Applied Numerical Methods with Python and Python Libraries (code Python)

Module 1 Essential Python Language

Overview/review of how to create functions, modules and classes that use numerical algorithms and related data structures.

- Creating Classes in Python
- Naming conventions
- My first class A-Z
- Constructors and initialisation
- Creating objects and class instantiation
- Access control issues

Creating Larger Classes

- Composition and Delegation
- Whole-part objects in Python
- Arrays and collections of objects
- Inheritance and subclassing
- Combining inheritance and composition

Fundamental Arrays and Data Structures

- Basic data types
- One-dimensional arrays; matrices
- n-dimensional arrays (`ndarray`)
- Data type objects (`dtype`)
- Tuples, dictionaries and lists

Modules and Packages

- Organising your classes
- Creating modules and accessing their contents
- Nested modules and packages
- Absolute and relative imports

Module 2 Functional Programming in Python

An introduction to functional programming and a discussion with easy-to-understand examples in numerical computation and its applications. In particular, *universal functions* play a central role. We also show how to write code that is a mix of the object-oriented and functional programming styles.

Introduction to Higher-order Functions (HOFs)

- What can we do with HOFs?
- Simplify HOFs by lambda forms and lambda expressions
- Lambdas and the lambda calculus
- Apply a function to a collection: `map ()`
- Pass/reject data with `filter ()`

Advanced Functional Programming

- Applications of `filter ()`
- Generator expressions
- Recursion and reduction
- Folds
- Iterables

Universal Functions

- Creating vectorised wrappers
- Handling floating-point errors and callbacks
- How Python implements the IEEE 754 standard
- Casting rules
- Universal function' methods (for example, reduce, accumulate)
- Math operations

Exception Handling in Python Programs

- Raising exceptions
- Handling exceptions
- Exception hierarchy and built-in exceptions
- User-defined exceptions

Module 3 Essential Mathematical Structures

Approximation of functions by polynomials is probably one of the most important activities in numerical analysis and its applications. To this end, we show how Python supports these activities. We also introduce univariate discrete and continuous statistical distributions as well as random number generators.

Polynomials

- 1d, 2d and 3d polynomials
- The algebra of polynomials
- Power series polynomials
- Operations on polynomials

Special Polynomials and Functionality

- Orthogonal polynomials: Chebychev, Legendre, Laguerre, Hermite
- 1d, 2d and 3d orthogonal polynomial grids
- Least Squares fitting
- Spline fitting

Random Sampling

- Simple random data
- Permuting and shuffling randomly

- Continuous and discrete distributions
- Drawing random samples
- Creating histograms

Arrays
- N-dimensional arrays `ndarray`
- Creating and manipulating arrays
- Iterating over arrays
- Applications

Module 4 Fundamental Numerical Methods
This module introduces several important libraries that are needed in many kinds of applications and that we use in later modules.

Integration
- General purpose integration schemes in one, two, three and n dimensions
- Gaussian and Romberg integration
- Trapezoid and Simpson's rules
- Gaussian quadrature roots of orthogonal polynomials

Numerical Solution of Ordinary Differential Equations (ODE, `odeint`)
- Real-valued and complex-valued ODEs
- First-order and higher-order ODEs
- Application areas

Statistics
- Random variables
- Probability and cumulative distribution functions
- T-test, Kolmogorov-Smirnov test
- Test for normality
- Comparing two samples
- Estimation
- Kernel density estimation (KDE)
- Univariate and multivariate estimation
- Applications

An Introduction to Optimisation
- Univariate minimisers and root finders
- Unconstrained and constrained multivariate optimization
- Least squares minimization and curve fitting
- Orthogonal distance regression (ODR)

Module 5 Advanced Numerical Methods
This module is central to all computationally-intensive applications because it discusses *numerical linear algebra* which consists of routines to solve matrix equations, eigenvalue and eigenvector computation as well as matrix decomposition methods based on LAPACK and Matlab. We also discuss interpolation algorithms in one and two dimensions.

Mathematical Functions
- Trigonometric and inverse trigonometric functions
- Rounding
- Sums, products and differences
- Exponential and logarithmic functions

Linear Algebra: Overview
- ATLAS LAPACK and BLAS libraries
- Basic routines
- Computing norms
- LU and Cholesky decomposition

Advanced Linear Algebra
- Eigenvalue and eigenvector computation
- Decomposition: QR, Schur, SVD (Singular Value Decomposition)
- Matrix functions (for example, the exponential of a matrix)
- Special matrices

Matrix Library (`numpy.matlib`)
- Matrix objects
- Creating and initializing matrices
- Using matrices in applications

Interpolation
- Overview of univariate and multivariate interpolation
- Interpolating a 1-d function
- Piecewise polynomial interpolation
- Piecewise linear interpolation in N dimensions
- Interpolation over a 2-d grid
- 2d splines

Module 6 Numerical Solution of Ordinary and Partial Differential Equations (ODE/PDE)
This module introduces modern finite difference (FDM) schemes that approximate the solution of time-dependent partial differential equations, in particular, parabolic PDEs. It prepares the way for work on computational finance that we discuss in Module 7.

Automatic Differentiation (AD) Packages
- What is AD?
- Using AD to compute gradient, Jacobian and Hessian
- Examples: Optimisation and ODE solvers
- Application to sensitivity analysis and Machine Learning (ML)

Solving ODEs Numerically
- Hand-crafted solutions versus `scipy.integrate.odeint`
- Scalar equations and systems of equations
- Stiff and non-stiff problems
- Using `scipy.integrate.odeint`

Some Important Finite Difference Schemes

- Explicit Euler, fully implicit
- Crank Nicolson
- Alternating Direction Explicit (ADE)
- Methods of Lines (MOL) using `scipy.integrate.odeint`

Model PDE: the one-Dimensional Heat Equation

- PDE formulation (initial boundary value problem)
- Finite difference methods for the heat equation
- Using Python libraries
- Creating a working program in Python

Implementing Convection-Diffusion-Reaction (CDR) Equations

- What is CDR?
- Numerical approximation
- Examples and applications

Module 7 Python for Computational Finance

In this module we introduce FDM, lattice and Monte Carlo (MC) methods to price financial derivatives containing state-of-the-art algorithms. The design was first implemented in C++ (by the originator of this course) and then ported to Python. This is a quick-start way to learn computational finance with the least effort.

Option Pricing Analytical Solutions

- The Black-Scholes option pricing formula
- Put-call parity
- Black Scholes greeks (delta, vega, theta, gamma..)
- Analytical formulae for American options

Trees and Binomial Method

- The binomial formula
- Creating a lattice data structure
- Cox-Ross-Rubinstein (CRR) and American options
- Binomial methods and greeks

Monte Carlo Simulation

- Valuation by simulation
- Antithetic variates and variance reduction
- Multiple stochastic factors
- Examples: Arithmetic and Geometric Asian options

Numerical Approximation of Stochastic Differential Equations (SDE)

- What is an SDE?
- Exact simulation
- Euler-Maruyama method
- Generating paths
- Modified predictor-corrector method

The Finite Difference Method (FDM), first Principles

- The one-factor Black Scholes PDE: preprocessing
- ADE, fully implicit and Crank Nicolson methods
- Computing option sensitivities
- Early exercise and Brennan-Schwartz condition

FDM, Part II

- FDM for interest rate problems
- Method of Lines (MOL)
- Cox-Ingersoll Ross (CIR) PDE/FDM
- Feller condition
- Callable bond PDE/FDM

Module 8 An Introduction to Machine Learning (ML)

This module is a gentle introduction to ML, mainly centered around ready-made Python libraries for clustering and training.

Background

- High-Level Overview of ML
- Essential underlying numerical methods
- Application areas
- Python for ML

Training Models

- Linear regression
- Gradient descent and its variants (e.g. SGD)
- Polynomial regression
- Learning curves
- Logistic regression

Clustering

- An introduction to vector quantisation
- An introduction to k-means clustering
- Clustering package
- Hierarchical clustering

Module 9 Auxiliary Libraries

This module consists of several *utility* libraries for serialisation, multi-dimensional data, date time functions and producing machine code.

Input and Output Essentials

- Load and save MATLAB files
- Birds'-eye overview of HDF5
- Dictionary of `numpy` arrays
- Working with NetCDF files
- Examples and applications

Python with HDF5

- HDF5 tools
- Reading and writing data
- Working with datasets
- Chunking and compression

Financial Functions

- Future and (net) present values
- Computing payments
- Internal Rate of Return (IRR)
- Interest rate computation

Datetime Support Functions

- Business day functions
- Valid business days
- Rolling
- Number of days between two dates

Advanced Statistical Functions

- Overview of (extensive) functions and their applications
- Chi-square test
- Kruskal-Wallis
- Kolmogorov-Smirnov
- Calculating regression line
- Geometric and harmonics means

JIT and fast Machine Code

- Introduction to Numba
- Decorating Python code
- When to use Numba

Module 10 Putting it all together: Structuring your Applications

We have included this module to create an awareness of methods and design patterns to help the software developer create maintainable and extendible code. This is needs when software systems begin to mature and extended after initial software prototypes have been created.

Big Picture

- Context diagram and data flow
- System Decomposition
- Finding modules and classes
- Creating a software prototype
- Testing and debugging code

An Introduction to Design Patterns

- What, why, when and how Design Patterns
- Creational, structural and behavioural patterns
- Discovering patterns in your applications
- The top 7 design patterns

Creational Patterns

- Factory Method
- Abstract Factory
- Builder

Structural Patterns

- Adapter
- Façade
- Decorator
- Bridge

Behavioural Patterns

- Mediator
- Command
- Strategy and Template Method
- Visitor

Your Trainer

Daniel J. Duffy started the company Datasim in 1987 to promote C++ as a new object-oriented language for developing applications in the roles of developer, architect and requirements analyst to help clients design and analyse software systems for Computer Aided Design (CAD), process control and hardware-software systems, logistics, holography (optical technology) and computational finance. He used a combination of top-down functional decomposition and bottom-up object-oriented programming techniques to create stable and extendible applications (for a discussion, see Duffy 2004 where we have grouped applications into domain categories). Previous to Datasim he worked on engineering applications in oil and gas and semiconductor industries using a range of numerical methods (for example, the finite element method (FEM)) on mainframe and mini-computers.

Daniel Duffy has BA (Mod), MSc and PhD degrees in pure and applied mathematics and has been active in promoting partial differential equation (PDE) and finite difference methods (FDM) for applications in computational finance. He was responsible for the introduction of the Fractional Step (Soviet Splitting) method and the Alternating Direction Explicit (ADE) method in computational finance. He is also the originator of the *exponential fitting method* for time-dependent partial differential equations.

He is also the originator of two very popular C++ online courses (both C++98 and C++11/14) on www.quantnet.com in cooperation with Quantnet LLC and Baruch College (CUNY), NYC. He also trains developers and designers around the world. He can be contacted dduffy@datasim.nl for queries, information and course venues, in-company course and course dates