Learning objectives of the subject

Computers and software for mathematical numerical analysis; Numbers, algorithms and error analysis; Determining the root of a function; Solving systems of equations by means of direct numerical methods and basic iterative methods;
Solving nonlinear systems of equations; Estimation and interpolation; Numerical quadrature; Solving ordinary differential equations.

Upon completion of the course, students will have acquired the ability to: 1. Use standard computer tools to solve basic problems (e.g. measurements). 2. Use numerical analysis software to carry out a sensitivity analysis of a problem in which an ordinary differential equation is solved. 3. Solve engineering problems using numerical techniques.

Computers and software for mathematical numerical analysis; Numbers, algorithms and error analysis; Determining the root of a function; Solving systems of equations by means of direct numerical methods and basic iterative methods; Solving nonlinear systems of equations; Estimation and interpolation; Numerical quadrature; Solving ordinary differential equations.

Basics manual and computer programming and knowledge utilitzarr numerical models in engineering problems. Ability to interpret the results provided by the models in the context of engineering.

By the end of the course, the student will be able to:
1. Use standard tools to solve basic problems (eg steps).
2. Systematize and codify a repetitive task in a more or less complex computer program.
3. Use the most appropriate method to approximate empirical data or design.
4. Identify and solve engineering problems as problems of integration, functions and zeros of ordinary differential equations.


### Study load

<table>
<thead>
<tr>
<th>Total learning time: 187h 30m</th>
<th>Theory classes: 18h 30m</th>
<th>9.87%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practical classes: 14h</td>
<td></td>
<td>7.47%</td>
</tr>
<tr>
<td>Laboratory classes: 42h 30m</td>
<td></td>
<td>22.67%</td>
</tr>
<tr>
<td>Guided activities: 7h 30m</td>
<td></td>
<td>4.00%</td>
</tr>
<tr>
<td>Self study: 105h</td>
<td></td>
<td>56.00%</td>
</tr>
</tbody>
</table>
## Content

### Introduction to Programming

<table>
<thead>
<tr>
<th>Learning time: 36h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes: 4h</td>
</tr>
<tr>
<td>Practical classes: 3h</td>
</tr>
<tr>
<td>Laboratory classes: 8h</td>
</tr>
<tr>
<td>Self study : 21h</td>
</tr>
</tbody>
</table>

**Description:**
Introduction to programming as a tool to solve problems throughout the course and as a tool to increase productivity and technical training in other academic and professional career. Specifically, an overview of the capabilities of Excel and programming language Visual Basic for Applications (VBA) and some skill in their use is given. This programming language has the advantage of being installed on any computer with the Windows operating system and the Excel program in relatively recent versions.

The Excel environment is approached at the user level. The goal is to learn what is controllable through the interface and what the basic controls are, how to define functions, plot graphs and, more interestingly, what macros are and how they will be used over the course.

**VBA: Variables and information flow**

**Guided work:** variables and information flow in VBA

**Conditional statement and loops**

**Guided work:** conditional statements and loops

### Programming Examples

**Description:**

**Specific objectives:**
- Understand the work environment of Excel and VBA
- See through examples some of the possibilities offered by the program.
- Use a spreadsheet as data storage.
- Assign formulas through the cells.
- Automatically create arrays by dragging and calculation.
- Draw the graph of a function and understand the difference between a line chart and a scatter chart.
- Understand what a call to a function is and be able to use input and output parameters correctly.
- Understand what a macro is and its potential use.
- Understand what is a programming language such as VBA.
- Know the difference between a subprocedure and a function, and be able to decide which one to use.
- Understand the sequential nature of a VBA program.
- Know the different types of variables that can be used and understand how they are declared and initialized.
- Be able to create subprocedures and functions.
- Know when to call subprocedures or functions.
- Create a function and be able to call it either from a cell in the spreadsheet or from a subprocedure.
- Be able to use variables and assign values to them.
- Write and read data from the spreadsheet.

Understanding what a control statement is and how it works: conditional (if) and loops (for, while).

- Understand the use of a conditional statement and be able to use it.
- Understand what a loop is and its programming possibilities.
- Be able to use a For and a While loop.
- Be able to write a code that computes a summation or any variant.
- Understand the possibilities of connecting loops and conditionals.
- Revise the programming concepts covered in previous sessions.
- Be able to code simple algorithms
- Be able to recognize and correct common programming errors
**Errors**

**Description:**
This topic is theoretical, but it has practical implications in the proper use of numerical algorithms and interpretation of results.


**Examples**

**Specific objectives:**
- Understand that numerical calculations always give approximate results, subject to error.
- Understand the sources of errors and their propagation in basic operations.
- Understand the concept of the error bound as a tool to control bugs and keep them small.
- Understand how a digital computer stores numbers, and the implications and limitations that this causes.
- Be able to identify operations that are unfavorable from the point of view of propagation and amplification of errors.

Use illustrative examples to:
- Understand that numerical calculations are subject to errors.
- Identify operations that spread errors, yielding poor results.
- Be able to propose a reasonable order of accuracy in the results.

<table>
<thead>
<tr>
<th><strong>Learning time:</strong></th>
<th>7h 11m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes:</td>
<td>2h</td>
</tr>
<tr>
<td>Laboratory classes:</td>
<td>1h</td>
</tr>
<tr>
<td>Self study:</td>
<td>4h 11m</td>
</tr>
</tbody>
</table>
# Root finding

**Learning time:** 16h 48m  
Theory classes: 2h  
Practical classes: 2h  
Laboratory classes: 3h  
Self study: 9h 48m

## Description:
Finding roots of algebraic equations is a fundamental numerical problem, as in many cases of interest it is not possible or practical to solve analytically. This topic lets us introducing general concepts in numerical methods, such as iterative algorithms or numerical tolerance.


Guided work: root finding  
Problems of root finding

## Specific objectives:
- Understand the mathematical problem (solving non-linear equations) and its relation to some practical problems.  
- Understand the architecture of a generic iterative algorithm.  
- Understand the convergence criteria and be able to reasonably choose numerical tolerances.  
- Understand the theoretical foundations and implementation of three methods listed in the course program (bisection, Newton and secant).  
- Understand the concept of order convergence and be able to obtain it numerically.

- Code in VBA the iterative methods for solving equations described in the course.  
- Experiment with the methods.  
- Be able to obtain convergence plots.  
- Be able to use the iterative algorithms described in the course to solve simple equations.  
- Be able to calculate the error of the approximations and decide when a result is good enough.
### Numerical Integration

**Learning time:** 16h 48m  
Theory classes: 2h  
Practical classes: 2h  
Laboratory classes: 3h  
Self study: 9h 48m

**Description:**  

Guided work: numerical integration  
Problems of numerical integration

**Specific objectives:**  
- Understand the mathematical problem and how it is related to some practical problems.  
- Understand the theoretical foundations and implementation of the three methods listed in the program (rectangular approximations, composite trapezoidal rule and composite Simpson's rule).  
- Understand the concept of convergence rate and be able to obtain it numerically.  
- Code in VBA the composite trapezoidal rule and the composite Simpson rule.  
- Experimentally obtain convergence plots and verify the formulas to estimate the error.  
- Use composite quadratures to approximate numerically the value of a definite integral.  
- Be able to estimate the error of an approximation and use this result to decide which discretization is needed to ensure a given accuracy.

### Systems of linear equations

**Learning time:** 9h 36m  
Theory classes: 2h  
Laboratory classes: 2h  
Self study: 5h 36m

**Description:**  
- Systems with immediate solution: diagonal matrix, triangular matrix.  
- Direct methods. Gauss.  
Guided work: matrices and systems of linear equations

**Specific objectives:**  
- Understand the mathematical problem and its relation to some practical problems.  
- Understand which characteristics of the methods for solving linear systems are interesting from a numerical point of view.  
- Understand how the Gauss method works.  
- Declare a static or dynamic container. In particular, a numerical container as a vector of dimension one or a matrix of dimension two.  
- Read and write vectors and matrices in an Excel spreadsheet using containers.  
- Assign values to the coefficients of a matrix or a vector.  
- Use subprocedures and functions that work with matrices and vectors.  
- Perform calculations using containers, such as a matrix times vector product.  
- Use containers in solving linear systems.
### Evaluation # 1

**Description:**
Resolution of assessment # 1

**Learning time:** 7h 11m
- Theory classes: 1h
- Laboratory classes: 2h
- Self study: 4h 11m

### Assessment # 2

**Description:**
Resolution reviews

**Learning time:** 7h 11m
- Practical classes: 1h
- Laboratory classes: 2h
- Self study: 4h 11m
Functional Approximation

Description:
- Interpolation
- Guided work: polynomial interpolation
- Problems of interpolation
- Least squares
- Guided work: least squares
- Problems of least squares approximation
- Splines
- Guided work: splines
- Problems splines
- Presentation of the work conducted functional approach

Specific objectives:

Learning time: 31h 12m
- Theory classes: 3h 30m
- Practical classes: 4h 30m
- Laboratory classes: 5h
- Self study: 18h 12m
- Understand the concept of interpolation
- Be able to calculate the polynomial that interpolates certain data
- Understand the approximation criteria for interpolation
- Be able to compute the polynomial that passes through some given points
- Programming in VBA:
  - Revise how to work with arrays
  - Revise how to assign values to the coefficients of a matrix or a vector.
  - Recall how to use functions and subprocedures and, in particular, how to solve linear systems of equations.
- Be able to calculate the polynomial that interpolates certain data
- Understand the properties of polynomial interpolator (passes through two points) and what problems can this approach have.
- Understand the criterion of least squares approximation.
- Be able to state linear and non-linear least squares approximations.
- Be able to solve linear least-squares problems and, in particular, be able to compute the polynomial that approximates some given data by a least-squares criterion.
- Write a VBA program to calculate the polynomial that approximates some given data using a least squares criterion.
- Learn to use some Excel tools to perform least-squares fittings:
  - "trend line" in a scatter chart
  - "Solver", which allows solving non-linear problems.
- Be able to calculate the polynomial that approximates some data according to the least squares criterion.
- Be able to solve linear least squares problems
- Be able to linearize some simple expressions to carry out parameter fitting by solving a linear least squares problem.
- Understand the definition of spline
- Understand how to choose the type of spline (degree of the polynomial approximation and smoothness).
- Be able to decide how many data are needed to calculate a spline (e.g., linear and cubic C0 C1).
- Be able to approximate derivatives of a function using its values at some given points.
- Understand the definition of spline
- Understand the development of C1 cubic splines and know them to schedule
- Programming in VBA:
  - Review how to work with vectors and matrices
  - Recall how to work with subprocedures and functions
- Be able to compute the C0 linear spline that interpolates some data.
- Be able to compute the C1 cubic spline that interpolates some data.
- Be able to approximate the value of the derivative of a function using its value at some points.
- Be able to verify the continuity of a spline.
- Understand when and why it is necessary to approximate functions
- Understand the different criteria for functional approximation (interpolation and least squares) and be able to decide when it is convenient to use one or the other.
Modeling with ODEs

**Description:**
- First order ODEs.
- Higher order ODEs: reduction to a system of first order ODEs.
- Initial value problems and boundary value problems.
- Resolution of initial value problems using the Euler method, the modified Euler method and the method of Heun.
- Convergence rate of the methods.

**Guided work:** solving ODEs
**Problems:** solving ODEs

**Specific objectives:**
- Understand the mathematical problem (which is an initial value problem) and its relation to some practical problems.
- Understand how to approximate numerically the solution of initial value problem.
- Understand the theoretical foundations and implementation of three methods listed in the course program (Euler, modified Euler and Heun).
- Be able to solve first order ODEs using the methods listed in the program.
- Understand how to measure the error in solving an ODE.
- Understand the concept of order convergence and get to know them numerically.
- Understand the difference between an initial value problem and a boundary value problem, and be able to decide which numerical methods have to be used to solve them.
- Understand and implement methods for solving ODEs: Euler, modified Euler and Heun.
- Understand the concept of convergence rate and be able to obtain it numerically.
- Be able to use the methods described in the program (Euler, modified Euler and Heun) to solve numerically initial value problems.
- Be able to, given the error of approximation, choose a discretization that ensures a given accuracy in the results.

**Learning time:** 19h 12m
- Theory classes: 1h 30m
- Practical classes: 1h 30m
- Laboratory classes: 5h
- Self study: 11h 12m

Modelling

**Description:**
Engineering design problem that can be written as a root-finding problem, an integration problem, an ODE or an approximation problem. Comprehends all the topics covered in the course, including programming.

**Modelling work conducted**

**Specific objectives:**
- Identify and solve engineering problems as a numerical problems.
- Choose and apply different numerical techniques that have been covered during the course.

**Learning time:** 16h 48m
- Practical classes: 3h
- Laboratory classes: 4h
- Self study: 9h 48m
The final grade is obtained from different assessments as
\[ N = 0.2 \times E + 0.2 \times TC + 0.3 \times P + 0.3 \times Ex \]
where
- \( E \) is the arithmetic average of the exercises done in class, including problems and practices
- \( TP \) is a programming assignment
- \( TC \) is a course assignment, in which numerical methods are used to solve an engineering problem
- \( P \) is the mark on programming, computed as the geometric mean of a programen assignment (TP) and a programming test (ExP). This is, \( P = TP^{(1/2)} \times ExP^{(1/2)} \)
- \( Ex \) is the mark of individual tests (E1 mid-term, E2 final). This mark is obtained as the maximum between \( 0.3 \times E1 + 0.7 \times E2 \) and \( E2 \).

All marks are out of 10 and an average mark \( N \) equal or higher than 5 is needed to pass the course.

In order to be evaluated, exercises and assignments must be submitted in due time.

Criteria for re-evaluation qualification and eligibility: Students that failed the ordinary evaluation and have regularly attended all evaluation tests will have the opportunity of carrying out a re-evaluation test during the period specified in the academic calendar. Students who have already passed the test or were qualified as non-attending will not be admitted to the re-evaluation test. The maximum mark for the re-evaluation exam will be five out of ten (5.0). The non-attendance of a student to the re-evaluation test, in the date specified will not grant access to further re-evaluation tests.

Students unable to attend any of the continuous assessment tests due to certifiable force majeure will be ensured extraordinary evaluation periods.
These tests must be authorized by the corresponding Head of Studies, at the request of the professor responsible for the course, and will be carried out within the corresponding academic period.
Regulations for carrying out activities

Failure to perform a laboratory or continuous assessment activity in the scheduled period will result in a mark of zero in that activity.

Some of the course assessments can be done in groups and, therefore, it is allowed to comment with other students. However, it is not allowed to copy results nor plagiarize parts of the code or reports. The cases of copying will be punished.

Students must attend examinations with a working calculator. Cell phones, computers, tablets or other electronic devices are not allowed. However, it is possible that the use of books and notes is allowed in some parts of the exams.

Bibliography

Basic:
