# TASKS FOR THE INTENSIVE COURSE AND HOMEWORK

In thehe TC part of the intensive course you are supposed to work on some programming tasks under the supervision of both instructors, with approximately half of the time for Elena’s problems and the other half part for numerical algebra problems.

For the part of numerical algebra, your task is to write a series of programs implementing some of the basic algorithms discussed in the on-line sessions. The programs in this part will **not be evaluated**, they are supposed as a previous step in helping you to prepare the homework programs that are mandatory and evaluable.

Since the time in the intensive course is quite short, I recommend you to start working on the assigned programs as soon as possible so that in case of having doubts or problems you can ask me in person during the intensive course. Of course I will also be available before the intensive course for any doubts or help you may need. Send me a mail (p.alemany@ub.edu) and I will try to solve your problems via mail or on-line if necessary (Skype is a good option for me).

You are supposed to write your programs in FORTRAN. FORTRAN 90/95 is the best choice since you will have access to matrix and vector operations, but nevertheless, if you want to use FORTRAN 77, there is no problem with me.

# SYSTEMS OF LINEAR EQUATIONS

## 1) Gauss - Jordan algorithm

Write a program to find a solution of a linear system by the Gauss – Jordan algorithm. Concentrate on the n. of unknowns = n. of equations case.

You have the example I used in class in the following slides, check that your program does what it is supposed to do in each step by writing intermediate results.

Nevertheless use other examples (see for instance System of linear equations in Wikipedia) to check if your program works well.

The first step in the Gauss-Jordan algorithm is Gaussian elimination:

$$\begin{aligned} x+2y+z&=5\\ 3x+2y+4z&=17\\ 4x+4y+3z&=26 \end{aligned} \longrightarrow \begin{bmatrix} 1 & 2 & 1 & 5\\ 3 & 2 & 4 & 17\\ 4 & 4 & 3 & 26 \end{bmatrix} \xrightarrow{r_2 - 3r_1}$$

$$\begin{bmatrix} 1 & 2 & 1 & 5\\ 0 & -4 & 1 & 2\\ 4 & 4 & 3 & 26 \end{bmatrix} \xrightarrow{¼ * r_2} \begin{bmatrix} 1 & 2 & 1 & 5\\ 0 & 1 & -\frac{1}{4} & -\frac{1}{2}\\ 4 & 4 & 3 & 26 \end{bmatrix} \xrightarrow{r_3 - 4r_1} \begin{bmatrix} 1 & 2 & 1 & 5\\ 0 & 1 & -\frac{1}{4} & -\frac{1}{2}\\ 0 & -4 & -1 & 6 \end{bmatrix} \xrightarrow{r_3 + 4r_2}$$

$$\begin{bmatrix} 1 & 2 & 1 & 5\\ 0 & 1 & -\frac{1}{4} & -\frac{1}{2}\\ 0 & 0 & -2 & 4 \end{bmatrix} \xrightarrow{-½ * r_3} \begin{bmatrix} 1 & 2 & 1 & 5\\ 0 & 1 & -\frac{1}{4} & -\frac{1}{2}\\ 0 & 0 & 1 & -2 \end{bmatrix} \longrightarrow \begin{aligned} x+2y+z &= 5\\ y-\frac{1}{4}z &= -\frac{1}{2}\\ z &= -2 \end{aligned}$$

The second step is the backward phase where we introduce 0 in the columns above the pivots

$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 1 & -\frac{1}{4} & -\frac{1}{2} \\ 0 & 0 & 1 & -2 \end{bmatrix} \xrightarrow{r_2+4r_3} \begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \xrightarrow{r_1-r_3} \begin{bmatrix} 1 & 2 & 0 & 7 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \xrightarrow{r_1-2r_2}$$

$$\begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \Rightarrow \begin{aligned} x &= 9 \\ y &= -1 \\ z &= -2 \end{aligned}$$

## 2) Matrix inversion via Gaussian elimination

Write a program to invert a square $n \times n$ matrix using the inversion algorithm.

To find the inverse of a $n \times n$ matrix $\mathbf{A}$ we create a partitioned $n \times 2n$ matrix $[\mathbf{A} \mid \mathbf{I_n}]$ and perform elementary row operations until the left side is reduced to $\mathbf{I_n}$. Those operations will produce $\mathbf{A}^{-1}$ in the right side of the expanded matrix that will turn into $[\mathbf{I_n} \mid \mathbf{A}^{-1}]$:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 3 \\ 1 & 0 & 8 \end{bmatrix} \Rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 5 & 3 & 0 & 1 & 0 \\ 1 & 0 & 8 & 0 & 0 & 1 \end{array}\right] \xrightarrow{\substack{r_2-2r_1 \\ r_3-r_1}}$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & -2 & 5 & -1 & 0 & 1 \end{array}\right] \xrightarrow{r_3+2r_2} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & -1 & -5 & 2 & 1 \end{array}\right] \xrightarrow{-1r_3}$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right] \xrightarrow{\substack{r_2+3r_3 \\ r_1-3r_3}} \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -14 & 6 & 3 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right] \xrightarrow{r_1-2r_1}$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -40 & 16 & 9 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right] \Rightarrow A^{-1} = \begin{bmatrix} -40 & 16 & 9 \\ 13 & -5 & -3 \\ 5 & -2 & -1 \end{bmatrix}$$

## 3) Determinant via LU-decomposition

Write a program that uses a LU decomposition subroutine to calculate the determinant of a square $n \times n$ matrix.

## Example on how to build L during the Gaussian elimination process:

$$\mathbf{A} = \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \qquad \begin{bmatrix} \bullet & 0 & 0 \\ \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} \color{red}{1} & -1/3 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \xleftrightarrow{\text{multiplier = 1/6}} \begin{bmatrix} 6 & 0 & 0 \\ \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} \color{red}{1} & -1/3 & 0 \\ \color{red}{0} & 2 & 1 \\ \color{red}{0} & 8 & 5 \end{bmatrix} \begin{matrix} \xleftrightarrow{\text{multiplier = -9}} \\ \xleftrightarrow{\text{multiplier = -3}} \end{matrix} \begin{bmatrix} 6 & 0 & 0 \\ 9 & \bullet & 0 \\ 3 & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} \color{red}{1} & -1/3 & 0 \\ \color{red}{0} & \color{red}{1} & 1/2 \\ \color{red}{0} & 8 & 5 \end{bmatrix} \xleftrightarrow{\text{multiplier = 1/2}} \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} \color{red}{1} & -1/3 & 0 \\ \color{red}{0} & \color{red}{1} & 1/2 \\ \color{red}{0} & \color{red}{0} & \color{red}{1} \end{bmatrix} \xleftrightarrow{\text{multiplier = -8}} \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & \bullet \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} \color{red}{1} & -1/3 & 0 \\ \color{red}{0} & \color{red}{1} & 1/2 \\ \color{red}{0} & \color{red}{0} & \color{red}{1} \end{bmatrix} \xleftrightarrow{\text{multiplier = 1}} \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & 1 \end{bmatrix} = \mathbf{L}$$

$$\Rightarrow$$

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1/3 & 0 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \mathbf{LU} \end{aligned}$$

When you are finished, check that A is actually A = LU

To compute the determinant you simply need to remember that:

1) If $\mathbf{B}$ is a $n \times n$ matrix, then $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$

2 ) If $\mathbf{A}$ is triangular, then $\det(\mathbf{A}) = \prod_{i=1}^{n} a_{ii}$

## EIGENVALUE PROBLEMS

## 4) Power method with Euclidean scaling

Write a program to find the dominant eigenvalue and a unit dominant eigenvector for a symmetric $n \times n$ matrix.

Use these data to check that your program is working:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ -2 & 1 & 2 \\ 1 & 3 & 1 \end{bmatrix}.$$

Use $\mathbf{x}_0 = (1, 1, 1)$ as the initial approximation.

Solution: x = (½, ½, 1) with eigenvalue λ = 3

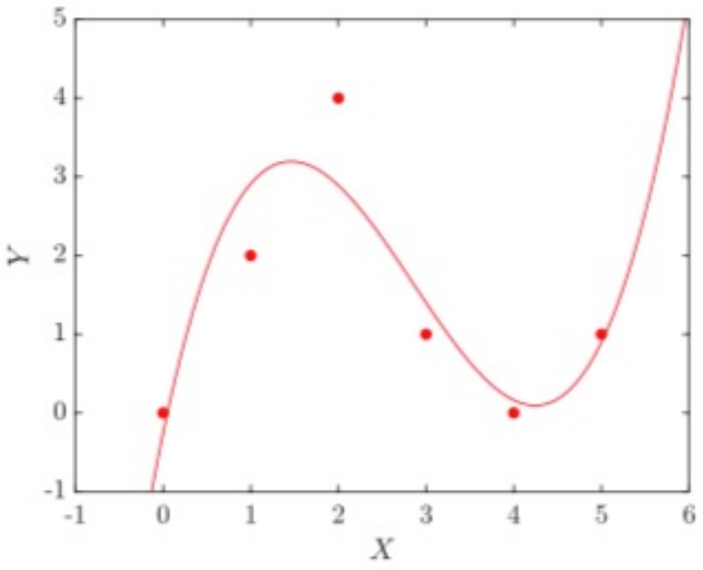
## HOMEWORK PROBLEMS

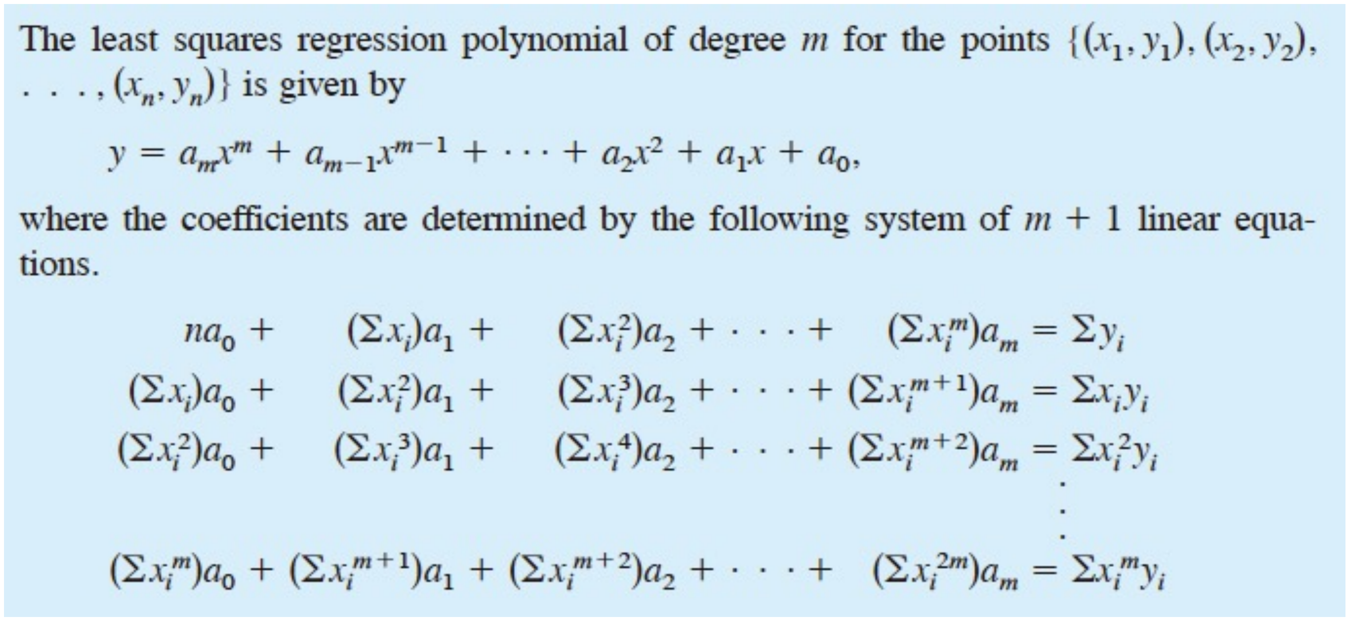
For homework you will have to write two programs, one on linear systems of equations and the other one on eigenvalue problems. You can find detailed explanations in a separate document that you will find in the campus.

Please take note that you will not only deliver the FORTRAN source codes as your homework. I ask you to write a short report for each of the two problems where you are supposed to write an introduction with the theoretical basis, a description in pseudocode of your program, the actual program written in an easy to follow style including comments explaining the principal features, examples of running the programs with the data I give you, and a final reference list. All this (except the program) should be included in a single PDF file.

Note that I will grade your excercises taking all into account. Just writing a program that works correctly does not automatically give you a 10 mark. Please be careful with the formal aspects of your report (include your name, check the spelling, include figure captions, ...). Please, do not just copy-paste a bunch of screen shots with a nice cover page!

## 1) Polynomial regression analysis




The least squares regression polynomial of degree $m$ for the points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is given by

$$y = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_2x^2 + a_1x + a_0,$$

where the coefficients are determined by the following system of $m + 1$ linear equations.

$$\begin{aligned}
na_0 + (\Sigma x_i)a_1 + (\Sigma x_i^2)a_2 + \cdots + (\Sigma x_i^m)a_m &= \Sigma y_i \\
(\Sigma x_i)a_0 + (\Sigma x_i^2)a_1 + (\Sigma x_i^3)a_2 + \cdots + (\Sigma x_i^{m+1})a_m &= \Sigma x_i y_i \\
(\Sigma x_i^2)a_0 + (\Sigma x_i^3)a_1 + (\Sigma x_i^4)a_2 + \cdots + (\Sigma x_i^{m+2})a_m &= \Sigma x_i^2 y_i \\
&\vdots \\
(\Sigma x_i^m)a_0 + (\Sigma x_i^{m+1})a_1 + (\Sigma x_i^{m+2})a_2 + \cdots + (\Sigma x_i^{2m})a_m &= \Sigma x_i^m y_i
\end{aligned}$$

Check if your program is working well using an alternative program (excel for instance) to do the regression

## 2) Eigenvalues and eigenvectors for the Hückel hamiltonian

The Hückel method is a very crude approximate method to find $\pi$-type MOs and their energies for planar conjugated hydrocarbons using the LCAO approximation:

$$\psi_g = N(c_1\phi_1 + \cdots + c_n\phi_n)$$

$$\sum_{j=1}^{n} c_j(H_{ij} - ES_{ij}) = 0 \quad (i = 1, \cdots, n).$$

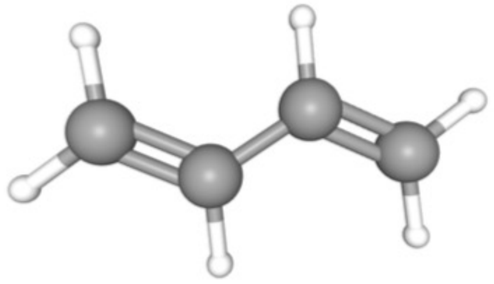
$$\det([H_{ij} - ES_{ij}]) = 0$$

Where (Hückel approximation):

$$S_{ij} = \delta_{ij}$$

$$H_{ij} = \begin{cases} \alpha, & i = j; \\ \beta, & i, j \text{ adjacent}; \\ 0, & \text{otherwise.} \end{cases}$$

Your task is to program the Jacobi algorithm to find the eigenvalues E and eigenvectors of matrix H. Remember that to find the coefficients for the orbitals you will need an extra step to normalize the eigenvectors. You should also compute net charges on atoms from the coefficients (Mulliken population analysis).



Butadiene

$$\mathbf{H} = \begin{bmatrix} \alpha & \beta & 0 & 0 \\ \beta & \alpha & \beta & 0 \\ 0 & \beta & \alpha & \beta \\ 0 & 0 & \beta & \alpha \end{bmatrix}.$$

$$E_{1,2,3,4} \approx \alpha + 1.618\beta, \alpha + 0.618\beta, \alpha - 0.618\beta, \alpha - 1.618\beta\,,$$

$$\Psi_1 \approx 0.372\phi_1 + 0.602\phi_2 + 0.602\phi_3 + 0.372\phi_4$$
$$\Psi_2 \approx 0.602\phi_1 + 0.372\phi_2 - 0.372\phi_3 - 0.602\phi_4$$
$$\Psi_3 \approx 0.602\phi_1 - 0.372\phi_2 - 0.372\phi_3 + 0.602\phi_4$$
$$\Psi_4 \approx 0.372\phi_1 - 0.602\phi_2 + 0.602\phi_3 - 0.372\phi_4$$