1. Record Nr. UNINA9910831174303321

Autore	Fletcher S (Shayne)
Titolo	Financial modelling in Python [[electronic resource] /] / S. Fletcher & C. Gardner
Pubbl/distr/stampa	Chichester, : Wiley, 2009
ISBN	0-470-68500-X 1-282-88892-7 9786612888922 0-470-74789-7
Edizione	[1st edition]
Descrizione fisica	1 online resource (246 p.)
Collana	Wiley finance series
Altri autori (Persone)	GardnerChristopher
Disciplina	332.0285/5133 332.02855133
Soggetti	Finance - Mathematical models - Computer programs Python (Computer program language)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Financial Modelling in Python; Contents; 1 Welcome to Python; 1.1 Why Python?; 1.1.1 Python is a general-purpose high-level programming language; 1.1.2 Python integrates well with data analysis, visualisation and GUI toolkits; 1.1.3 Python 'plays well with others'; 1.2 Common misconceptions about Python; 1.3 Roadmap for this book; 2 The PPF Package; 2.1 PPF topology; 2.2 Unit testing; 2.2.1 doctest; 2.2.2 PyUnit; 2.3 Building and installing PPF; 2.3.1 Prerequisites and dependencies; 2.3.2 Building the C++ extension modules; 2.3.3 Installing the PPF package; 2.3.4 Testing a PPF installation 3 Extending Python from C++3.1 Boost.Date Time types; 3.1.1 Examples; 3.2 Boost.MultiArray and special functions; 3.3 NumPy arrays; 3.3.1 Accessing array data in C++; 3.3.2 Examples; 4 Basic Mathematical Tools; 4.1 Random number generation; 4.2 N(.); 4.3 Interpolation; 4.3.1 Linear interpolation; 4.3.2 Loglinear interpolation; 4.3.3 Linear on zero interpolation; 4.3.4 Cubic spline interpolation; 4.4 Root finding; 4.4.1 Bisection method; 4.4.2 Newton-Raphson method; 4.5 Linear algebra; 4.5.1 Matrix multiplication; 4.5.2 Matrix inversion; 4.5.3 Matrix pseudo-inverse

4.5.4 Solving linear systems4.5.5 Solving tridiagonal systems; 4.5.6 Solving upper diagonal systems; 4.5.7 Singular value decomposition; 4.6 Generalised linear least squares; 4.7 Quadratic and cubic roots; 4.8 Integration; 4.8.1 Piecewise constant polynomial fitting; 4.8.2 Piecewise polynomial integration; 4.8.3 Semi-analytic conditional expectations; 5 Market: Curves and Surfaces; 5.1 Curves; 5.2 Surfaces; 5.3 Environment; 6 Data Model; 6.1 Observables; 6.1.1 LIBOR; 6.1.2 Swap rate; 6.2 Flows; 6.3 Adjuvants; 6.4 Legs; 6.5 Exercises; 6.6 Trades; 6.7 Trade utilities
7 Timeline: Events and Controller7.1 Events; 7.2 Timeline; 7.3 Controller; 8 The Hull-White Model; 8.1 A component-based design; 8.1.1 Requestor; 8.1.2 State; 8.1.3 Filler; 8.1.4 Rollback; 8.1.5 Evolve; 8.1.6 Exercise; 8.2 The model and model factories; 8.3 Concluding remarks; 9 Pricing using Numerical Methods; 9.1 A lattice pricing framework; 9.2 A Monte-Carlo pricing framework; 9.2.1 Pricing non-callable trades; 9.2.2 Pricing callable trades; 9.3 Concluding remarks; 10 Pricing Financial Structures in Hull-White; 10.1 Pricing a Bermudan; 10.2 Pricing a TARN; 10.3 Concluding remarks
11 Hybrid Python/C++ Pricing Systems11.1 nth imm of year revisited; 11.2 Exercising nth imm of year from C++; 12 Python Excel Integration; 12.1 Black-scholes COM server; 12.1.1 VBS client; 12.1.2 VBA client; 12.2 Numerical pricing with PPF in Excel; 12.2.1 Common utilities; 12.2.2 Market server; 12.2.3 Trade server; 12.2.4 Pricer server; Appendices; A Python; A.1 Python interpreter modes; A.1.1 Interactive mode; A.1.2 Batch mode; A.2 Basic Python; A.2.1 Simple expressions; A.2.2 Built-in data types; A.2.3 Control flow statements; A.2.4 Functions; A.2.5 Classes; A.2.6 Modules and packages
A.3 Conclusion

Sommario/riassunto

""Fletcher and Gardner have created a comprehensive resource that will be of interest not only to those working in the field of finance, but also to those using numerical methods in other fields such as engineering, physics, and actuarial mathematics. By showing how to combine the high-level elegance, accessibility, and flexibility of Python, with the low-level computational efficiency of C++, in the context of interesting financial modeling problems, they have provided an implementation template which will be useful to others seeking to jointly optimize the use of computational and human r