Problem Set 7: Hybridization, molecular structures and the Hückel model

For questions contact: hanning.zhang@epfl.ch

Exercise 1: Hybridization

In this exercise we discuss hybridizations involving the s and p orbitals.

a) Recall that any action of an element in the rotation group $R \in SO(3)$ transforms a state ψ as

$$\psi(\vec{r}) \to \psi'(\vec{r}) = \psi(R^{-1}\vec{r})$$

Show that this action leaves the s-state $|s\rangle$ unaltered, while a p-state $|p\rangle = \sum_{i\in\{x,y,z\}} a_i|p_i\rangle$ is transformed to $|p'\rangle = \sum_{i,j\in\{x,y,z\}}(R_{ij}a_j)|p_i\rangle = \sum_{i\in\{x,y,z\}} a'_i|p_i\rangle$. In other words taking the coefficients a_i as a vector, R acts as a matrix multiplication from the left side on $\vec{a}$ giving the new vector $\vec{a}'$.

Hint: The s-state $|s(\vec{r})\rangle = s(r)$ doesn't have an angular dependence, while the p-states can be written as $|p_i(\vec{r})\rangle = r_i p(r)$ (with $i = x, y, z$), where $p(r)$ doesn't show any angular dependence.

b) We first look at the sp hybridization, where we construct two hybrid orbitals out of a $|s\rangle$ and a $|p_z\rangle$ orbital

$$|sp_1\rangle = s_1|s\rangle + p_{z1}|p_z\rangle$$
$$|sp_2\rangle = s_2|s\rangle + p_{z2}|p_z\rangle$$

This hybridization is used, if a central atom forms bonds with two equal partners on both its sides of a given axis (here z-axis). The symmetry constraint is hence that $|sp_2\rangle$ can be obtained from $|sp_1\rangle$ by a 180° rotation. Using this together with the orthonormality condition, as well as the requirement that all coefficients are real, derive coefficients s_1, s_2, p_{z1}, p_{z2}.

c) Simiarly the sp^2 hybridization constructs the 3 hyrbid orbitals out of the $|s\rangle$, $|p_x\rangle, p_y\rangle$ orbitals

$$|sp_1^2\rangle = s_1|s\rangle + p_{x1}|p_x\rangle + p_{y1}|p_y\rangle$$
$$|sp_2^2\rangle = s_2|s\rangle + p_{x2}|p_x\rangle + p_{y2}|p_y\rangle$$
$$|sp_3^2\rangle = s_3|s\rangle + p_{x3}|p_x\rangle + p_{y3}|p_y\rangle$$

This hybridization is used, if the central atom forms 3 equivalent bonds with partners in the same plane. Proceed as in exercise b) and find the coefficients s_i, p_{xi}, p_{yi}.

$Remark:$ Note, there is an additional degree of freedom to choose one of the hybridized orbitals to point along a specific direction in the plane. We will put $p_{y1} = 0$, which means that $|sp_1^2\rangle$ will point along the x-direction.

d) Finally we look at the sp^3 hybridization, which constructs the 4 hyrbid orbitals out of the $|s\rangle$, $|p_x\rangle$, $|p_y\rangle$, $|p_z\rangle$ orbitals.

$$|sp_1^3\rangle = s_1|s\rangle + p_{x1}|p_x\rangle + p_{y1}|p_y\rangle + p_{z1}|p_z\rangle$$
$$|sp_2^3\rangle = s_2|s\rangle + p_{x2}|p_x\rangle + p_{y2}|p_y\rangle + p_{z2}|p_z\rangle$$
$$|sp_3^3\rangle = s_3|s\rangle + p_{x3}|p_x\rangle + p_{y3}|p_y\rangle + p_{z3}|p_z\rangle$$
$$|sp_4^3\rangle = s_3|s\rangle + p_{x4}|p_x\rangle + p_{y4}|p_y\rangle + p_{z4}|p_z\rangle$$

This hybridization is used, if the central atom forms 4 equivalent bonds with partners yielding a tetrahedral symmetry. Proceed as in exercise b) and find the coefficients.

$Remark:$ Keep in mind that there are three rotational degrees of freedom. You may choose them such that the calculation will be as simple as possible. We propose to start with $p_{x1} = p_{y1} = p_{z1}$, which fixes two of the three degrees of freedom.

e) Plot the orbitals of the sp, sp^2 and sp^3 hybridization.

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt  # Import library for direct plotting functi
import numpy as np  # Import Numerical Python
import math
from cmath import phase
vphase =np.vectorize(phase)

from scipy.special import sph_harm,genlaguerre
from IPython.display import HTML
```

In [ ]:
```
theta, phi = np.linspace(0,  np.pi, 40), np.linspace(0, 2*np.pi, 40)
THETA, PHI = np.meshgrid(theta, phi)


s = sph_harm(m,l,PHI,THETA).real #Please fill in l, m
px = np.sqrt(2)*(-1.)**1*sph_harm(m,l,PHI,THETA).real #Please fill in l, m
py = np.sqrt(2)*(-1.)**(-1)*sph_harm(m,l,PHI,THETA).imag #Please fill in l,
```

```
pz = sph_harm(m,l,PHI,THETA).real #Please fill in l, m
allorb = np.concatenate((s[:,:, np.newaxis], px[:,:, np.newaxis], py[:,:, np
```

In [ ]:
```
#%matplotlib widget use matplotlib widget if you want to interactively rotat
%matplotlib inline

coeffs =  #Write your coefficient matrix from b)
n = len(coeffs)



fig = plt.figure(figsize=(9.5, 4))
fig.suptitle('Orbitals sp')
for i in np.arange(n):
    aux = np.zeros([len(theta), len(phi)])
    for j in np.arange(n):
        aux = aux + coeffs[i, j]*allorb[:,:,j]
    R = abs(aux)
    arg = np.sign(aux)
    X = R * np.sin(THETA) * np.cos(PHI)
    Y = R * np.sin(THETA) * np.sin(PHI)
    Z = R * np.cos(THETA)

    fcolors = arg
    fmax, fmin = fcolors.max(), fcolors.min()
    if fmax-fmin > 0:
        fcolors = (fcolors - fmin)/(fmax - fmin)

    ax = fig.add_subplot(1,n,i+1, projection='3d')
    ax.set_axis_off()
    plot = ax.plot_surface(X, Y, Z,
    linewidth=0, antialiased=False, alpha=0.4,facecolors=plt.cm.seismic(fcol
    ax.quiver(0, 0, 0, 0.4, 0, 0, color='k', arrow_length_ratio=0.1)  # X-ax
    ax.quiver(0, 0, 0, 0, 0.4, 0, color='k', arrow_length_ratio=0.1)  # Y-ax
    ax.quiver(0, 0, 0, 0, 0, 0.4, color='k', arrow_length_ratio=0.1)  # Z-ax
    ax.text(0.4, 0, 0, "x", color='k', fontsize=12)
    ax.text(0, 0.4, 0, "y", color='k', fontsize=12)
    ax.text(0, 0, 0.4, "z", color='k', fontsize=12)
    ax.set_aspect('equal')
plt.show()
```

In [ ]:
```
#%matplotlib widget use matplotlib widget if you want to interactively rotat
%matplotlib inline

coeffs =  #Write your coefficient matrix from c)
n = len(coeffs)


fig = plt.figure(figsize=(9.5, 4))
fig.suptitle(r'Orbitals $sp^2$')
for i in np.arange(n):
    aux = np.zeros([len(theta), len(phi)])
    for j in np.arange(n):
        aux = aux + coeffs[i, j]*allorb[:,:,j]
    R = abs(aux)
```

```
    arg = np.sign(aux)
    X = R * np.sin(THETA) * np.cos(PHI)
    Y = R * np.sin(THETA) * np.sin(PHI)
    Z = R * np.cos(THETA)

    fcolors = arg
    fmax, fmin = fcolors.max(), fcolors.min()
    if fmax - fmin > 0:
        fcolors = (fcolors - fmin)/(fmax - fmin)

    ax = fig.add_subplot(1,n,i+1, projection='3d')
    ax.set_axis_off()
    plot = ax.plot_surface(
    X, Y, Z,
    linewidth=0, antialiased=False, alpha=0.4,facecolors=plt.cm.seismic(fcol
    ax.quiver(0, 0, 0, 0.4, 0, 0, color='k', arrow_length_ratio=0.1)  # X-ax
    ax.quiver(0, 0, 0, 0, 0.4, 0, color='k', arrow_length_ratio=0.1)  # Y-ax
    ax.quiver(0, 0, 0, 0, 0, 0.4, color='k', arrow_length_ratio=0.1)  # Z-ax
    ax.text(0.4, 0, 0, "x", color='k', fontsize=12)
    ax.text(0, 0.4, 0, "y", color='k', fontsize=12)
    ax.text(0, 0, 0.4, "z", color='k', fontsize=12)
    ax.set_aspect('equal')
plt.show()
```

In [ ]:

```
#%matplotlib widget use matplotlib widget if you want to interactively rotat
%matplotlib inline

coeffs = #Write your coefficient matrix from d)
n = len(coeffs)


fig = plt.figure(figsize=(9.5, 4))
fig.suptitle(r'Orbitals $sp^3$')
for i in np.arange(n):
    aux = np.zeros([len(theta), len(phi)])
    for j in np.arange(n):
        aux = aux + coeffs[i, j]*allorb[:,:,j]
    R = abs(aux)
    arg = np.sign(aux)
    X = R * np.sin(THETA) * np.cos(PHI)
    Y = R * np.sin(THETA) * np.sin(PHI)
    Z = R * np.cos(THETA)

    fcolors = arg
    fmax, fmin = fcolors.max(), fcolors.min()
    if fmax - fmin > 0:
        fcolors = (fcolors - fmin)/(fmax - fmin)

    ax = fig.add_subplot(1,n,i+1, projection='3d')
    ax.set_axis_off()
    plot = ax.plot_surface(
    X, Y, Z,
    linewidth=0, antialiased=False, alpha=0.4,facecolors=plt.cm.seismic(fcol
    ax.quiver(0, 0, 0, 0.4, 0, 0, color='k', arrow_length_ratio=0.1)  # X-ax
    ax.quiver(0, 0, 0, 0, 0.4, 0, color='k', arrow_length_ratio=0.1)  # Y-ax
    ax.quiver(0, 0, 0, 0, 0, 0.4, color='k', arrow_length_ratio=0.1)  # Z-ax
```

```
    ax.text(0.4, 0, 0, "x", color='k', fontsize=12)
    ax.text(0, 0.4, 0, "y", color='k', fontsize=12)
    ax.text(0, 0, 0.4, "z", color='k', fontsize=12)
    ax.set_aspect('equal')
plt.show()
```

Exercise 2: Molecules

In this exercise we determine the structure of different molecules using the knowledge

We are interested in the structure of the following molecules $\mathrm{C_2H_6(H_3\ce{C-CH_3})}$, $\mathrm{N_2H_4(H_2\ce{N-NH_2})}$, $\mathrm{H_2O_2(H\ce{O-OH})}$, \text{H}\text{C}\text{N} (\text{H}\ce{C#N}) and $\mathrm{H_2CCCH_2(H_2\ce{C=C=CH_2})}$.

a) For every atom (C, N, O) determine the hybridization, such that the bond orders given in the brackets can be satisfied.

b) Sketch the 3D structure of the molecules.

Remark: You can use this interactive website https://app.molview.com/ to visualize your molecules.

Exercise 3: The Hückel model

We use the Hückel model for computing of the energy spectra and charge distribution

In first part of this exercise we will study the butadiene molecule $\mathrm{C_4H_6}$ (see Figure 1).

ai) According to the Hückel model, set up the Hamiltonian for butadiene describing the p_z orbitals of the carbon chain. Write the Hamiltonian in terms of the onsite energies α and the hoppings between two neighbouring carbon atoms $\beta < 0$.

aii) Calculate the eigenvalues of the matrix.

aiii) Keeping in mind that each molecular orbital can be populated by two electrons, what is the energy of the HOMO (highest occupied molecular orbital) and the LUMO (lowest unoccupied molecular orbital) in the ground state. What is the HOMO-LUMO energy difference (HOMO-LUMO gap)?

aiv) We will now examine the eigenvectors:
$(a, b, b, a), (a, -b, b, -a), (b, a, -a, -b), (b, -a, -a, b)$ with $a, b > 0$. Without calculating them explicitely, can you order them in energetically ascending order.


Figure 1: butadiene molecule.

The second part of this exercise will be concerned with the benzene molecule (see Figure 2)

b) Numerically calculate the molecular orbital energies and wavefunctions. To do so, use the provided code and setup the corresponding Hückel matrix. We will take the common approximation $\beta = -2.7$ eV for the hoppings, while α is set to zero. You can use the second python cell to visualize your results.


Figure 2: benzene molecule.

In [2]:
```
import numpy as np
from scipy.linalg import eigh
import matplotlib.pyplot as plt

def benzene():
    #Calculate the spectrum in eV and eigenfunctions
    beta = -2.7 # hopping in eV
    mat = np.zeros([6, 6])
    ### Set the hoppings by hand mat[i, j] = beta
    evs, evecs = eigh(mat)
    return evs, evecs

evs, evecs = benzene()
```

In [ ]:
```
##No additional input needed
## If the hybridization orbitals are displayed (from exercise 1), please rer
%matplotlib inline

n = 6
r = 1
angles = np.linspace(0, 2 * np.pi, n, endpoint=False)
x_coords6 = r * np.cos(angles)
y_coords6 = r * np.sin(angles)

fig, ax = plt.subplots(2, 3, figsize = (17, 10))
fig.suptitle('Benzene molecule eigenfunctions')
for i in np.arange(6):
    ax[int(i/3), i%3].set_title('Energy = {:.2f} eV'.format(evs[i]))
    sign = np.sign(evecs[:, i])
    ax[int(i/3), i%3].plot( np.append(x_coords6, x_coords6[0]),  np.append(y
    ax[int(i/3), i%3].scatter(x_coords6, y_coords6, c=sign, cmap = 'grey',
    # Filled circles have positive sign, while empty circles have negative s
    # The size of the circle reflects the absolute value
    plt.axis('scaled')
```

The last part of this exercise is concerned with naphtalene and azulene(see Figure 3 and 4) that are isomers, This means that both have the same molecular formula $\mathrm{C_{10}H_8}$, but

different structures. Furhtermore all carbon atoms in the two molecules have sp^2 hybridization. Nevertheless we will see that their energy spectra and molecular orbitals, and hence their physical properties, are quite different. We will demonstrate this with the example of the electrical dipole moment.

ci) Set up the Hückel Hamiltonian and calculate the spectra and eigenvectors for both molecules. Use $\beta = -2.7$ eV and onsite energy $\alpha = 0$ eV.

cii) Compare the sizes of the HOMO-LUMO energy gaps of the two molecules.

\newline

Figure 3: Azulene molecule.

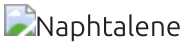

Figure 4: Naphtalene molecule.

In [1]:
```
import numpy as np
from scipy.linalg import eigh
import matplotlib.pyplot as plt

def naphtalene():
    #Calculate the spectrum in eV and eigenfunctions
    t = -2.7 # hopping in eV
    mat = np.zeros([10, 10])
    ### Set the hoppings by hand mat[i, j] = t
    evs, evecs = eigh(mat)
    return evs, evecs

def azulene():
    #Calculate the spectrum in eV and eigenfunctions
    t = -2.7 # hopping in eV
    mat = np.zeros([10, 10])
    ### Set the hoppings by hand mat[i, j] = t
    evs, evecs = eigh(mat)
    return evs, evecs
```

In [ ]:
```
evs, evecs = naphtalene()
print('The band gap of azulene is', , 'eV')
evs, evecs = azulene()
print('The band gap of naphtalene is', , 'eV')
```

The azulene molecule has a remarkably big dipole moment, which we will try to estimate in the following

ciii) Numerically define the positions of the carbon atoms assuming all (nearest neighbour) $\mathrm{C}-\mathrm{C}$ distances are $l = 1.39$ Ang. You can use the provided code snipplets.

In [ ]:
```
## If the hybridization orbitals are displayed (from exercise 1), please rer
%matplotlib inline

l = 1.39 # C-C bond length in Ang widget

# Generating a unit heptagon
n = 7
r = 1
rotation_angle = ##Fill in a rotation angle
angles = np.linspace(0 + rotation_angle, 2 * np.pi + rotation_angle, n, endp
x_coords7 = r * np.cos(angles)
y_coords7 = r * np.sin(angles)


## Scaling
scale =  ##appropriately scale the hexagon
x_coords7 = scale *x_coords7
y_coords7 = scale *y_coords7



# Generating a unit pentagon
n = 5
r = 1
rotation_angle = ## Fill in a rotation angle
angles = np.linspace(0 + rotation_angle, 2 * np.pi + rotation_angle, n, endp
x_coords5 = r * np.cos(angles)
y_coords5 = r * np.sin(angles)

##Scaling
scale = ##appropriately scale the pentagon
x_coords5 = scale *x_coords5
y_coords5 = scale *y_coords5


##Shift
x_shift = #appropriately shift the pentagon
y_shift = #appropriately shift the pentagon
x_coords5 = x_coords5 + x_shift
y_coords5 = y_coords5 + y_shift


# Plotting the heptagon + pentagon
plt.title('Azulene molecule')
plt.plot(x_coords7, y_coords7,  'o',  color='k')
plt.plot(x_coords5, y_coords5,  'o',  color='k')
plt.plot( np.append(x_coords5, x_coords5[0]),  np.append(y_coords5, y_coords
plt.plot( np.append(x_coords7, x_coords7[0]),  np.append(y_coords7, y_coords
plt.axis('scaled')
plt.xlabel('x [Ang]')
plt.ylabel('y [Ang]')
plt.show()

pos_a = np.zeros([10, 2])   #positions of the azulene molecule
pos_a[:,0] = np.concatenate((x_coords7[#:#], x_coords5[#:#])) #Two of the tw
```

```
pos_a[:,1] = np.concatenate((y_coords7[#:#], y_coords5[#:#])) #Two of the tw

#Depending on how you set up your Hückel matrix it might be appropriate to r
#your pos_a vector such that evecs and pos_a use the same convention (i.e. t
#to the same carbon atom)
```

civ) Use the results from ci) to calculate the electron occupation for each carbon atom in the ground state. Deduce from this the dipole moment of azulene.

Hint: The electron occupation ρ_i of an atom i is given by

$$\rho_i = 2 \sum_j |\psi_j(i)|^2$$

where the sum extends over the occupied molecular orbitals ψ_j and the factor 2 is accounting for the spin degeneracy. In the exercise, $\psi_j(i)$ corresponds to the i-th entry of the j-th eigenvector. Using the electron occupation ρ_i the dipole moment is furthermore given by

$$\vec{d} = \sum_i (1 - \rho_i)\vec{r}_i$$

cv) Show, using explicit numerical computation that the dipole moment of naphtalene is 0. Justify this result using symmetry.

In [ ]:
```
#cv) ## If the hybridization orbitals are displayed (from exercise 1), pleas
%matplotlib inline


l = 1.39 # C-C bond length in Ang

# Generating a unit hexagon
n = 6
r = 1
rotation_angle = ##Fill in a rotation angle
angles = np.linspace(0 + rotation_angle, 2 * np.pi + rotation_angle, n, endp
x_coords61 = r * np.cos(angles)
y_coords61 = r * np.sin(angles)
## Scaling
scale = ##appropriately scale the hexagon
x_coords61 = scale *x_coords61
y_coords61 = scale *y_coords61



# Generating a unit hexagon
n = 6
r = 1
rotation_angle = ##Fill in a rotation angle
angles = np.linspace(0 + rotation_angle, 2 * np.pi + rotation_angle, n, endp
x_coords62 = r * np.cos(angles)
y_coords62 = r * np.sin(angles)
```

```
## Scaling
scale = ##appropriately scale the hexagon
x_coords62 = scale *x_coords62
y_coords62 = scale *y_coords62
x_shift = #appropriately shift the hexagon
y_shift = #appropriately shift the hexagon
x_coords62 = x_coords62 + x_shift
y_coords62 = y_coords62 + y_shift


# Plotting the two hexagons
plt.title('Naphtalene molecule')
plt.plot(x_coords61, y_coords61,  'o',  color='k')
plt.plot(x_coords62, y_coords62,  'o',  color='k')
plt.plot( np.append(x_coords61, x_coords61[0]),  np.append(y_coords61, y_coo
plt.plot( np.append(x_coords62, x_coords62[0]),  np.append(y_coords62, y_coo
plt.axis('scaled')
plt.xlabel('x [Ang]')
plt.ylabel('y [Ang]')
plt.show()

pos_n = np.zeros([10, 2])   #positions of the naphtalene molecule

pos_n[:,0] = np.concatenate((x_coords61[#:#], x_coords62[#:#]))#Two of the t
pos_n[:,1] = np.concatenate((y_coords61[#:#], y_coords62[#:#]))#Two of the t


#Depending on how you set up your Hückel matrix it might be appropriate to r
#your pos_a vector such that evecs and pos_a use the same convention (i.e. t
#to the same carbon atom)
```