I. Numerical Computing

A. Lectures 1-3: Foundations of Numerical Computing

Things you should know

Lecture 1 – Intro to numerical computing

- Understand difference and pros/cons of analytical versus numerical solutions

Lecture 2 – Introduction to Excel and Python

- The four basic elements of a computer: input, output, CPU, memory
- What a spreadsheet is
- What a programming language is

Lecture 3 – Units, Data Types, Error

- Concept of dimensional homogeneity
- Data types: boolean, integer, float, string
- How data types are stored in memory
- Understand the origin of finite precision errors

Things you should be able to do

Lecture 1 – Intro to numerical computing

- Classify math problems: linear/nonlinear, algebraic/differential, single/system, coupled/uncoupled

Lecture 2 – Introduction to Excel and Python

- Excel: navigation, copy/paste/fill, formatting, arithmetic, formulas, locked cells
- Python: console I/O, define variables, arithmetic, use comments
- Use Excel and Python help docs

Lecture 3 – Units, Data Types, Error

- Conversions between "Big 13" units (kg, m, s, N, lbm, ft, hr, lbf, slug, K, °C, °F, °R)
- Put an equation in consistent units for use in a numerical tool
- Define variables of different data types in Excel and Python

B. Lectures 4-5: Structured Programming

Things you should know

Lecture 4 – Functions

- What a function is (map, formula, procedure, abstraction) and when it is useful (name, organize, re-use code)
- Difference between local and global variable scope

Lecture 5 – Conditional Statements

- What a conditional (`if`, `if...else`) statement is and when it is useful (branch code, make decisions)
- The difference between the comparison (`==`) and the assignment operator (`=`)

Things you should be able to do

Lecture 4 – Functions

- Call a function in Excel
- Import modules in Python
- Call a function in Python
- Define a function in Python (with or without arguments, with or without return statements)
- Use local and global variables

Lecture 5 – Conditional Statements

- Use conditional and logical functions in Excel
- Use conditional (`if`, `if...else`) statements, logical (`and`, `or`, `not`) operators and comparison operators (`==`, `>`, `<`, ...) in Python

C. Lectures 6-9: Loops and Arrays

Things you should know

Lecture 6 – Loops

- What a loop is and when it is useful (repeat code, fill arrays, ...)
- What an infinite loop is and why it occurs

Lecture 7 – Arrays

- How an array relates to a vector and a matrix
- What an array variable is ("ice cube tray")
- How Excel stores arrays
- Difference between a Python list, Python tuple

and Numpy array

Lecture 8 – File I/O and Plotting

- How text files are formatted (header, delimiter, values) and what a file extension means (e.g. `.txt`, `.dat`, `.csv`)

Lecture 9 – Debugging

- Define the three types of bugs: syntax, execution, logical
- Identify a bug as either syntax, execution or logical

Things you should be able to do

Lecture 6 – Loops

- Define and use `for` loops and `while` loops

Lecture 7 – Arrays

- Define and use arrays, lists and tuples
- Access elements of arrays, lists and tuples in Python using indices and slicing
- Use Numpy arrays to store data and do element-wise arithmetic
- Use a loop to fill an array

Lecture 8 – File I/O and Plotting

- Import/export data into Excel from/to a text file (`.txt`, `.dat`, `.csv`)
- Make a scatter plot in Excel
- Use `loadtxt` and `savetxt` to read from and write to a text file (`.txt`, `.dat`, `.csv`) in Python
- Make, format and save plots in Python

Lecture 9 – Debugging

- Use the scientific method (observation, hypothesis, experiment) to debug a faulty Python code or Excel worksheet

II. Numerical Algebra

A. Lectures 10-12: LLMs and Linear Algebra

Things you should know

Lecture 10 – Review of Matrix Algebra

- Whether a system of equations is linear or nonlinear
- Vector notation and index notation

Lecture 11 – Gauss Elimination

- How the Gauss Elimination algorithm works

Lecture 12 – Numpy linear algebra

- Concept of computational cost in terms of memory and CPU time

Things you should be able to do

Lecture 10 – Review of Matrix Algebra

- Calculate vector norms, dot products and matrix products by hand and using Python
- Convert a system of linear equations to matrix notation
- Solve a linear system using the Gauss elimination algorithm by hand

Lecture 11 – Gauss Elimination

- Write or debug a Python code that performs Gauss elimination (forward elimination and back substitution)

Lecture 12 – Numpy linear algebra

- Calculate the number of bytes a variable or program uses
- Calculate the asymptotic behavior and approximate running time of a code
- Use Numpy functions (`add`, `subtract`, `multiply`, `transpose`, `norm`) to manipulate matrices
- Solve a system of linear equations using Python's linear algebra package (`np.linalg.inv`, `np.lingalg.solve`)

B. Lectures 13-16: Nonlinear Algebra

Things you should know

Lecture 13 – Fixed Point Methods

- Why nonlinear equations are more difficult to solve than linear equations (no guaranteed or unique solution)
- Identify types of nonlinear equations (implicit equation, polynomials)
- The concept of a fixed-point method, i.e. iterative $x^{(k+1)} = g(x^{(k)})$

- Picard's method and Newton's method: formulas and pros/cons

Lecture 14 – NLEs with SciPy

- What is in the SciPy module

Lecture 15 – Optimization

- Difference between fixed-point methods and optimization (minimum of the squared error or residual)

Lecture 16 – Engineering Problems

- Strategies to solve systems of linear/nonlinear engineering equations (units, vector notation, good guesses)

Things you should be able to do

Lecture 13 – Fixed Point Methods

- Write a single nonlinear equation in standard/residual form
- Solve a single nonlinear equation by Picard's or Netwon's method by hand, in Excel and in Python

Lecture 14 – NLEs with SciPy

- Write a system of nonlinear equations in standard/residual form in vector notation
- Solve a single NLE or a system of NLEs using `scipy.optimize.root`

Lecture 15 – Optimization

- Solve a single NLE or a system of NLEs using Excel's Solver
- Solve a single NLE or a system of NLEs via `scipy.optimize.minimize`

Lecture 16 – Engineering Problems

- Solve a system of linear/nonlinear engineering equations (with units, etc.) using root-finding or optimization methods in Excel and Python

III. Numerical Calculus

A. Lectures 17-20: Interpolation and Integration

Things you should know

Lecture 17 – Least Squares Fitting

- The concept of least-square fitting
- The definition and concept of the coefficient of determination, R^2

Lecture 18 – Interpolation

- Difference between curve fitting and interpolation
- Why we do piecewise interpolation: Polynomial uniqueness theorem and Runge's phenomenon
- What a spline is

Lecture 19 – Newton-Coates Integration

- What quadrature is
- Intuition behind composite trapezoidal rule (integrate a linear interpolation)

Lecture 20 – Jupyter and SciPy Integrals

- What a Jupyter notebook is and how to use it
- The difference between code and markdown cells in Jupyter
- When to use `quad` versus `simps`

Things you should be able to do

Lecture 17 – Least Squares Fitting

- Fit a curve in Excel and find R^2 using a trendline when appropriate
- Fit a curve in Excel by minimizing the squared-error using solver and find R^2 using RSQ.
- Fit a polynomial in Python using `polyfit`
- Fit any curve in Python using `scipy.optimize.curve_fit`

Lecture 18 – Interpolation

- Perform a 1D linear interpolation by hand or using Excel
- Interpolate a 1D function using `scipy.interpolate.interp1d`

Lecture 19 – Newton-Coates Integration

- Use the composite trapezoidal rule to integrate a function

Lecture 20 – Jupyter and SciPy Integrals

- Use `scipy.integrate.quad` and `scipy.integrate.simps` to integrate a function
- Do symbolic algebra and perform derivatives and integrals using `sympy`

B. Lectures 21-22: Initial Value Problems

Things you should know

Lecture 21 – ODEs and Explicit Euler

- How to classify a system of differential equation by: order, boundary conditions (IVP/BVP), ordinary/partial, linearity
- How numerical accuracy and stability relate to the step size in Explicit Euler

Lecture 22 – Systems of ODEs

- The standard format for writing systems of ODE IVPs in vector notation.

Lecture 23 – Integrating Excel and Python

- What a Macro is and what VBA is in Excel.
- What package you use in Python to communicate with Excel (XLWings).

Things you should be able to do

Lecture 21 – ODEs and Explicit Euler

- Solve a first-order ODE IVP using the Explicit Euler method in Excel and Python.

Lecture 22 – Systems of ODEs

- Solve a single ODE IVP or system of ODE IVPs using `scipy.integrate.solve_ivp`.
- Write a system of ODE IVPs in the standard, vector format.
- Convert a higher order ODE IVP into a system of first order ODE IVPs.

Lecture 23 – Integrating Excel and Python

- Record and run macros in Excel.
- Open/edit VBA code in Excel using the editor and execute user-defined functions.
- Export/import values to/from Excel cells using XLWings in Python.