C# and QF Applications

- Overview of the application of C# to computational finance
- Integrated approach (discuss important methods and applications)
- The main 'categories' in the course
- Advantages of this course and approach
- The student perspective

Contents

- C# Fundamentals (syntax, basic OOP)
- C# Advanced (interfaces, libraries)
- QF I in C# (lattice models, PDE, Monte Carlo)
- QF II (fixed income, equity)
- Excel Interop (almost all aspects)
- Parallel processing; functional programming in LINQ

C# Fundamentals

- Object-Oriented Thinking (migration path for non-OO developers)
- Fundamental C# syntax
- Classes 101
- Data structures (.NET libraries, matrices, dates)
- Inheritance and composition

C# Advanced

- Interfaces (protocols) and their implementations
- Data management and serialisation
- .NET libraries
- Assemblies and namespaces
- Generics (compare templates in C++)

QF in C#, I

- Exact solutions to pricing problems
- Lattice models (binomial, trinomial methods)
- Finite difference methods (FDM)
- Flexible design frameworks

QF in C#, II

- Fixed-income applications
- Bootstrapping and interpolation
- Single curve and multi curve software
- Integration into .NET framework
- Design patterns and software architectures for computational finance

Advantages/ Unique Features of Course

- Complete discussion A-Z of C#
- All examples taken from computational finance (equity, fixed income)
- Robust numerical solutions
- Migration path for VBA (and other) developers
- Excel integration
- State-of-art multithreading, LINQ

The Role of Excel

- Kind of important
- The ability to create worksheet and COM addins
- Compatibility with VBA and C++
- Using Excel for I/O and as a database
- LINQ and Excel interface

Parallel Processing etc.

- C# supports multithreading (improved *speedup*)
- Using MT to design high-performance applications
- Task Parallel Library (TPL) and Parallel For
- Parallel data processing using PLINQ

Student Perspective

- Discussion on the different ways to support the learning process
- We consider different factors that influence the schedule
- Online compared to 'face-to-face'courses
- Support and feedback

The Time Element

- In general, you should aim for [1,2] years
- Better to do less more often than a lot in spurts
- Combine theory + practice
- Build on concrete examples and previous experience

The 'Process'

- Module-based (listen to each module in sequences)
- Do the exercises for that module
- Send your answers to Datasim
- Go to next module

Project

- This is an optional part of the course (at the discretion of the student)
- Student + trainer agree on a suitable topic, ideally asap
- Student applies the methods learned
- Trainer helps with scoping and design of system