DATASIM
EDUCATION BV

Datasim's Financial Courses

Course guide for November 2007 until July 2008

Introduction

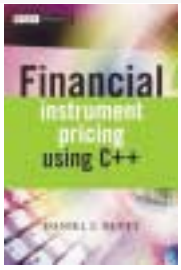
The Datasim range of courses in Quantitative Finance have been designed for those professionals who model and develop derivatives products using robust and accurate numerical methods and who deploy the resulting algorithms in powerful object-oriented languages such as C++ and C#. Datasim is one of the few training institutions that can offer a number of courses on topics that are of vital importance in quantitative finance:

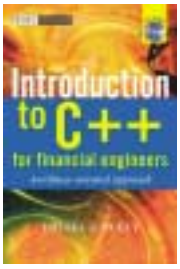
- The Finite Difference Method (FDM) for one, two and three factor derivative modeling
- The Monte Carlo method and simulation techniques for multi-factor problems
- Numerical Methods for Quantitative Analysts
- Introductory and Advanced C++ courses for finance
- Multi-threaded and parallel processing in C++

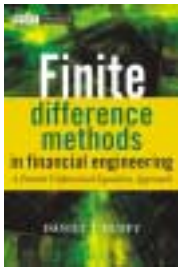
We give these courses in various venues throughout the world, for example, London, New York City and Frankfurt. Another popular option is in-company courses where the same courses as above can be given or alternatively a customized course based on your specifications. Once you have finished our course you can avail of the free forums at the Datasim web site to ask questions on the course, download relevant code and communicate with other course attendees.

These courses were developed by Dr. Daniel J. Duffy, an internationally known trainer, Wiley author and practitioner of numerical methods in C++ for derivatives pricing. He is the author of three popular books on finance. At the moment he is developing a customisable C++ Monte Carlo framework which will be published as a Wiley book in 2007 (co-author is Dr. Jörg Kienitz).

Books written by Daniel Duffy and published by Wiley







Contents

Prices and dates

FDM	The Finite Difference Method for Quantitative Finance	03 - 05 March 2008	London	3.399,-- euro
BDMC	Building and Deploying C++ Frameworks for the Monte Carlo Method	21 - 23 July 2008	London	3.339,-- euro
NMF	Numerical Mathematics for Finance Recipes, Applications and C++ code	31 Janyary 1-2 February 2008 19 - 20 June 2008 3 - 4 July 2008	London New York London	2.266,-- euro
CPP-Quant	C++ for Quant Developers	14 - 16 April 2008	London	3.399,-- euro
CPPFI	Advanced C++ for Financial Instrument Pricing	28 - 30 January 2008 16 - 18 June 2008 02 - 04 June 2008 30 June 1 - 2 July 2008	London New Yor Frankfurt London	3.399,-- euro
DIMP	Designing and Implementing Multi-threaded Applications in C++ With Applications to OpenMP		London	2.545,-- euro
CPP-FIS	C++ for MSc and Phd Students; Preparation for a Life in Quantitative Finance			

The Finite Difference Method for Quantitative Finance

Theory, Applications and Computation

This intensive hands-on course is concerned with the discovery and detailed analysis of finite difference schemes for the pricing of a range of one-factor and multi-factor equity and other kinds of derivatives. This course assembles some of the most powerful and proven finite difference schemes and applies them to approximating the PDEs for financial instruments. Your trainer is Dr. Daniel J. Duffy.

Overview of Course

This course introduces state-of-the art and proven finite difference schemes. We apply them to a range of derivatives products.

The main features of this course are:

- A complete analysis of FDM, from mathematical formulation to algorithm design and mappings to an object model
- One-factor, two-factor and three-factor problems
- Which schemes work and which do not (and why)
- Schemes for early exercise, jumps and path dependencies
- Accurate schemes for problems with mixed derivatives (correlation)
- Iterative methods and domain decomposition techniques

In short, this course deals with the most important schemes that will allow you to implement derivatives using the partial differential approach in combination with finite difference methods.

What previous delegates have said?

"My expectations were topped; can go now and implement 3d models using splitting"
"Really liked it. Very informative"
" I would enthusiastically recommend this course to colleagues"
"Trainer knows FDM 150%"

What do you learn?

After having completed this course you will be in a position to apply FDM to Quantitative Finance applications. You can bring your own test cases and examples as input. The focus is on the full project, from the financial model through PDE, FDM and numerical experimentation in C++ and Excel. You may bring your own laptop if you wish to try out the provided code during the course.

What do you receive?

Full set of slides, CD with software and a copy of Daniel Duffy's book "*Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*". You are invited to ask questions on your own specific applications as well.

Course contents updated October 2006

Course contents

Part I: Fundamentals

Quick Review of Partial Differential Equations
- Parabolic, hyperbolic and elliptic PDE
- Boundary and initial (payoff) conditions
- One-factor and multi-factor PDE
- Existence and uniqueness results
- PDE and its relationships with Stochastic differential equations

The PDE taxonomy in Quantitative Finance
- Properties of the original and generalised PDEs
- Fixed and free boundaries
- PDE and payoffs for correlation options
- Heston stochastic PDE
- Jump models and Partial Integro-Differential Equations (PIDE)
- Introduction to FDM for PDE

Meshes, divided differences and finite difference schemes
- Semi-discretisation (Method of Lines, Rothe's method)
- Full discretisation in time: Euler, Crank-Nicolson, Runge-Kutta
- Solving discrete sets of equations

Consistency, stability and convergence
- Conditional and unconditional stability
- Von Neumann stability analysis versus M-matrix approach
- Examples from QF
- Important Issue: Matrix Computation

First Path-Dependent Case
- Barrier options
- Continuous versus discrete monitoring
- Using different kinds of payoff function
- Discontinuous rebates
- Complex barriers

Part II: Core Principles

One-factor Models
- Explicit and implicit schemes
- Discretisation of boundary conditions (Dirichlet, Neumann, linearity)
- Payoff and smoothing effects
- Setting up the discrete systems of equations
- Barrier and Lookback Options

Continuous and Discrete monitoring
- Modelling jumps and time-marching schemes
- Approximating the greeks (delta, gamma, theta, vega)
- Asian Options

Parabolic and hyperbolic components
- The problems with centred differencing and numerical boundary conditions
- Upwinding and downwinding schemes
- Stability and convergence
- One-factor Models (early exercise)

Penalty methods, change of variables, LCP: comparisons
- The Landau transformation in detail
- The penalty method in detail
- Implicit, semi-implicit and explicit schemes
- The predictor-corrector method

Creating a generic time-marching scheme
- Implementing boundary conditions: first-order and second-order schemes
- Solving tridiagonal systems
- Modelling integral equations and jumps
- Test Cases

Approximation of the smooth pasting conditions
- Comparing the performance of the different schemes
- Predictor-corrector versus penalty method
- Calculating greeks near the strike price
- C++ payoff classes and integration into framework

Part III: Applications and Extensions

An Overview of the FDM Categories
- Overview and summary of the schemes from the first two days
- Alternating Direction Implicit (ADI)
- Splitting and Locally One-Dimensional (LOD) schemes
- ADE: explicit and unconditionally stable methods
- Rothe's method and iterative elliptic systems
- Comparing the different schemes
- Attention Points and challenges

Achieving good accuracy and performance
- Approximating mixed derivatives
- Convection-dominated problems and spurious oscillations
- Exponential fitting algorithms

ADI in Detail
- ADI 'classico'
- Craig-Sneyd scheme for problems with mixed derivatives
- Solution by LU decomposition
- Examples
- Splitting in Detail

Explicit and implicit methods
- Approximate factorization methods
- Handling Dirichlet, Neumann and linearity boundary conditions
- Examples
- PIDE

The Merton jump model
- Approximation of PIDEs
- Splitting and predictor-corrector methods
- Two-factor American Options

Equity Models
- Approximation of PIDEs
- Two-factor American Options
- Multi-Asset options

Fixed-Income Applications
- Convertible bonds
- Two-cycle second order splitting methods
- Two-factor IR applications
- FX swaps with stochastic volatility

Setting up the Equations
- Mixed derivatives
- Solving the system of equations
- Multi-asset payoff function: handling non-smooth payoffs
- Approximating the greeks
- N-Factor Models, N = 3, 4, ...< /LI > < /UL >

Who should attend?
This course has been developed so that you can use the theory to solve existing problems as well as applying the knowledge to the pricing of new financial instruments. In particular, the course is for professionals with a strong mathematical background:
- Financial engineers who design new pricing models
- Analystsand quants
- Other professionals who need to understand and apply advanced numerical methods to derivatives pricing

Course duration
3 days.

Follow-up courses
Advanced C++ for Financial Instrument Pricing.

Building and Deploying C++

Frameworks for the Monte Carlo Method

From financial model to running code

This hands-on workshop applies C++ and related software design techniques to creating code that prices and hedges one-factor and multi-factor financial derivatives products using the Monte Carlo (MC) method. We discuss a representative number of one-factor and multi-factor option models and we show how they are approximated using MC and how to set up running C++ code to test these models.

Overview of Course

This practical course is for those professionals who design and implement MC models in C++. To this end, the course focuses on a number of essential issues:

- Modelling one-factor and multi-factor option models using MC
- Detailed review of underlying theory (RNG, SDE, distributions)
- State-of-the art customizable software frameworks for MC modelling
- Full C++ code for all models

What do you learn?

The main goal of this course is to show how to apply C++ to the pricing of derivatives using the Monte Carlo method. The emphasis is on customising a ready-made C++ framework to suit your own needs.

What do you receive?

In this course you get the course slides, CD with full source code and Daniel Duffy/Joerg Kienitz book "*Monte Carlo Object-Oriented Frameworks in C++; Building Customisable and High-Performance Applications*".

This means that you can practice at your own location and in your own time after the course. Your trainers are Dr. Joerg Kienitz and Dr. Daniel J. Duffy.

PLEASE NOTE: To participate in the course, you need to bring your own laptop computer with a C++ compiler (ideally, Microsoft's Visual Studio)

Course contents

Part I: Fundamental Models and Building Blocks

Distributions

- Overview of distributions in finance
- Normal, gamma, Poisson and others

Inverse cumulative distribution

- C++ implementation

Random Number Generation

- Uniform number generation
- Quasi-random number generation
- Non-uniform number generation
- C++ implementation

Review of Stochastic Processes

- Brownian motion
- Ornstein-Uhlenbeck
- Poisson and Compound Poisson processes
- Levy processes
- Variance Gamma
- Normal Inverse Gaussian
- Modelling stochastic processes in C++

An Introduction to Stochastic Differential Equations

- Motivating SDEs
- Linear and non-linear SDEs
- One-factor and n-factor SDEs
- Examples and categories of SDE
- Modelling SDE in C++

Part II: Numerical Methods

Quick Review of the Finite Difference Method

- Discretising ordinary and stochastic differential equations
- Stability and convergence
- Low-order and high-order methods

Numerical Approximation of SDE
- Euler-Maruyama method
- Milstein method
- Predictor-Corrector method
- Advantages and disadvantages

Advanced Numerical Methods
- Symplectic methods
- Splitting methods
- Designing C++ classes for SDE

Performance and Accuracy Issues
- Extrapolation
- Choosing a good scheme
- C++ and multi-processing

Part III: C++ Design Patterns and Software Frameworks

Overview and Background
- Design rationale
- Review of the essential GOF (Gamma) patterns
- Essential system-level patterns
- Documentation: class diagrams and component diagrams

The Monte Carlo Framework: The Structure
- Blackboard architecture
- Mediator and Layers
- Customisation: using your own code in the frameworks
- Using object orientation and C++ templates

Walkthrough
- Approximation formulae
- Pricing barrier options in the framework
- Continuous and discrete monitoring
- Modelling option sensitivities
- Autocallable certificates

Part IV: Applications in Quantitative Finance

Overview
- Path-dependent options
- Taxonomy of path-dependent options
- Modelling option classes for path-dependency
- Performance and accuracy issues

Stochastic Volatility Models and Monte Carlo
- Introduction
- The Heston model
- Cliquet options
- C++ implementation

Early Exercise and American Options
- Longstaff-Schwarz regression method
- Dual method
- Case studies
- Performance improvements

Multifactor Models
- Introduction and overview
- Spread options and other correlated options

Presentation and Visualisation
- Excel output
- Integration with XLL
- Statistics and reporting

Prerequisites
We assume that you have C++ knowledge and that you have experience of the Monte Carlo method and its related topics. In this workshop the exercises take the form of integrating your code in the framework. The percentage theory/practice is approximately 80:20.

Who should attend?
Quantitative Analysts, Treasury stuff using MC, Asset Managers using MC, Risk Controller, IT Developers for Derivatives Pricing Models

Duration
2 days.

Numerical Mathematics for Finance

Recipes, Applications and C++ code

This two-day course is meant as a good introduction to, and overview of a number of important mathematical and numerical techniques that are used in many areas of Quantitative Finance. These numerical techniques underpin many of the important pricing and hedging algorithms in finance and for this reason it is important that they are well-understood. To consolidate the theory we provide working C++ code to show how algorithms are mapped to software.

Overview of Course

The goal of this course is to prepare you for algorithmic work in derivatives pricing and hedging. Many of the numerical techniques – for example Monte Carlo, Finite Difference Method and quadrature techniques – rely on more fundamental numerical methods and it is here that these are discussed in detail:

- Solving linear systems of equations using direct and iterative methods
- Eigenvalues, eigenvectors and Cholesky decomposition
- Solving non-linear systems of equations
- Approximation (interpolation, extrapolation)
- Numerical integration in one and several dimensions
- Continuous and discrete Fourier analysis
- Linear and non-linear optimisation
- Statistics and modeling of data

We discuss each topic in enough detail so that you can apply it in your daily work.

What do you learn?

In this course you learn numerical analysis of each of the above topics in a number of steps:

- Presentation of the fundamental theory
- Several useful, simple examples to show how the theory works
- A more complex example (usually associated with finance)
- Using the delivered C++ code to test some examples
- Many exercises (and answers) that you can work out at home

We also discuss and show by examples in C++ a number of relevant problems:

- Option pricing using Fast Fourier Transform
- Quadrature methods for derivatives pricing
- Interpolation and curve fitting
- Generating correlated random variables
- Calibration and optimization

In short, the goal of this course is to get you up to speed as soon as possible so that you can start on real-life systems without having to reinvent the wheel.

What do you receive?

In this course you receive the course slides, CD with full source code. This means that you can practice at your own location and in your own time after the course. You can also participate on the author's forum where we have support for this course.
Your trainer is Dr. Daniel J. Duffy.

PLEASE NOTE: If you wish to run the C++ programs related to the course, you need to bring your own laptop computer with a C++ compiler (ideally, Microsoft's Visual Studio). However, it is not mandatory.

Course Contents

Part I: Algebraic Systems

Linear Algebra

- Review of matrices and vectors
- Matrix multiplication
- Some properties of matrices
- Categories of matrices

Solution of Linear Systems, Part I

- Gaussian elimination
- LU decomposition
- Tridiagonal systems
- Cholesky decomposition

Solution of Linear Systems, Part II
- What are iterative methods?
- Jacobi, Gauss-Seidel and SOR methods
- Accelerating convergence
- Sparse linear systems

Nonlinear Systems
- One-dimensional and multi-dimensional problems
- Secant and Regula Falsi methods
- Newton-Raphson method
- Nonlinear systems of equations
- Continuation (homotopy) methods

Part II: Approximation

Interpolation and Extrapolation
- Polynomial versus rational interpolation
- Piecewise polynomial interpolation (linear, cubic spline)
- Interpolation in two and more dimensions
- Polynomial versus rational extrapolation

Numerical Integration
- Some standard integration formulae
- Gauss-Lobatto
- Adaptive Gauss Kronrod
- Numerical integration in multiple dimensions

Fourier Analysis and Fast Fourier Transform
- Review of complex numbers
- Functions of a complex variable
- Fourier transform and its inverse
- Parseval's formula
- Fast Fourier Transform (FFT)
- Discretely sampled data
- Multiple dimensions

Part III: Optimisation

One-dimensional and multi-dimensional problems
- Brent's method
- Simplex method
- Powell's method
- Conjugate gradient

Part IV: Statistics and Modelling

Statistics
- Moments of a distribution
- Linear correlation
- Nonparametric correlation
- Comparing two distributions

Modelling of Data
- An introduction to least squares
- Nonlinear models
- Confidence intervals

Prerequisites
Some mathematical knowledge is assumed. In particular, knowledge of calculus and basic linear algebra is essential. If you have any queries about these issues please do not hesitate to contact me at dduffy@datasim.nl

Who should attend?
This course is for all those professionals who are involved with quantitative finance applications and who need to understand the fundamental numerical algorithms that underpin many pricing and hedging models. It can be followed by those who wish to get a good overview of the most important techniques as well as those who wish to refresh their knowledge of numerical mathematics.
This course is also suitable for entry-level quants and IT developers.

Duration
2 days.

C++ for IT, Quant Developers and Quant Analysts in Financial Institutions

This hands-on course teaches you about C++ and how to apply it to real-life financial engineering examples and applications. We build the knowledge in an incremental fashion and we dovetail the theory with appropriately chosen examples. By the end of this course you should be in a position to apply C++ in your own financial engineering applications.

Overview of course

This course is unique in the sense that the theory is interleaved with real examples for financial engineering and instrument pricing. The advantage for the student is that the examples, test cases and applications in C++ are of direct relevance to the financial world. Thus, this is not a generic C++ course but is timely and relevant to your needs. In this course you will develop C++ classes almost from the word go:

- Option classes, portfolios
- Temporal data structures and curve classes (e.g. yield curves)
- Classes for hedging: spreads and straddles
- Vectors, matrices and tensors
- Lattice models in C++
- Modelling option sensitivities

What do you receive?

In this course you get the course slides, CD with full source code and Daniel Duffy's book on *C++ for Financial Instruments* (John Wiley). This means that you can practice at your own location after the course. You can also e-mail the trainer if you have questions. Your trainer is Dr. Daniel J. Duffy.

PLEASE NOTE: To participate in the course, you need to bring your own laptop computer with a C++ compiler (e.g. Microsoft's Visual Studio)

Course contents

Part I: Basics of C++ and Object-Oriented Programming

Introduction to the Object-Oriented Paradigm

- Classes and objects
- Encapsulation and Information Hiding
- Inheritance and reusability
- Aggregation and association structures
- Generic classes

Examples in Finance

- Option and Asset classes
- Spreads and strangle
- Modelling option prices and sensitivities
- Portfolio in object-oriented languages
- Temporal data structures and yield curves

C++ Class Basics

- Member data and member functions
- Constructors and destructors
- Accessing members
- How to write a C++ class: the steps
- Creating my first Option class

Moving up: improving your Class

- Call-by-reference and call-by-value
- Function overloading
- Copy constructor
- The 'this' keyword
- Creating classes for Spread, Straddles and Strangles

Operator Overloading

- Applying operator overloading
- Binary and unary operators
- Friend and non-friend operators
- Where and when to use in Financial Engineering
- Example: operator overloading for Vectors and Matrices

Memory Management
- The Stack and the Heap
- The 'new' and 'delete' operators
- Working with pointers
- Pointers and arrays of Option

Part II: Advanced C++ for Financial Engineering

Introduction to Inheritance in C++
- Implementing ISA and AKO relationships
- What the different scenarios for inheritance?
- Inheritance and memory management
- Exotic and dangerous applications of inheritance

Advanced Inheritance
- Abstract and concrete classes
- Pointers to base class
- Static and dynamic casting
- Polymorphic (virtual and pure virtual functions)

Applying Inheritance to Financial Engineering
- Creating a portfolio class hierarchy
- Composite and nested classes
- Polymorphism: calculating risk and price of a portfolio
- Some tips and Guidelines

An Introduction to Templates in C++
- What is a template?
- Template classes and template functions
- Designing and implementing templates
- Example: the Property pattern
- Modelling an option's parameters

Standard Template Library (STL)
- Overview of functionality
- Data containers
- Lists, vectors and maps
- Navigating in containers with iterators

Part III: Applications and Test Cases

Programming Lattice Models
- Creating lattice structures in C++
- Integrating functionality for European and American styles
- Using iterators for backward and forward induction
- Modelling option sensitivities

Time-dependent Data Structures
- Date and Time classes
- A hierarchy of curve classes
- Specials: discount, dividend and yield curves
- Applying curves in fixed income examples

Prerequisites
We assume that the student has a reasonable knowledge of a high-level programming language such as C, Pascal, Java or Fortran. Of course, the examples are taken from the financial engineering world.

Who should attend?
Quantitative and IT developers, programmers, analysts and designers and those who are involved with software development using C++ for financial engineering applications. This course is suitable for applications involving option pricing, risk management and fixed income, for example.

Course duration
3 days.

Follow-up courses
Advanced C++ for Financial Instrument Pricing.

Advanced C++ for Financial Instrument Pricing

This intensive hands-on C++ course is meant for those professionals in Quantitative Finance who design and implement finance models for a variety of derivatives products. The course contents are based on Daniel J. Duffy's 20 years experience in C++ and its applications and it is the only course of its type to offer all the techniques that are needed in order to develop flexible and robust applications.

What previous delegates have said?

"Very good style and knowledge was far above the norm"
"Excellent hands-on teaching"
"Good balance of C++ and finance, theory and practice"
"The book I wish I had had when I first started studying C++"

What do you learn?

In this course we introduce state-of-the-art design and programming techniques in C++. In particular, the following topics are discussed in detail:

- Advanced C++ syntax and its application in QF
- Template classes and the Standard Template Library (STL)
- Combining the object-oriented and generic programming paradigms
- The famous Gamma (GOF) design patterns applied to QF
- Interfacing to Excel: COM Add-ins
- Creating applications: Monte Carlo, Finite Difference and lattice methods

What do you receive?

As attendee you receive a full set of slides, C++ source code and a copy of Daniel Duffy's book "*Financial Instrument Pricing using C++*" (Wiley 2004), including CD with C++ code. In short, you will receive what is needed to start developing your own QF applications. The price includes coffee, tea, lunch and refreshments.

PLEASE NOTE: You are expected to bring your own laptop to the course.

Course contents updated September 2006

Course contents

Part I: Fundamentals

Basic C++ Refresher

- Classes, member data and member functions
- Values, references and pointers
- Operator overloading
- The 'const' keyword
- Inheritance in C++

Memory Management

- Stack and heap memory allocation and deallocation
- Creation of single objects and arrays of objects
- The keywords 'new' and 'delete'
- Avoiding memory leaks

Run-time Behaviour in C++

- Run-Time Type Information (RTTI)
- Static and dynamic casting
- Client-server programming and exception handling
- The keywords 'try' 'throw' and 'catch'
- Where to use exception handling

Mini Application: Creating Payoff Hierarchies

- Abstract and concrete payoff classes
- Implementing payoff classes; the strategies
- Using payoff classes in QF applications
- Two-factor and multi-factor payoff hierarchies

An Introduction to C++ Templates

- The Generic Programming (GP) paradigm
- Template classes and template functions
- Nested templates
- Advantages of templates

Part II: Core Principles

An Introduction to the Standard Template Library (STL)

- Major components in STL
- Containers, algorithms and iterators
- How can I use STL in Quantitative Finance?
- Using STL to create your own classes

STL Containers
- Sequential and associative containers
- Lists, vector and queues
- Maps, sets and multimaps; hash_map
- Modelling option data with maps and Property Sets

STL Algorithms and Iterators
- Mutating and non-mutating algorithms
- Searching and sorting
- Inserting and removing data
- The role of iterators
- Complexity Analysis and performance issues

An Introduction to Design Patterns
- Overview of the 23 GOF patterns
- Pattern categories
- Most important patterns in Quantitative Finance
- System patterns (POSA) and domain architectures

Essential Patterns: Creational
- Factory method and abstract factory
- Creating complex objects using Builder
- Singleton
- Creational patterns in Quantitative Finance

Essential Patterns: Structural
- Composite and nested objects (for example a CDO)
- Bridge and device-independence
- Extending object structure with Decorator
- Smart pointer ('handle'); auto_ptr
- Structural patterns in Quantitative Finance

Essential Patterns: Behavioural
- Command and its application to user-interface components
- Strategy and algorithms
- Extending class functionality with Visitor
- Template method pattern and customisable frameworks
- Behavioural patterns in Quantitative Finance

Part III: Applications

Overview and Introduction
- C++ as a language for application development
- Input, processing and output issues
- Types of applications

Excel Interfacing and Add-ins
- Excel Visualisation package
- Using the package to verify algorithm output
- An overview of Excel add-ins
- Creating Automation add-ins and worksheet functions
- Creating COM add-ins: The steps
- The XLOPER interface

The Monte Carlo Method
- Description of the problem
- Creating a software framework for a MC engine
- Using design patterns to create flexible MC systems
- Plain options, Asians and barriers
- Presentation and statistics
- Extensions to multi-factor MC

The Finite Difference Method
- A quick introduction to FDM
- C++ classes for a FDM solver
- Explicit and implicit schemes
- Presentation in Excel
- One-factor and two-factor models

Creating Libraries for QF Applications
- Interpolation and yield curves
- Numerical linear algebra
- Numerical integration

Prerequisites
We assume that the student has some experience of C++. This is not a beginners course and we assume you know what constructors, destructors and operator overloading are in C++ and how memory management works.

Who should attend?
This course has been developed for financial professionals who design and implement pricing and hedging models in C++ and Excel. The course introduces and elaborates on how to apply C++ to creating flexible and reliable applications in Quantitative Finance using the most modern software design techniques. There is ample room for questions on your own specific applications as well as hands-on programming sessions. It is assumed that the attendees have some working knowledge of C++ and have developed applications or prototype applications in that language.

Course duration
3 days.

Designing and Implementing Multi-threaded Applications in C++

With Applications to OpenMP

This 2 day hands-on course discusses how to create efficient software applications on multi-core computers in C++ in combination with OpenMP, a C++ library for specifying shared-memory parallelism in C and C++. The course is self-contained and no previous knowledge of multi-threading (MT) is needed in order to follow the course.

Overview of Course

This course builds the concepts, techniques and tools from the ground up, so that the student will be in good position to design and code efficient and robust applications that use the full potential of powerful multi-core processors. In order to realise this goal the course discusses the following topics in detail:

- The foundations for multithreading; major ideas and techniques
- Designing MT applications; models and design patterns
- Learning and applying the OpenMP library
- Using OpenMP to write applications for multi-core machines

In short, this course covers all the activities to be executed when developing multi-threaded applications.

Contents

Part I: Fundamental Concepts

Introduction and Overview
- Concurrency and Parallelism
- Coarse-grained and fine-grained parallelisms
- Processes and threads

Threads in Detail
- Thread structure
- Creating and starting a thread
- Stopping and killing a thread
- Inter-thread communication

Synchronisation
- High-level and low-level mechanisms
- Semaphore and mutex variable
- Condition variable
- Critical section
- Reader/writer locks

Suitable Applications for Multi-Threading
- Task independence
- Responding to asynchronous events
- Buffering
- Task priority in applications
- CPU cycles and time-consuming calculations

Part II: Threading Models and OpenMP Support

Fundamental Models
- Boss/Worker (Master-Slave) model
- Peer model
- Pipeline model

Multi-threaded Design and System Patterns
- Whole-Parts pattern
- Builder pattern and parallel object construction
- The importance of the Mediator pattern
- Parallel Observer and Strategy patterns

Part III: Using the OpenMP Library

Overview of OpenMP Functionality
- Shared-memory parallelism
- Compiler directives
- Library functions
- Environment variables

OpenMP Directives; Overview
- `#pragma`
- Conditional compilation
- The `parallel` construct
- "Hello World" program

Work-Sharing Constructs
- Loop optimization (`for`)
- Static, dynamic, guided and run-time scheduling
- Performance issues
- The `sections` directive

Data Environment
- Access to data in parallel regions
- Data-sharing attribute clauses
- Private and public data

Run-time Library Functions
- Execution environment functions
- Thread information
- Lock functions
- Simple locks and nested locks
- Timing routines

Part IV: Applications

Overview
- Parallelising existing applications; the alternatives
- Designing new parallel applications
- Embarassingly parallel applications
- Speed-up factor

Application Categories
- Matrix and vector processing
- Pipelining applications
- Interactive applications
- Database applications

Prerequisites
We assume that the student has several years of programming experience in a high-level language, for example C/C++, Fortran, Matlab or Pascal.

Duration
2 days.

C++ for MSc and Phd Students
Preparation for a Life in Quantitative Finance

A C++ course is being proposed for MSc/Phd students. A provisional syllabus of the material to be covered is detailed below. The course will be taught by Dr. Daniel Duffy who has taught and written a number of books on C++ and Finance (Financial Instrument Pricing Using C++). Please email if you have any further questions or are interested in attending the course.

Goals of the Course

The goals of this intensive three-day hands-on C++ course is to introduce the essential syntax of the C++ language and apply it to a number of relevant examples and applications in Quantitative Finance (QF) .

This course is unique for a number of reasons: first, it delivers C++ knowledge with all examples taken from QF and numerical mathematics. This approach will help you later during your professional career. Second, the course fee has been priced so that it is affordable for students with a tight budget. After having attended this course you will have a good understanding of C++ and how to apply it to a number of problems in QF:

- Lattice methods (binomial and trinomial trees)
- Monte Carlo simulation Modelling the Black Scholes model in C++
- Finite Difference Methods

On the last day it is possible to do an exam to test what you have learned.

What do your get?

Course documentation, full source code and other relevant material.

Since this is a special low priced course for MSc/Phd students, you should bring your own laptop and arrange your own lunch.

This course will be given in English.

Course contents

The course consists of three major blocks (one block per day)

Part I: C++ Fundamentals

Classes and objects

- Member data and member functions
- Improving your classes
- Operator overloading

Memory management

- The operators 'new' and 'delete'
- Pointers and arrays
- Tips, pitfalls and guidelines

Basic Inheritance

- What is inheritance?
- Inheritance and memory management
- Examples in QF

Part II: Advanced C++ Features

Advanced Inheritance

- Abstract and concrete classes
- Dynamic casting
- Polymorphic functions
- Polymorphic containers

Introduction to C++ Templates

- Template classes and template functions
- Creating template classes
- Extending using inheritance and composition
- Nesting template classes

Standard Template Library (STL)
- Overview of functionality
- Containers (vector, list, map)
- Iterators
- Algorithms

Advanced Templates
- Creating reusable data structures
- Vectors and matrices
- Lattice data structures
- The Property pattern

Introduction to Component Object Model (COM)
- What is COM?
- Interfaces and component objects
- Implementing interfaces
- Using COM in QF applications

Part III: Applications in Quantitative Finance

C++ and Numerical Mathematics
- Using C++ in numerical analysis
- Matrix computation
- Interpolation and curve fitting
- Numerical integration

Black Scholes Model
- C++ classes Implementing option sensitivities
- Creating a simple option engine
- Visualisation in Excel

Lattice Models
- Creating generic lattice structures
- Forward and backward induction
- Option pricing problems

Finite Difference Methods
- Quick overview partial differential equations
- Mapping finite difference methods to C++
- Implicit and explicit methods
- Relationships with the trinomial methods

Prerequisites
You must be a full-time student at an institution of higher learning. Some knowledge of at least high-level programming language, for example Pascal, Matlab, Java or C. No previous knowledge of C is assumed

Who should attend?
This course is a special course organised for MSc/Phd students. You must be a full-time student at an institution of higher learning to attend this course.

Duration
3 days.

Distance Learning For Financial Engineers

This practical, hands-on course teaches you C++ - from the fundamental to advanced level – and how to use it to create flexible and robust applications in computational and quantitative finance.

Some key features of the course are:

- You can start in January 2008 and continue up to October 2008. You can take the course in your own time in a period of up to 10 months.
- You are eligible to take exams and present a small project dissertation if you wish; a graded certificate will be given to the student.
- End of month milestones; review of all exercises, new material dispatched to the student for the coming month.
- Students have full access to the Datasim forums, source code and other relevant documentation. In this sense we shall have a community of people working on the same problems and can help – and be helped – with all those little annoying problems that get in the way of the real work.

Student Profile

Ideally, we assume that the student is working in finance and is acquainted with derivatives modeling and implementation. Knowledge of at least one high-level language is recommended but it is not necessary to know C.

Course Contents

Module 1: Primer and Fundamental C++ Syntax

In this module we introduce basics of the C++ language, including essential syntax, how to create functions and classes and how to integrate the code into a C++ project.

- The 'survival guide' to the C language
- Learning the C++ project environment
- From source code to running program
- Creating basic C++ classes: header and code files
- Creating robust classes (const, call by value/reference)
- Operator overloading in C++
- Creating user-defined operators
- Memory management: heap, stack and static
- Implementing contracts: exception handling in C++
- Project: creation simple C++ classes for financial derivatives

After having completed this module you will be in a position to write, compile and run C++ applications and be able to test and debug code quickly and effectively. This means that you will not lose valuable time. We take a number of examples from finance, namely exact formulas for option pricing and the creation of C++ classes that model derivatives.

Module 2: Advanced C++

In this module we introduce a number of advanced techniques that promote the flexibility and robustness of your C++ applications. This is a crucial module because many C++ applications use these techniques and they allow us to extend and modify system code with a minimum of impact on the stability of the application.

- Pointers: native C++, Boost shared pointers
- Modelling functions: by pointers and by function objects
- Applications in finance
- An introduction to inheritance and composition in C++
- Virtual and pure virtual functions
- Tips and guidelines when using inheritance
- Combining inheritance and composition
- Run-Time Type Information (RTTI)
- Factoring common code using the Template Method Pattern
- Project: creating flexible payoff hierarchies

After having completed this module you will be able to create extendible and understandable C++ class hierarchies for financial derivatives. We shall use these classes as reusable building blocks when we develop applications in later modules.

Module 3: C++ Templates and the Standard Template Library (STL)

This module introduces the student to Generic Programming (GP) and its implementation in C++, namely the template mechanism. We discuss the fundamental syntax issues and we show how to create templated functions and classes. Furthermore, we show how to integrate and combine templates with the inheritance and composition techniques that we discussed in previous modules. Having learned what templates are we then proceed to discussing the most important components of STL and their applications.

- An introduction to the generic programming model

- C++ templates: functions and classes
- Template specialization
- Combining templates with inheritance and composition
- An overview of STL
- STL sequence containers: list, vector, deque
- STL iterators
- Associative containers: map, set, multimap, multiset
- STL algorithms: searching, sorting, extraction
- Project: using templates for financial applications

After having completed this module the student will understand template programming in C++.

Module 4: Design, Patterns and Multi-Threading

In this module we introduce a number of design techniques that we deploy in C++ so that our applications can be customized when requirements change (as they inevitably do). In particular, we give an overview of the famous Design Patterns (23 in total) and we apply the most important ones to examples and applications in finance.

- What is software design?
- Quick overview of the Unified Modeling Language (UML)
- The Gamma ('Gang Of Four') classification
- Creational patterns: Factory, Singleton, Builder, Prototype
- Structural patterns: Bridge, Composite, Facade, Proxy
- Behavioural patterns I: Template method, Strategy, Observer
- Behavioural patterns II: Visitor, Command, Mediator
- Applying design patterns in finance: the steps
- Project: designing and implementing FDM for Black-Scholes PDE

After having completed this module you will be able to discover and apply the most appropriate design patterns for a given problem in finance.

Module 5: Libraries and Interfacing Issues

Whereas the code in Module 4 was concerned with application logic and algorithms this module discusses a number of features and tools that allow us to develop fully-fledged applications, in particular the input, processing and output modules in an application.

- C++ Excel integration: xll, Automation and COM Addins
- Creating xll applications
- Automation Addins and worksheet functions
- COM Addins
- Registration, activation, libraries
- An introduction to ATL (Active Template Library)
- Overview of the Boost library
- Boost random number generators
- Boost multi-array and property map libraries
- Introduction to XML
- DLLs and Libs
- Calibration
- Project: developing Excel Addins

After having completed this module you will be able to integrate your code with a number of standard software environments, such as Excel, Boost and XML.

Module 6: Integration and Applications

This is the last module of the course and it is here that we create a fully-fledged application using the experience that we have gained in the first five modules. You can choose the kind of application (equity, fixed income, commodity) and the numerical technique (FDM, Monte Carlo, ...) you wish to use.

- Analysis and system decomposition
- Defining inter-system interfaces
- Applying the GOF patterns
- An introduction to multi-threading and parallel programming
- Implementing finance applications in C++ with OpenMP
- Testing and profiling your application
- Integration with Excel
- Equity, interest rate and other applications in finance
- Monte Carlo, FDM, quadrature and lattice solutions
  Project: term (final) project

After having completed this module you will have used C++ in combination with mathematical methods for finance to produce a working system.
The examiners will review your project and a small exam will be given.

Price

- Modules 1 - 6 cost 3945 euro (excluding 19% VAT). Total price including VAT will be 4694.55 euro.
- Modules 4 - 6 cost 2367 euro (excluding 19% VAT). Total price including VAT will be 2816.73 euro.

What do receive?

The course material is:

- The four books by Daniel J. Duffy (see www.datasimfinancial.com)
- Workbooks, exercises and source code
- Access to our site: forums, videos and email
- Examination at the end of the course
- Student project (at the discretion of the student) and evaluation
- Certificate

Datasim Education BV

Schipluidenlaan 4
1062 HE Amsterdam
Phone +31 (0)20 6240055
Fax +31 (020) 4200075
email: info@datasim.nl
website: www.datasim.nl

DATASIM EDUCATION BV

I am interested in

- ☐ Open courses
- ☐ In-Company courses
- ☐ In-Company customized courses (2 days)
- ☐ Distance learning program

Please contact me:

Name: .....

Telephone: .....

Email: .....