	1.836170

[20]:
```
####This table here is for DDEC values

dfTable2 = pd.DataFrame()
selectionPW = df[df["calcType"]=="PW"]
selectionCDFT = df[df["calcType"]=="cDFT"]
dfTable2["Compound"] = selectionPW.sysName.values
dfTable2["varphi"] = selectionPW.potDropDDEC.values
dfTable2["varphi_cDFT"] = selectionCDFT.potDropDDEC.values
dfTable2["sigma"] = selectionPW.sigmaDDEC.values
dfTable2["sigma_cDFT"] = selectionCDFT.sigmaDDEC.values
dfTable2["q_ion[e]"] = -selectionPW["sigmaDDEC[e]"].values
dfTable2["q_ion[e]_cDFT"] = -selectionCDFT["sigmaDDEC[e]"].values
dfTable2 = dfTable2.sort_values(by="varphi", ascending=False)
dfTable2
```

[20]:

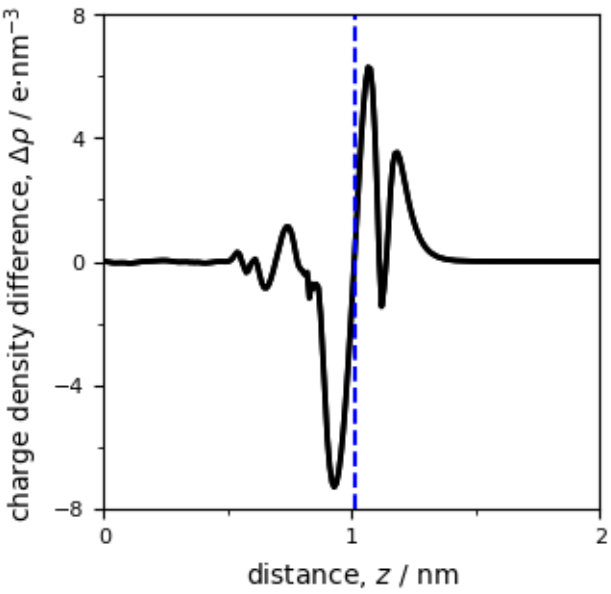
	Compound	varphi	varphi_cDFT	sigma	sigma_cDFT	q_ion[e]	\
6	Au_C95N	-1.865382	-1.285449	-4.975822	-3.778944	1.117542	
5	Au_C53N	-2.098997	-1.953677	-5.537318	-5.568377	0.913695	
1	Au_C19O3v	-2.412207	-4.482663	-6.599277	-13.418918	0.483976	
7	Au_C19NO2v	-2.646046	-4.230852	-8.145548	-13.481873	0.597376	
4	Au_C23N	-2.896697	-4.290684	-8.650390	-13.445684	0.634400	

```
3  Au_C19N2Ov -2.989137    -4.166551  -9.118458  -13.293089  0.668727
2   Au_C19N3v -3.286638    -4.262455  -9.640590  -13.234988  0.707019
0      Au_C5N -3.805396    -7.678217 -13.770355  -26.805852  0.568029

   q_ion[e]_cDFT
6       0.848730
5       0.918820
1       0.984113
7       0.988730
4       0.986076
3       0.974885
2       0.970624
0       1.105745
```

1.7 Figure illustrating the location of metal-molecule boundary

[21]:
```
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(3.33, 3.33), dpi=100)
sysIndex = 0
print(systems[sysIndex].sysName)
ax.plot(systems[sysIndex].z_Ed/10, systems[sysIndex].chgChange*1000, color="k",␣
 ↪lw=2)
ax.plot(systems[sysIndex].z_Ed/10, systems[sysIndex].chgChange*1000, color="k",␣
 ↪lw=2)

ax.axvline(systems[sysIndex].metalMoleculeBoundary/10, color="blue", ls="--")
ax.set_xlabel(r"distance, $z$ / nm", fontsize=10)
ax.set_ylabel(r"charge density difference, $\Delta\rho$ / e$\cdot$nm$^{-3}$",␣
 ↪fontsize=10)



ax.set_xlim(0.5, 2.5)
x_labels = np.arange(0.5, 2.6, 1)
ax.set_xticks(x_labels)
ax.set_xticklabels((x_labels-min(x_labels)).round(0).astype(int))
ax.xaxis.set_minor_locator(MultipleLocator(0.5))

ax.set_ylim(-8, 8)
y_labels = np.arange(-8, 9, 4)
ax.set_yticks(y_labels)
ax.yaxis.set_minor_locator(MultipleLocator(2))

ax.tick_params(axis="both", which="both", labelsize=8)
#fig.savefig(os.path.join(path,"deltaRho-vs-distance.svg"))
```

```
Au_C5N
```



1.8 Fitting Linear model to sigma × distance vs potential drop

[22]:
```
df_PW = df[df["calcType"] == "PW"]
```

[23]:
```
distanceTimesSigma = df_PW["sigma"].values*df_PW["distance"].values

print("From WF potential")

ols_WF = LinearRegression(fit_intercept=False)
ols_WF.fit(distanceTimesSigma.reshape(-1,1), df_PW["potDropHartree"])
print('R2 %3.3f ' % (ols_WF.score(distanceTimesSigma.reshape(-1,1),␣
 ↪df_PW["potDropHartree"])))

#Fit two point for line illustration
X_test = np.asarray([-60,0]).reshape(-1,1)
y_test = ols_WF.predict(X_test)

ols_coef_WF = ols_WF.coef_[0]
print(f"Incline: {ols_coef_WF}")

epsilon_estimated_WF = 1/(ols_coef_WF*1.602*10**3) #e/V/Angs
print(f"Estimated epsilon: {epsilon_estimated_WF} e/(V*Angs)")
print(f"eps_estimated/eps0 {epsilon_estimated_WF/eps0}")
```

```
From WF potential
R2 0.968
Incline: 0.09670172229423957
```

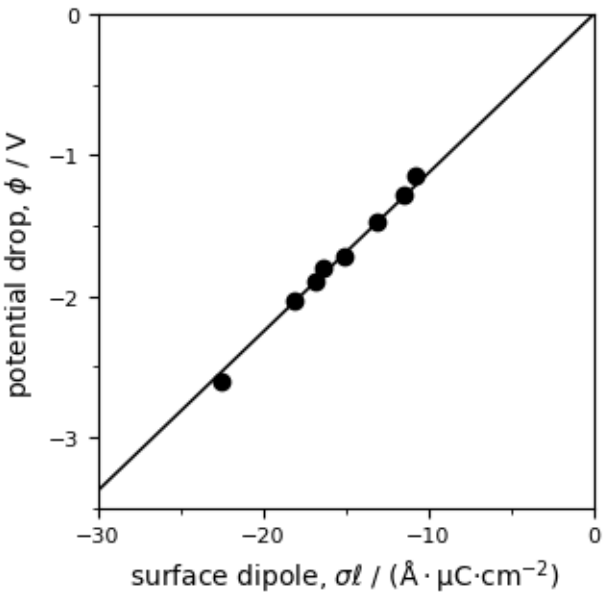
```
Estimated epsilon: 0.0064551045269284 22 e/(V*Angs)
eps_estimated/eps0 1.1680594294179847
```

[24]:
```
ols_ChgDiff = LinearRegression(fit_intercept=False)
ols_ChgDiff.fit(distanceTimesSigma.reshape(-1,1), df_PW["potDropEdDiff"])
print("From Charge difference potential")

print('R2 %3.3f ' % (ols_ChgDiff.score(distanceTimesSigma.reshape(-1,1),␣
 ↪df_PW["potDropEdDiff"])))

#Fit two point for line illustration
X_test = np.asarray([-60,0]).reshape(-1,1)
y_test = ols_ChgDiff.predict(X_test)

ols_coef_ChgDiff = ols_ChgDiff.coef_[0]
print(f"Incline: {ols_coef_ChgDiff}")
epsilon_estimated_ChgDiff = 1/(ols_coef_ChgDiff*1.602*10**3) #e/V/Angs
print(f"Estimated epsilon: {epsilon_estimated_ChgDiff} e/(V*Angs)")
print(f"eps_estimated/eps0 {epsilon_estimated_ChgDiff/eps0}")
```

```
From Charge difference potential
R2 0.993
Incline: 0.11271581696807884
Estimated epsilon: 0.0055379958388635025 e/(V*Angs)
eps_estimated/eps0 1.0021074380247297
```

1.8.1 Figure illustrating charge density × distance vs potential drop

[25]:
```
fig, ax1 = plt.subplots(nrows=1, ncols=1, figsize=(3.33, 3.33), dpi=100)
ax1.plot(df_PW["sigma"].values*df_PW["distance"].values,␣
 ↪df_PW["potDropEdDiff"], 'o', markerfacecolor="k", markeredgecolor="k", ␣
 ↪label=r'$\Delta \rho$')
ax1.plot(X_test, y_test, color="k", ls="-", lw=1.1, label="Fitted line")
ax1.set_xlabel(r"surface dipole, $\sigma \ell$ / ($\mathrm{\AA} \cdot␣
 ↪\mathrm{\mu}$C$\cdot$cm$^{-2}$)", fontsize=10)
ax1.set_ylabel(r"potential drop, $\phi$ / V", fontsize=10)
ax1.set_xlim(-30,0)
ax1.set_ylim(-3.5, 0)
ax1.set_xticks(np.arange(-30, 1, 10))
ax1.xaxis.set_minor_locator(MultipleLocator(5))
ax1.set_yticks(np.arange(-3.0, 0.1, 1))
ax1.yaxis.set_minor_locator(MultipleLocator(0.5))
ax1.tick_params(axis="both", which="both", labelsize=8)
#fig.savefig(os.path.join(path,"potentialDrop-vs-surfaceDipole.svg"))
```


0
−1
−2
−3
potential drop, ϕ / V
−30
−20
−10
0
surface dipole, $\sigma\ell$ / (Å · µC·cm^{-2})

[ ]: