Homework - Huckel method

Numerical linear algebra - TC (course 2023)

Felipe Reibnitz Willemann

Autonomous University of Madrid

1 Introduction

One of the fundamental problems in quantum chemistry is the accurate calculation of orbital energies of molecules. A first approximation to do this for conjugated hydrocarbons is the Huckel method, for which a code in FORTRAN90 has been developed to solve.

This report contains a theoretical description of the method, explanations of the code used and the obtained results.

1.1 Huckel's method

This is one of the most simple methods for calculating the electronic energies of molecular orbitals of π-electrons in π-delocalized molecules. It's based in two main ideas: the linear combination of atomic orbitals (LCAO) of a n-dimension basis composed by the 2p$_z$ orbitals of the carbon atoms of the molecule $\{|\phi_i^{2p_z}\rangle\}_{i=1,...,n}$; and the first neighbor effective Hamiltonian approximation, which is defined as:

$$H_{ij} = \begin{cases} \alpha, \text{ if } i = j \\ \beta, \text{ if i,j adjacent} \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where α and β are negative energy parameters that need to be defined independently and depend on the atoms and the structure of the molecule. Given the molecular orbitals and the overlap matrix:

$$|\psi_i\rangle = \sum_{j=1}^{n} c_{i,j}|\phi_j^{2p_z}\rangle \tag{2}$$

$$S_{ij} = \langle\psi_i|\psi_j\rangle \tag{3}$$

the MO's energies ϵ_i and expansion coefficients c_j can be obtained from solving the set of equations for $\mathbf{H}$:

$$\sum_{j=1}^{n} c_j(H_{ij} - \epsilon_i S_{ij}) = 0, \text{ where } i = 1, ..., n \tag{4}$$

Finding non-trivial solutions to these require that $\det(H_{ij} - \epsilon_i S_{ij}) = 0$, which is the same as diagonalizing **H** if **S** is the identity matrix.

1.2 Jacobi diagonalization

The method[1] is based on the iterative elimination of off-diagonal terms by applying two dimensional rotations. Let the matrix of interest **S** be symmetric and $\mathbf{G} = G(i, j, \theta)$ a Givens rotation matrix, which is defined as:

$$\mathbf{G} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & cos(\theta) & \cdots & -sin(\theta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & sin(\theta) & \cdots & cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \tag{5}$$

Where the sines are in the positions (i, j) and (j, i) of the matrix. This corresponds to the matrix representation of a rotation in the plane spanned by the ith and jth coordinate axes. Then:

$$\mathbf{S}' = \mathbf{G}\mathbf{S}\mathbf{G}^T \tag{6}$$

is also symmetric and similar to **S**. The transformed elements are:

$$\begin{align} S'_{ij} &= S'_{ji} = [cos^2(\theta) - sin^2(\theta)]S_{ij} + sin(\theta)cos(\theta)(S_{ii} - S_{jj}) \tag{7} \\ S'_{ii} &= cos^2(\theta)S_{ii} - 2sin(\theta)cos(\theta)S_{ij} + sin^2(\theta)S_{jj} \tag{8} \\ S'_{jj} &= sin^2(\theta)S_{ii} + 2sin(\theta)cos(\theta)S_{ij} + cos^2(\theta)S_{jj} \tag{9} \\ S'_{ik} &= S'_{ki} = cos(\theta)S_{ik} - sin(\theta)S_{jk} \tag{10} \\ S'_{jk} &= S'_{kj} = sin(\theta)S_{ik} + cos(\theta)S_{jk} \tag{11} \\ S'_{kl} &= S_{kl} \tag{12} \end{align}$$

where $k, l \neq i, j$. The objective if to eliminate off-diagonal terms, so θ is chosen such that $S'_{ij} = S'_{ji} = 0$, then from equation 7:

$$\begin{aligned} 0 &= \left[cos^2(\theta) - sin^2(\theta)\right] S_{ij} + sin(\theta)cos(\theta)(S_{ii} - S_{jj}) \\ &\Rightarrow \left[cos^2(\theta) - sin^2(\theta)\right] S_{ij} = sin(\theta)cos(\theta)(S_{jj} - S_{ii}) \\ &\Rightarrow \frac{cos^2(\theta)}{sin(\theta)cos(\theta)} \left[1 - tan^2(\theta)\right] = \frac{S_{jj} - S_{ii}}{S_{ij}} \\ &\Rightarrow \frac{1 - tan^2(\theta)}{tan(\theta)} = \frac{S_{jj} - S_{ii}}{S_{ij}} \\ &\Rightarrow tan^2(\theta) + \left(\frac{S_{jj} - S_{ii}}{S_{ij}}\right) tan(\theta) - 1 = 0 \end{aligned}$$

solving for $tan(\theta)$:

$$tan(\theta) = \frac{1}{\frac{S_{jj}-S_{ii}}{S_{ij}} \pm \sqrt{\left(\frac{S_{jj}-S_{ii}}{S_{ij}}\right)^2 + 1}} = \frac{1}{\sigma \pm \sqrt{\sigma^2+1}} \tag{13}$$

$$\text{where } \sigma = \frac{S_{jj} - S_{ii}}{S_{ij}} \tag{14}$$

of the two solution, the tangent with the lowest absolute value should be taken as it correspond to a smaller rotation, affecting less the rest of the terms. Therefore:

$$tan(\theta) = \frac{\Theta(\sigma)}{|\sigma| + \sqrt{\sigma^2+1}} \tag{15}$$

where Θ is the Heaviside theta function (sign function). Finally, the sine and cosine can be obtained from the tangent as:

$$cos(\theta) = \frac{1}{\sqrt{1 + tan^2(\theta)}} \tag{16}$$

$$sin(\theta) = tan(\theta)cos(\theta) \tag{17}$$

Applying this in equations 8-12 the rotated matrix is obtained. This procedure is sequentially done to all the off-diagonal terms of $\mathbf{S}$, which define a “scan”. At the end of each scan a error parameter is calculated, such as the sum of the squares of the off-diagonal elements:

$$\Delta^2 = \sum_{p=1}^{N} \sum_{q=1}^{p-1} (S'_{pq})^2 \tag{18}$$

The algorithm stops when a convergence criteria is meet, e. g., $\Delta < 10^{-8}$. The end result is an “almost” diagonal matrix, where the diagonal terms approximate its eigenvalues. To get the eigenvectors a matrix $\mathbf{U}$ is initially defined as the identity and, during the diagonalization, every Givens rotation is also applied to it as:

$$\mathbf{U}' = \mathbf{U} \cdot \mathbf{G} \tag{19}$$

With the eigenvalues, the total energy will be:

$$E = \sum_{i=1}^{n} N_i^{occ} \epsilon_i \tag{20}$$

where N_i^{occ} is the number of occupation of the ith molecular orbital. For closed shell systems this value is usually 2 or 0, given that the molecule is in its fundamental state, but it also depends on the number of electrons and total charge of the system.

2 The code

The code is divided into three modules where all subroutines are defined and a main program that brings them together. The first module is for two "tools", i. e., a subroutine to print matrices in a more readable format and another that sorts the eigenvectors with the Bubble sort method. The second module is used to set up the Huckel hamiltonian from a molecular geometry input file. It contains 5 subroutines: "read_xyz", "build_dist", "build_topol":, "build_hamiltonian" and "muliken". The third module is used to diagonalize an arbitrary symmetric matrix with the Jacobi method. It contains: "rotate":, "calculate_delta" and "diagonalize".

A pseudo-code description of the FORTRAN90 implementation of the above subroutines follows.

2.1 Read "xyz"

This subroutine opens the molecular geometry input file in the "xyz" format and read the atomic symbols and coordinates.

```
subroutine read_xyz(file_name, atoms, coords)
    ! inputs:
    file_name : character ! stores the name of the xyz input file

    ! outputs (allocatable):
    atoms : character, 1D array ! stores the symbols of the atoms
    coords : real, 2D array     ! stores the coordinates of the atoms

    ! local variables:
    i : integer        ! used to index loops
    un : integer       ! used to store the unit number of the input file
    n_atoms : integer ! used to store the number of atoms of the molecule

    open 'file_name' as 'un'
    read the first line into 'n_atoms'
    allocate 'atoms' with dimension 'n_atoms'
    allocate 'coords' with domensions 'n_atoms'x'n_atoms'

    do 'i' until 'n_atoms' ! for each line in the input
        read 'un' line : first letter into 'atoms' and rest into 'coords'
    end do
```

2.2 Build distance matrix

This subroutine calculates the distance (in angstrom) matrix of the molecules from the coordinates extracted from the geometry input file.

```
subroutine build_dist(coords, dist)
    ! inputs:
    coords : real, 2D array ! stores the atoms coordinates as [x, y, z]
    ! outputs (allocatable):
    dist : real, 2D array ! stores the distance matrix

    ! local variables:
    i, j : integer     ! used to index loops
    n_atoms : integer ! used to store the number of atoms in the input

    allocate 'dist' with dimensions 'n_atoms'x'n_atoms'
    initialize 'dist' as 0 ! for the diagonal terms

    do 'i' as 'coords' rows and 'j' as 'coords' columns ! just lower half
        ! the matrix is symmetric
        dist(i, j) = dist(j, i) = norm of the difference between ith and jth atoms
    end do

    return
end subroutine build_dist
```

2.3 Build topology matrix

This subroutine builds a topology matrix that describe the connections between the carbon atoms in the molecule according to the first neighbor approximation.

```
subroutine build_topol(atoms, dist, topol)
    ! inputs:
    atoms : character, 1D array ! stores the symbols of the atoms
    dist : real, 2D array       ! stores distance matrix

    ! outputs:
    topol : integer, 2D array ! stores the topology matrix

    ! local variables:
    i, j : integer     ! used to index loops
    n_atoms : integer ! used to store the number of atoms in the molecule
    n_C : integer     ! used to store the number of carbons in the molecule
    dist_C : real     ! defined as the maximum acceptable C—C bond distance
    idx_C : integer, 1D array ! used to store the index of the carbon atoms
                              ! in distance matrix

    do 'i' until 'n_atoms'
        if atoms(i) is a carbon : add to 'n_C' and keep index in 'idx_C'
    end do

    allocate 'topol' with dimensions 'n_C'x'n_C'
    initialize 'topol' as 0

    do 'i' as 'topol' rows and 'j' as 'topol' columns ! just lower half
        get the C—C distance in 'dist' with indexes from 'idx_C'
        if the distance allows a bond : topol(i, j) = (j, i) = 1
    end do

    return
end subroutine build_dist
```

2.4 Build Hamiltonian matrix

This subroutine build the Huckel hamiltonian matrix from the topology matrix following the definition in equation 1.

```
subroutine build_hamiltonian(topol, alpha, beta, H)
    ! inputs:
    topol : integer, 2D array ! stores the topology matrix
    alpha : real ! stores the 2pz orbital energy parameter
    beta : real  ! stores the interaction energy parameter

    ! outputs:
    H : real, 2D array ! stores the hamiltonian matrix

    ! local variables:
    i : integer ! used to index loops

    allocate 'H' with dimensions 'n_C'x'n_C'
    initialize 'H' as 'topol' with 'beta' in the non-zero entries

    do 'i' until 'n_C'
        define diagonal elements H(i, i) = 'alpha'
    end do

    return
end subroutine build_hamiltonian
```

2.5 Muliken analisys

This subroutine perforams a Muliken population analisys[2]. Given that the molecular orbitals are defined as linear combinations of the basis $\{|\phi_i^{2pz}\rangle\}_{i=1,\dots n}$, the density matrix $\mathbf{D}$ is calculated as:

$$|\psi_i\rangle = \sum_{j=1}^{n} c_{i,j}|\phi_j^{2pz}\rangle \tag{21}$$

$$\Rightarrow D_{ij} = \sum_{k=1}^{n} N_k^{occ} c_{k,i}c_{k,j} = \sum_{k=1}^{n} N_k^{occ} U_{ik}U_{jk} \tag{22}$$

then the population matrix $\mathbf{P}$ is obtained by:

$$\mathbf{P} = \mathbf{D}\cdot\mathbf{S} = \mathbf{D} \tag{23}$$

since the overlap matrix is the identity in this case. The diagonal terms refer to the $\pi-$electron atomic orbitals populations and off-diagonal to overlap populations, i. e., bond orders. Summing over the lines of $\mathbf{P}$ gives the gross orbital populations, which in this case corresponds to the total number of electrons in each carbon atom. The total charge in each atom is then computed subtracting the nucleus charge.

```
subroutine mulliken(U, n_occ, pop)
    ! inputs:
    U : real, 2D array          ! stores the eigenvectors matrix
    n_occ : integer, 1D array ! stores the MOs occupation numbers

    ! outputs:
```

```
    pop : real, 2D array ! stores the population matrix

    ! local variables:
    i, j : integer    ! used to index loops
    n_atoms : integer ! used to store the number of atoms in the molecule

    allocate 'pop' with dimensions 'n_atoms'x'n_atoms'
    initialize 'pop' as 0

    do 'i' and 'j' as the atomic obrital basis ! run through 'pop'
        pop(i, j) = pop(j, i) = summation over :
                                ('n_occ' * ith eigenvector * jth eigenvector)
        ! where the ith and jth eigenvectors are the lines in U
    end do

    return
end subroutine
```

2.6 Rotate matrix

This subroutine perform a Givens rotation to eliminate the target element of a matrix following the equations 8-11.

```
subroutine rotate(s, c, p, q, A, U)
    ! inputs:
    s : real    ! stores the value of the sine for the rotation
    c : real    ! stores the value of the cossine for the rotation
    p : integer ! stores the number of the row of the target element
    q : integer ! stores the number of the column of the target element

    ! (in)-outputs:
    A : real, 2D array ! stores the working matrix
    U : real, 2D array ! stores the eigenvectors matrix

    ! local variables:
    r : integer          ! used to index loops
    elp, elq, elr : real ! used to store old elements of the matrices

    define 'elp' and 'elq' as the old pth and qth diagonal elements of A
    calculate the new diagonal elements of A with 'elp' and 'elq'

    do 'r' for the rows of A
        if r not p or q : define 'elr' as the old value of A(r, p)
                          calculate the new off-diagonal elements with 'elr'
        end if
        define 'elr' as the old value of U(r, p)
        calculate all new elements of U with 'elr'
    end do

    return
end subroutine rotate
```

2.7 Calculate Δ

This subroutine calculates the convergence parameter Δ as defined in equation 18.

```
subroutine calculate_delta(A, delta)
    ! inputs:
    A : real, 2D array ! stores the working matrix

    ! outputs:
    delta : real ! stores the value of delta

    ! local variables:
    i, j : integer ! used to index loops

    initialize delta as 0

    do 'i' as 'A' rows and 'j' as 'A' columns ! just lower half
        add to delta the square of the (i, j) element of A
    end do

    recalculate 'delta' as its square root

    return
end subroutine rotate
```

2.8 Perform diagonalization

This subroutine performs the diagonalization following the Jacobi method described in section 1.2.

```
subroutine diagonalize(A, U, error)
    ! inputs:
    error : real ! stores the value of the error tolerance for convergence

    ! outputs:
    A : real, 2D array ! stores the working matrix
    U : real, 2D array ! stores the eigenvectors matrix

    ! local variables:
    s : real        ! stores the value of the sine for the rotation
    c : real        ! stores the value of the cossine for the rotation
    sigma : real    ! stores the value for the rotation parameter
    delta : real    ! stores the value of the convergence parameter
    t : real        ! stores the value of the tangent for the rotation
    p, q : integer  ! used to index loops
    n : integer     ! used to store the dimension of the matrix

    initialize 'U' as the identity and 'delta' as 1

    while 'delta' > 'error' do:
        do 'p' as 'A' rows and 'q' as 'A' columns ! just lower half -> scan
            skip the loop if the target (p, q) element os already zero
            calculate 'sigma'
            calculate 't', 'c' and 's' with 'sigma'
            update the matrix by calling the ‘rotate’ subroutine
            eliminate target elements : A(p, q) = A(q, p) = 0
        end do
        uptade delta by calling the ‘calculate_delta’ subroutine
    end do

    set all off−diagonal elements of A to zero ! return only the diagonal

    return
end subroutine rotate
```

2.9 Main program

This program systematizes the whole algorithm using the three mentioned modules and prints the results.

```
program main
    import the modules ‘tools’, ‘huckel’ and ‘jacobi’
    ! local variables:
    coords : real, 2D array       ! stores the atoms coordinates
    dist : real, 2D array         ! stores distance matrix
    topol : integer, 2D array     ! stores the topology matrix
    H : real, 2D array            ! stores the hamiltonian matrix
    U : real, 2D array            ! stores the eigenvectors matrix
    pop : real, 2D array          ! stores the population matrix
    n_occ : integer, 1D array     ! stores the MOs occupation numbers
    atoms : character, 1D array   ! stores the symbols of the atoms
    total_energy : real           ! stores the value of the total energy
    file_name : character         ! stores the name of the xyz input file
    charge : integer              ! stores the total charge of the system
    n_electrons : integer         ! stores the total number of electrons
    i, j : integer                ! used to index the loops

    ! parameters:
    alpha : real ! stores the 2pz orbital energy parameter
    beta : real  ! stores the interaction energy parameter
    error : real ! stores the value of the error tolerance for convergence

    read ‘file_name’ and ‘charge’ from the keyboard
    get ‘atoms’ and ‘coords’ by calling ‘read_input’

    ! build the system
    define ‘dist’ by calling ‘build_dist’ with ‘coords’
    define ‘topol’ by calling ‘build_topol’ with ‘dist’ and ‘atoms’
    define ‘H’ by calling ‘build_hamiltonian’ with ‘topol’, ‘alpha’ and ‘beta’

    ! solve
    diagonalize ‘H’ by calling ‘diagonalize’, ‘U’ is also computed
    sort ‘U’ by energy (eigenvalues) by calling ‘sort_eigenvalues’

    ! get occupations
    allocate ‘n_occ’ with the dimension of ‘H’ and initialize as 0
    get ‘n_electrons’ from the dimension of ‘H’ anf ‘charge’
    do ‘i’ until ‘n_electrons’
```

```
        define 'j' as the index of the current orbital
        add 1 to 'n_occ' of the orbital 'j' ! n_occ(j) ++
        if HOMO is degenerated (energy gap to next orbital is small):
            add 1 to 'n_occ' of the orbital 'j+1' ! n_occ(j+1) ++
        end if
    end do

    ! results
    calculate 'total_energy' with 'H' and 'n_occ'
    calculate 'pop' by calling 'mulliken' with 'H' and 'n_occ'

    return
end program main
```

3 Results

The systems chosen to test the code were: 1-3-butadiene, cyclopentadiene anion, benzene, naphthalene and pyrene. The energy-coefficient tables for the last two systems are in the appendix, as they are too big.

3.1 1-3-Butadiene

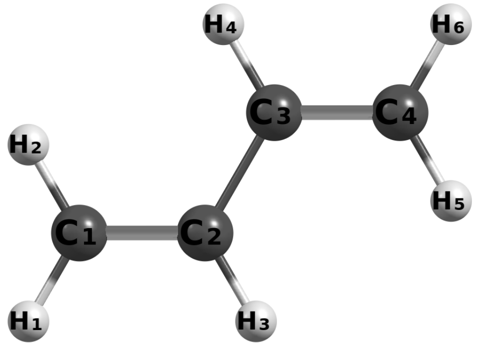

Figure 1: 1-3-butadiene's labeled molecular structure.

There is 4 $\pi-$electrons in this system, with a double occupation in the first two MOs, giving a total energy of:

$$E = -49.18eV$$

Table 1: Orbital energies and expansion coefficients of each of the MOs of 1-3-butadiene.

ψ_i	ϵ_i (eV)	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$
1	-12.69	0.37	0.60	0.60	0.37
2	-11.89	-0.60	-0.37	0.37	0.60
3	-10.91	0.60	-0.37	-0.37	0.60
4	-10.11	0.37	-0.60	0.60	-0.37

From table 1 its clear that the molecular orbitals energies follow the order of least nodes and the coefficients characterize the symmetries of the system, as expected. Moreover, table 2 shows that each atom has an electronic population of 1, while the π bonds present an order value of 0.89 and the σ bonds of 0.45. Negative values for bond orders don't have a physical meaning, but one could interpret it as the atom pair 1-4 not sharing a bond.

Table 2: Electron populations (diagonal) and bond orders (off-diagonal) of the atomic orbitals centered in the atoms of 1-3-butadiene.

Atom (i)	$\mathrm{BO}^{\pi}_{i,1}$	$\mathrm{BO}^{\pi}_{i,2}$	$\mathrm{BO}^{\pi}_{i,3}$	$\mathrm{BO}^{\pi}_{i,4}$
1	1.00	0.89	0.00	-0.45
2	0.89	1.00	0.45	-0.00
3	0.00	0.45	1.00	0.89
4	-0.45	-0.00	0.89	1.00

3.2 Cyclopentadiene anion

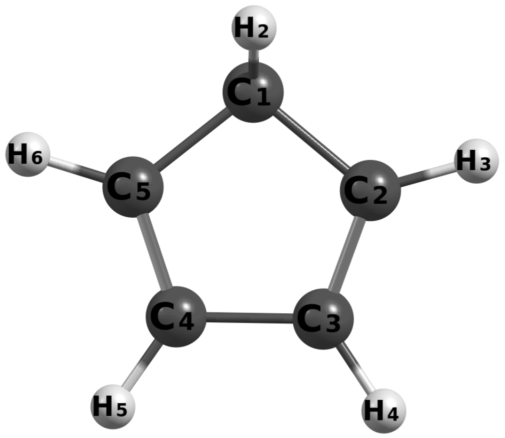

Figure 2: Cyclopentadiene's labeled molecular structure.

There is 6 $\pi-$electrons in this system (because of the negative charge) with a double occupation in the first three MOs, giving a total energy of:

$$E = -73.58eV$$

Table 3: Orbital energies and expansion coefficients of each of the MOs of cyclopentadiene anion.

ψ_i	ϵ_i (eV)	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$	$c_{i,5}$
1	-13.00	0.45	0.45	0.45	0.45	0.45
2	-11.89	-0.53	-0.50	0.22	0.63	0.17
3	-11.89	0.35	-0.39	-0.59	0.02	0.61
4	-10.11	0.23	0.16	-0.49	0.63	-0.53
5	-10.11	0.59	-0.61	0.40	-0.04	-0.34

Again, from table 3 its clear that the molecular orbitals energies follow the order of least nodes including the degeneracy introduced by the cyclic structure, but the symmetry is not well defined. Moreover, table 4 shows that each atom has an exaggerated electronic population of 1.2, which points out the anionic characteristic of the system, while every bond present an equal order value of 0.65 and no second neighbor carbons share a bond.

Table 4: Electron populations (diagonal) and bond orders (off-diagonal) of the atomic orbitals centered in the atoms of cyclopentadiene anion.

Atom (i)	$\mathrm{BO}^{\pi}_{i,1}$	$\mathrm{BO}^{\pi}_{i,2}$	$\mathrm{BO}^{\pi}_{i,3}$	$\mathrm{BO}^{\pi}_{i,4}$	$\mathrm{BO}^{\pi}_{i,5}$
1	1.20	0.65	-0.25	-0.25	0.65
2	0.65	1.20	0.65	-0.25	-0.25
3	-0.25	0.65	1.20	0.65	-0.25
4	-0.25	-0.25	0.65	1.20	0.65
5	0.65	-0.25	-0.25	0.65	1.20

3.3 Benzene

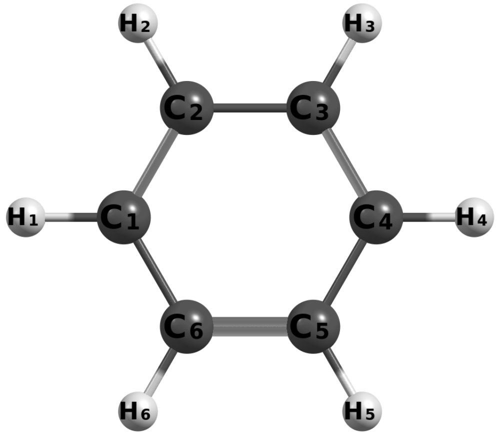

Figure 3: Benzene's labeled molecular structure.

There is 6 $\pi-$electrons in this system with a double occupation in the first three MOs, giving a total energy a little over that of the cyclopentadiene anion:

$$E = -74.80eV$$

Table 5: Orbital energies and expansion coefficients of each of the MOs of benzene.

ψ_i	ϵ_i (eV)	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$	$c_{i,5}$	$c_{i,6}$
1	-13.00	0.41	0.41	0.41	0.41	0.41	0.41
2	-12.20	0.00	-0.50	-0.50	-0.00	0.50	0.50
3	-12.20	-0.58	-0.29	0.29	0.58	0.29	-0.29
4	-10.60	0.54	-0.44	-0.10	0.54	-0.44	-0.10
5	-10.60	0.20	0.37	-0.57	0.20	0.37	-0.57
6	-9.80	0.41	-0.41	0.41	-0.41	0.41	-0.41

Table 5 shows that the molecular orbitals energies follow the order of least nodes including the degeneracy introduced by aromaticity and the coefficients reproduce the symmetry predicted by group theory. Moreover, table 6 shows the expected electronic population of 1 in each atom, but again every bond presents the same 0.67 bond order, suggesting an extra $\pi-$bond when compared to the usual Lewis structure. This can be attributed to the additional stabilization that results from the aromaticity of the system.

Table 6: Electron populations (diagonal) and bond orders (off-diagonal) of the atomic orbitals centered in the atoms of benzene.

Atom (i)	$BO^{\pi}_{i,1}$	$BO^{\pi}_{i,2}$	$BO^{\pi}_{i,3}$	$BO^{\pi}_{i,4}$	$BO^{\pi}_{i,5}$	$BO^{\pi}_{i,6}$
1	1.00	0.67	0.00	-0.33	-0.00	0.67
2	0.67	1.00	0.67	0.00	-0.33	-0.00
3	0.00	0.67	1.00	0.67	0.00	-0.33
4	-0.33	0.00	0.67	1.00	0.67	-0.00
5	-0.00	-0.33	0.00	0.67	1.00	0.67
6	0.67	-0.00	-0.33	-0.00	0.67	1.00

3.4 Naphthanele

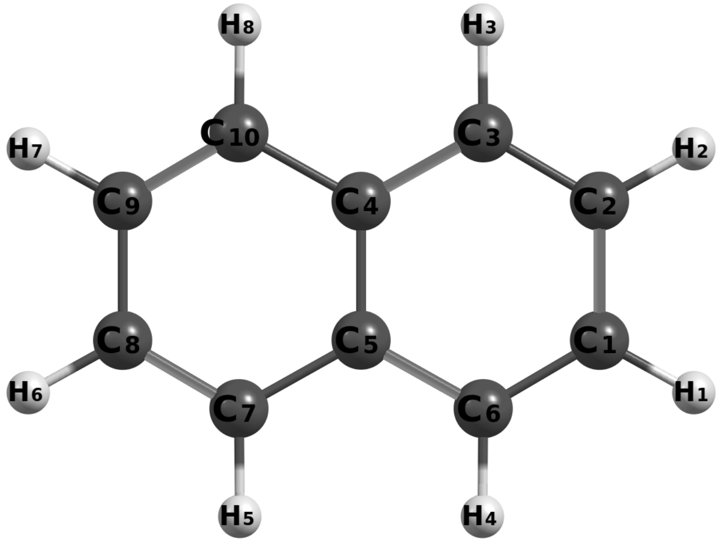

Figure 4: Naphthanele's labeled molecular structure.

There is 10 $\pi-$electrons in this system with a double occupation in the first three MOs, giving a total energy of:

$$E = -124.95eV$$

Table 7 shows no degeneracy in the molecular orbitals but the symmetries are preserved. Moreover, table 8 shows the expected electronic population of 1 in each atom and also the > 0.5 $\pi-$bond orders of, for example, the pairs 1-2 and 1-6. Unexpectedly, the pairs 1-9, 2-8, 2-7 and 6-10 also present bond orders greater than zero despite of their large separation in the structure. Finally, for the central atoms 4 and 5, the bond orders with their first neighbors are about the same (0.55), suggesting again the influence of aromaticity.

3.5 Pyrene

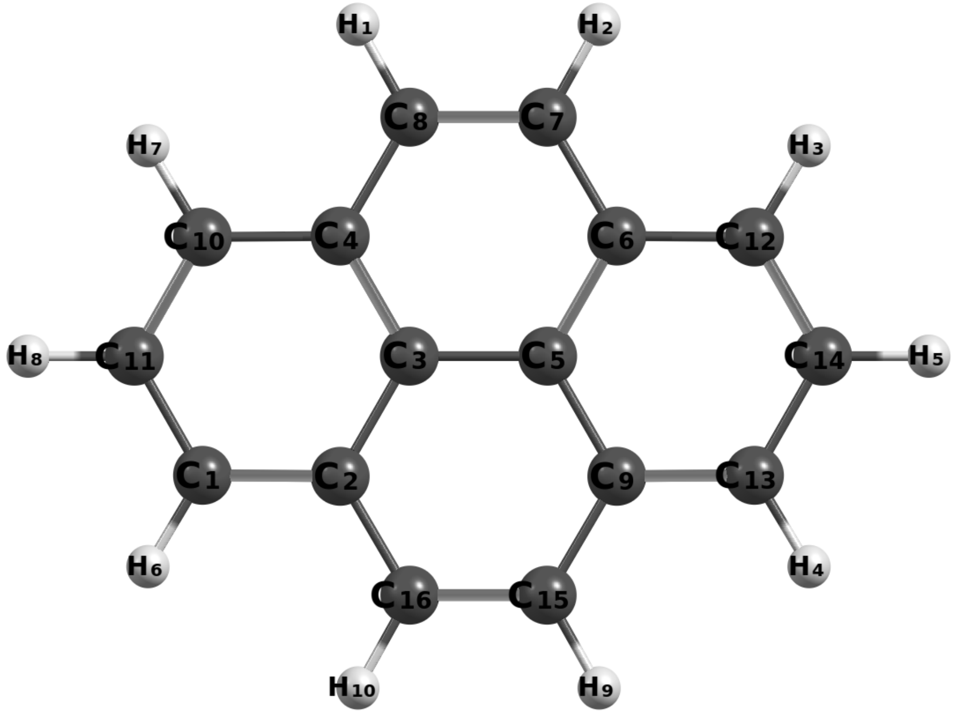

Figure 5: Pyrene's labeled molecular structure.

There is 16 $\pi-$electrons in this system with a double occupation in the first three MOs, giving a total energy of:

$$E = -200.41eV$$

From table 9 no molecular orbital is degenerated, but they seem to preserve the symmetries of the system. Moreover, table 10 shows again the electronic population of 1 in each atom and the bond orders above 0.5 in most of the neighboring atom pairs, as well as the small but positive bond orders of atoms in opposites sides of the molecule and the extra $\pi-$bonds of the central carbons.

References

[1] *Jacobi eigenvalue algorithm* (2023, 17 January). Wikipedia. https://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm

[2] *Huckel method* (2023, 22 February). Wikipedia. https://en.wikipedia.org/wiki/Huckel_method

Appendix

Table 7: Orbital energies and expansion coefficients of each of the MOs of naphthalene.

ψ_i	ϵ_i (eV)	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$	$c_{i,5}$	$c_{i,6}$	$c_{i,7}$	$c_{i,8}$	$c_{i,9}$	$c_{i,10}$
1	-13.24	0.23	0.23	0.30	0.46	0.46	0.30	0.30	0.23	0.23	0.30
2	-12.69	-0.43	-0.43	-0.26	-0.00	-0.00	-0.26	0.26	0.43	0.43	0.26
3	-12.44	0.17	-0.17	-0.40	-0.35	0.35	0.40	0.40	0.17	-0.17	-0.40
4	-12.20	0.41	0.41	0.00	-0.41	-0.41	-0.00	0.00	0.41	0.41	0.00
5	-11.89	0.26	-0.26	-0.43	-0.00	-0.00	0.43	-0.43	-0.26	0.26	0.43
6	-10.91	-0.26	-0.26	0.43	-0.00	-0.00	0.43	-0.43	0.26	0.26	-0.43
7	-10.60	0.41	-0.41	-0.00	0.41	-0.41	0.00	0.00	0.41	-0.41	-0.00
8	-10.36	0.17	0.17	-0.40	0.35	0.35	-0.40	-0.40	0.17	0.17	-0.40
9	-10.11	0.43	-0.43	0.26	0.00	-0.00	-0.26	0.26	-0.43	0.43	-0.26
10	-9.56	0.23	-0.23	0.30	-0.46	0.46	-0.30	-0.30	0.23	-0.23	0.30

Table 8: Electron populations (diagonal) and bond orders (off-diagonal) of the atomic orbitals centered in the atoms of naphthalene.

Atom (i)	$\mathrm{BO}^{\pi}_{i,1}$	$\mathrm{BO}^{\pi}_{i,2}$	$\mathrm{BO}^{\pi}_{i,3}$	$\mathrm{BO}^{\pi}_{i,4}$	$\mathrm{BO}^{\pi}_{i,5}$	$\mathrm{BO}^{\pi}_{i,6}$	$\mathrm{BO}^{\pi}_{i,7}$	$\mathrm{BO}^{\pi}_{i,8}$	$\mathrm{BO}^{\pi}_{i,9}$	$\mathrm{BO}^{\pi}_{i,10}$
1	1.00	0.60	0.00	-0.24	-0.00	0.72	-0.17	-0.00	0.16	0.00
2	0.60	1.00	0.72	-0.00	-0.24	-0.00	0.00	0.16	-0.00	-0.17
3	0.00	0.72	1.00	0.55	0.00	-0.36	0.08	-0.00	-0.17	-0.00
4	-0.24	-0.00	0.55	1.00	0.52	-0.00	-0.00	-0.24	0.00	0.55
5	-0.00	-0.24	0.00	0.52	1.00	0.55	0.55	0.00	-0.24	-0.00
6	0.72	-0.00	-0.36	-0.00	0.55	1.00	0.00	-0.17	0.00	0.08
7	-0.17	0.00	0.08	-0.00	0.55	0.00	1.00	0.72	0.00	-0.36
8	-0.00	0.16	-0.00	-0.24	0.00	-0.17	0.72	1.00	0.60	0.00
9	0.16	-0.00	-0.17	0.00	-0.24	0.00	0.00	0.60	1.00	0.72
10	0.00	-0.17	-0.00	0.55	-0.00	0.08	-0.36	0.00	0.72	1.00

Table 9: Orbital energies and expansion coefficients of each of the MOs of pyrene.

ψ_i	ϵ_i (eV)	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$	$c_{i,5}$	$c_{i,6}$	$c_{i,7}$	$c_{i,8}$	$c_{i,9}$	$c_{i,10}$	$c_{i1,1}$	$c_{i1,2}$	$c_{i1,3}$	$c_{i1,4}$	$c_{i1,5}$	$c_{i1,6}$
1	-13.43	0.17	0.30	0.40	0.30	0.40	0.30	0.20	0.20	0.30	0.17	0.14	0.17	0.17	0.14	0.20	0.20
2	-13.00	-0.30	-0.30	-0.20	-0.30	0.20	0.30	0.10	-0.10	0.30	-0.30	-0.30	0.30	0.30	0.30	0.10	-0.10
3	-12.84	-0.16	-0.30	-0.00	0.30	0.00	0.30	0.37	0.37	-0.30	0.16	-0.00	0.16	-0.16	0.00	-0.37	-0.37
4	-12.48	0.31	-0.04	-0.25	-0.04	-0.25	-0.04	-0.12	-0.12	-0.04	0.31	0.46	0.31	0.31	0.46	-0.12	-0.12
5	-12.40	-0.30	-0.37	0.00	0.37	0.00	-0.37	-0.16	0.16	0.37	0.30	-0.00	-0.30	0.30	-0.00	0.16	-0.16
6	-12.20	0.00	0.00	-0.41	-0.00	-0.41	-0.00	0.41	0.41	0.00	0.00	0.00	-0.00	-0.00	-0.00	0.41	0.41
7	-12.10	-0.19	0.26	0.28	0.26	-0.28	-0.26	-0.14	0.14	-0.26	-0.19	-0.42	0.19	0.19	0.42	-0.14	0.14
8	-11.76	-0.37	-0.16	-0.00	0.16	0.00	0.16	-0.30	-0.30	-0.16	0.37	0.00	0.37	-0.37	-0.00	0.30	0.30
9	-11.04	0.37	-0.16	-0.00	0.16	0.00	-0.16	-0.30	0.30	0.16	-0.37	0.00	0.37	-0.37	-0.00	0.30	-0.30
10	-10.70	-0.19	-0.26	0.28	-0.26	0.28	-0.26	0.14	0.14	-0.26	-0.19	0.42	-0.19	-0.19	0.42	0.14	0.14
11	-10.60	-0.00	0.00	-0.41	0.00	0.41	0.00	-0.41	0.41	-0.00	0.00	-0.00	-0.00	0.00	-0.00	-0.41	0.41
12	-10.40	0.30	-0.37	0.00	0.37	-0.00	0.37	-0.16	-0.16	-0.37	-0.30	0.00	-0.30	0.30	-0.00	0.16	0.16
13	-10.32	0.31	0.04	-0.25	0.04	0.25	-0.04	0.12	-0.12	-0.04	0.31	-0.46	-0.31	-0.31	0.46	0.12	-0.12
14	-9.96	-0.16	0.30	0.00	-0.30	0.00	0.30	-0.37	0.37	-0.30	0.16	0.00	-0.16	0.16	-0.00	0.37	-0.37
15	-9.80	0.30	-0.30	0.20	-0.30	0.20	-0.30	0.10	0.10	-0.30	0.30	-0.30	0.30	0.30	-0.30	0.10	0.10
16	-9.37	0.17	-0.30	0.40	-0.30	-0.40	0.30	-0.20	0.20	0.30	0.17	-0.14	-0.17	-0.17	0.14	-0.20	0.20

Continue below

Table 10: Electron populations (diagonal) and bond orders (off-diagonal) of the atomic orbitals centered in the atoms of pyrene.

Atom (i)	$BO^{\pi}_{i,1}$	$BO^{\pi}_{i,2}$	$BO^{\pi}_{i,3}$	$BO^{\pi}_{i,4}$	$BO^{\pi}_{i,5}$	$BO^{\pi}_{i,6}$	$BO^{\pi}_{i,7}$	$BO^{\pi}_{i,8}$	$BO^{\pi}_{i,9}$	$BO^{\pi}_{i,10}$	$BO^{\pi}_{i,11}$	$BO^{\pi}_{i,12}$	$BO^{\pi}_{i,13}$	$BO^{\pi}_{i,14}$	$BO^{\pi}_{i,15}$	$BO^{\pi}_{i,16}$
1	1.00	0.59	0.00	-0.28	-0.03	-0.00	0.18	-0.00	0.00	-0.00	0.67	-0.14	0.16	0.00	-0.21	-0.00
2	0.59	1.00	0.52	-0.00	0.00	-0.08	0.00	0.02	-0.17	-0.28	-0.00	0.00	-0.00	0.09	0.00	0.50
3	0.00	0.52	1.00	0.52	0.54	-0.00	-0.23	0.00	-0.00	-0.00	-0.23	-0.03	-0.03	0.00	-0.23	-0.00
4	-0.28	-0.00	0.52	1.00	0.00	-0.17	-0.00	0.50	-0.08	0.59	-0.00	-0.00	0.00	0.09	-0.00	0.02
5	-0.03	0.00	0.54	0.00	1.00	0.52	-0.00	-0.23	0.52	-0.03	-0.00	0.00	-0.00	-0.23	0.00	-0.23
6	-0.00	-0.08	-0.00	-0.17	0.52	1.00	0.50	-0.00	0.00	0.00	0.09	0.59	-0.28	-0.00	0.02	0.00
7	0.18	0.00	-0.23	-0.00	-0.00	0.50	1.00	0.78	0.02	-0.21	-0.00	-0.00	0.00	-0.12	0.00	-0.01
8	-0.00	0.02	0.00	0.50	-0.23	-0.00	0.78	1.00	0.00	0.00	-0.12	-0.21	0.18	0.00	-0.01	-0.00
9	0.00	-0.17	-0.00	-0.08	0.52	0.00	0.02	0.00	1.00	0.00	0.09	-0.28	0.59	-0.00	0.50	0.00
10	-0.00	-0.28	-0.00	0.59	-0.03	0.00	-0.21	0.00	0.00	1.00	0.67	0.16	-0.14	-0.00	0.18	0.00
11	0.67	-0.00	-0.23	-0.00	-0.00	0.09	-0.00	-0.12	0.09	0.67	1.00	0.00	-0.00	-0.07	-0.00	-0.12
12	-0.14	0.00	-0.03	-0.00	0.00	0.59	-0.00	-0.21	-0.28	0.16	0.00	1.00	0.00	0.67	-0.00	0.18
13	0.16	-0.00	-0.03	0.00	-0.00	-0.28	0.00	0.18	0.59	-0.14	-0.00	0.00	1.00	0.67	0.00	-0.21
14	0.00	0.09	0.00	0.09	-0.23	-0.00	-0.12	0.00	-0.00	-0.00	-0.07	0.67	0.67	1.00	-0.12	-0.00
15	-0.21	0.00	-0.23	-0.00	0.00	0.02	0.00	-0.01	0.50	0.18	-0.00	-0.00	0.00	-0.12	1.00	0.78
16	-0.00	0.50	-0.00	0.02	-0.23	0.00	-0.01	-0.00	0.00	0.00	-0.12	0.18	-0.21	-0.00	0.78	1.00

Continue below