The QAM Library

A Gauss Implementation for Quantitative Asset Management Modelling

Thierry Roncalli

University of Évry

This version: September 20, 2010

Contents

Chapter 1

Introduction

1.1 Installation

1. The file *gauss-qam2.zip* is a zipped archive file. Copy this file under the root directory of Gauss, for example **D:\GAUSS60**.

2. Unzip the file. Directories will then be created and files will be copied over them:

target_path	*readme.txt*
target_path\dlib	DLLs
target_path\lib	library file
target_path\qam\\[**...**]	example and tutorial files
target_path\qam\src	source code files
target_path\src	source code files

3. If your root of Gauss is **D:\GAUSS60**, the installation is finished, otherwise you have to modify the paths of the library using notepad or the LibTool. Another way to update the library is to run Gauss, l**og on to the *qam\src* directory**, delete the path with the command **lib qam -n** and add the path to the library with the command **lib qam -a**.

1.2 Getting started

Gauss 6.0.57+ for Windows and the library **optmum** are required to use the **QAM** routines.

1.2.1 readme.txt file

The file *readme.txt* contains last minute information on the **QAM** procedures. Please read it before using them.

1.2.2 Setup

In order to use these procedures, the **QAM** library must be active. This is done by including **QAM** in the LIBRARY statement at the top of your program:

```
library qam;
```

To reset global variables in subsequent executions of the program and in order to load DLLs, the following instruction should be used:

```
qamSet;
```

1.3 What is QAM ?

QAM is a Gauss library designed to accompany the French book ”La Gestion d’Actifs Quantitative” [1].

QAM contains the procedures whose list is given below.

1. **Backtest Computing**

   (a) **Add_Fees**: Adds managing and performance fees (yearly basis).

   (b) **Compute_Basktest**: Computes the backtest of a basket of strategies with given rebalancing dates.

   (c) **Compute_Basktest_With_TC**: Computes the backtest of a basket of strategies with transaction costs.

   (d) **Compute_Capitalized_Eonia**: Computes the backtest of an investment in Eonia (or Libor).

   (e) **Compute_Capitalized_Eonia_Plus**: Computes the backtest of a non-risky investment Eonia + x bp (or Libor + x bp).

   (f) **Compute_Global_Reporting**: Computes the statistics of a fund.

   (g) **Compute_Leverage_Strategy**: Leverages a strategy.

   (h) **Compute_Loss_Function**: Computes the loss function.

   (i) **Compute_Maximum_Drawdown**: Computes the maximum drawdown.

   (j) **Compute_Monthly_Basktest**: Computes the backtest of a basket of strategies (with a rebalancing at the end of the month).

   (k) **Compute_Monthly_Basktest2**: Computes the backtest of a basket of funded and non-funded strategies (with a rebalancing at the end of the month).

   (l) **Compute_Monthly_Statistics**: Computes monthly return and volatility.

   (m) **Compute_Reporting**: Computes the main statistics of a fund (return, volatility, Sharpe ratio, Information ratio, MDD, skewness and kurtosis).

   (n) **Compute_Running_Covariance**: Estimates the covariance matrix using a moving window.

   (o) **Compute_Running_Statistics**: Estimates the performance or the volatility using a moving window.

   (p) **Compute_Weekly_Returns**: Computes weekly returns.

   (q) **Currency_Hedging**: Performs currency hedging.

   (r) **Delete_Fees**: Deletes managing and performance fees (yearly basis).

   (s) **Excel_Reporting**: Computes the reporting of a fund in Excel format.

2. **Optimization Methods**

(a) **Bellman_Optimize_Tree**: Solves the Bellman problem in discrete time.

(b) **Bilinear_Interpolation**: Computes bilinear interpolation.

(c) **Bisection**: Performs the bi-section algorithm.

(d) **Bisection_Path**: Stores the path of the bi-section algorithm.

(e) **Black_Litterman_mu**: Computes the parameter μ_{cond} in the Black-Litterman model.

(f) **Black_Litterman_Pi**: Computes the parameter π in the Black-Litterman model.

(g) **Black_Litterman_Solve**: Solves the Black-Litterman model fro a given value of τ.

(h) **Black_Litterman_Solve2**: Solves the Black-Litterman model for a given tracking-error.

(i) **Compute_Constrained_Portfolio**: Computes the constrained portfolio using QP algorithm.

(j) **Compute_ERC_Portfolio**: Computes the ERC portfolio.

(k) **Compute_Diversification_Ratio**: Computes the diversification ratio $D(x)$.

(l) **Compute_MDP_Portfolio**: Computes the MDP portfolio.

(m) **Compute_Risk_Contribution**: Computes the risk contribution decomposition of a portfolio.

(n) **Lprog**: Solves a linear programming problem using an interior-point algorithm.

(o) **Optimize_Omega**: Optimizes the Ω measure.

(p) **PrintBellman**: Prints the optimal tree of the Bellman problem.

(q) **Qprog_Allocation_Solve**: Solves the portfolio allocation problem (ϕ, μ and σ problems).

(r) **Qprog_Allocation_Solve_With_TC**: Solves the portfolio allocation problem with transaction costs.

(s) **Qprog_Index_Sampling**: Performs a sampling of an equity index (or a benchmark).

(t) **Qprog_Index_Solve**: Solves the enhanced indexing problem.

(u) **Qprog_Index_130_30**: Solves the 130/30 indexing problem.

(v) **Qprog_Min_Variance**: Computes the Minimum variance portfolio.

(w) **Qprog_Sharpe Maximize**: Optimizes the Sharpe ratio.

(x) **Qprog_TE_Solve**: Solve the tracking-error problem.

(y) **Quadratic_Interpolation**: Computes mixed linear-quadratic interpolation.

(z) **regKernelQuantile**: Non-parametric quantile regression.

(aa) **regQuantile**: Linear quantile regression.

(ab) **regQP**: Linear regression using quadratic programming.

(ac) **regSharpeStyle**: Sharpe style analysis.

(ad) **Risk_Budgeting_Solve**: Solves the risk budegting allocation problem.

(ae) **Utility_cara**: Computes the CARA utility function.

(af) **Utility_crra**: Computes the CRRA utility function.

(ag) **Utility_ln**: Computes the logarithmic utility function.

3. **Numerical Algorithms**

(a) **BDFS_ZeroCoupon**: Solves the BDFS yield curve model.

(b) **ConstantCorrelation**: Defines a constant correlation matrix.

(c) **cosMatrix**: Computes the matrix cosine.

(d) **Compute_Definite_Correlation**: Estimates a definite correlation matrix.

(e) **Compute_Nearest_Correlation**: Estimates the nearest correlation matrix.

(f) **Compute_Discrete_Simplex**: Discretization of the simplex set.

(g) **Compute_Markov_Generator**: Computes the Markov generator of a one-year transition matrix.

(h) **Compute_Nearest_Correlation**: Computes the nearest correlation matrix.

(i) **Compute_Ponzi_Model**: Computes the Ponzi model.

(j) **ConstantCorrelation**: Generates a constant correlation matrix $C_n(\rho)$.

(k) **Estimate_Markov_Generator**: Estimates a valid Markov generator.

(l) **expMatrix**: Computes the matrix exponential.

(m) **funcMatrix**: Computes a general matrix function.

(n) **gaussHermite**: Computes weights and nodes of Hermite quadrature rules.

(o) **gaussJacobi**: Computes weights and nodes of Jacobi quadrature rules.

(p) **gaussLaguerre**: Computes weights and nodes of Laguerre quadrature rules.

(q) **gaussLegendre**: Computes weights and nodes of Legendre quadrature rules.

(r) **lnMatrix**: Computes the matrix logarithm.

(s) **ODE_Adams_Bashforth**: Solves a system of ordinary differential equations using the Adams-Bashforth algorithm.

(t) **ODE_Adams_Moulton**: Solves a system of ordinary differential equations using the Adams-Moulton algorithm.

(u) **ODE_Euler**: Solves a system of ordinary differential equations using the Euler algorithm.

(v) **ODE_Runge_Kutta**: Solves a system of ordinary differential equations using the Runge-Kutta algorithm.

(w) **quadHermite1**: Integrates a 1D function using Gauss-Hermite quadrature.

(x) **quadHermite2**: Integrates a 2D function using Gauss-Hermite quadrature.

(y) **quadLaguerre1**: Integrates a 1D function using Gauss-Laguerre quadrature

(z) **quadLaguerre2**: Integrates a 2D function using Gauss-Laguerre quadrature..

(aa) **quadLegendre1**: Integrates a 1D function using Gauss-Legendre quadrature.

(ab) **quadLegendre2**: Integrates a 2D function using Gauss-Legendre quadrature.

(ac) **quadLegendre3**: Integrates a 3D function using Gauss-Legendre quadrature.

(ad) **random_LC**: Uniform LC generator.

(ae) **regFLS**: Flexible least squares.

(af) **regLDP**: Linear dependency analysis.

(ag) **regPCA**: Principal component analysis.

(ah) **regSpline**: Spline interpolation and smoothing.

(ai) **rndmn**: Simulates normal random vectors using the Cholesky decomposition.

(aj) **rndmn_eig**: Simulates normal random vectors using the Eigenvalue decomposition.

(ak) **rndmn_svd**: Simulates normal random vectors using the SVD decomposition.

(al) **rndn_Box_Muller**: Simulates normal random numbers using the Box-Muller algorithm.

(am) **rndu_Halton**: Simulates quasi random numbers using Halton sequences.

(an) **rndu_Hammersley**: Simulates quasi random numbers using Hammersley sequences.

(ao) **Simpson1**: Integrates a 1D function using Simpson rules.

(ap) **simulate_Brownian_Bridge**: Simulates a Brownian bridge.

(aq) **simulate_Correlation**: Simulates a random correlation matrix using the random orthogonal algorithm.

(ar) **simulate_Correlation2**: Simulates a random correlation matrix using the nearest correlation algorithm.

(as) **simulate_GBM**: Simulates a GBM process.

(at) **simulate_SDE**: Simulates a SDE process.

(au) **sinMatrix**: Computes the matrix sine.

(av) **sqrtMatrix**: Computes the matrix square root.

(aw) **TDG_Solve**: Solves a tradiagonal system.

4. **Statistical Tools**

(a) **ComputeCenteredMoment**: Computes the centered moment.

(b) **ComputeKurtosis**: Computes the kurtosis coefficient.

(c) **ComputeSkewness**: Computes the skewness coefficient.

(d) **Compute_Gini**: Computes the Gini coefficient.

(e) **Compute_Lorenz**: Computes the Lorenz curve.

(f) **Compute_Shannon_Entropy**: Computes the Shannon entropy.

(g) **Compute_Shannon_Entropy_MC**: Computes the Shannon entropy of markov chains.

(h) **regCorr1**: Estimates the 1F correlation model.

(i) **regCorr2**: Estimates the multi-factor correlation model.

(j) **regCLS**: Estimates the parameters by the method of conditional least squares.

(k) **regFactorModel**: Estimates the factor model.

(l) **regGMM**: Estimates the parameters by the generalized method of moments.

(m) **regHuber**: Estimates the parameters by the Huber robust method.

(n) **regKernel**: Estimates the model by the non-parametric Kernel method.

(o) **regKernelDensity**: Estimates the probability density function by the Gaussian Kernel method.

(p) **regLAD**: Estimates the parameters by the robust method of least absolute deviations.

(q) **regLogit**: Estimates the Logit model by ML.

(r) **regMars**: Estimates the MARS model of Friedman.

(s) **regMarsForecast**: Forecasting with Mars model.

(t) **regML**: Estimates the parameters by the method of maximum likelihood.

(u) **regNLS**: Estimates the parameters by the method of non-linear least squares.

(v) **regOLS**: Estimates the parameters by the method of ordinary least squares.

(w) **regOU**: Estimates the parameters of the Ornstein-Uhlenbeck process by ML.

(x) **regProbit**: Estimates the Probit model by ML.

(y) **regQR**: Estimates the parameters by the robust quantile regression method.

(z) **regRestrict**: Fixes parameters in regression models (regGMM, regML, regNLS, regOLS).

(aa) **regRobust**: Estimates the parameters by the general robust method (M estimation).

(ab) **regTobit**: Estimates the Tobit model by ML.

(ac) **Scoring_Curve**: Computes the scoring curves (performance, discrimination and selection) and the Gini index.

(ad) **Scoring_Distribution**: Computes the scoring functions $\mathbf{F}(s)$, $\mathbf{F}_0(s)$ and $\mathbf{F}_1(s)$.

(ae) **vcx_cc**: Estimates the covariance matrix using the constant correlation method.

(af) **vcx_cc_shrinkage**: Estimates the covariance matrix using the constant correlation / shrinkage method.

(ag) **vcx_factor**: Estimates the covariance matrix using the factor method.

(ah) **vcx_factor_shrinkage**: Estimates the covariance matrix using the shrinkage method.

(ai) **vcx_rmt**: Estimates the covariance matrix using the RMT method.

5. **Time Series Analysis**

(a) **compute_autocorrelation**: Computes the autocorrelation function.

(b) **compute_autocovariance**: Computes the autocovariance function.

(c) **Compute_EWMA_Volatility**: Estimates the volatility using the exponential-weighted moving average method.

(d) **Garch_Vovol**: Computes the vovol measure of a GARCH process.

(e) **IGarch1_Vovol**: Computes the vovol measure of an integrated GARCH(1,1) process.

(f) **regGarch**: Estimates the GARCH(p,q) model.

(g) **regHurst**: Estimates the Hurst exponent.

(h) **regIGarch**: Estimates the integrated GARCH(p,q) model.

(i) **regIGarch1**: Estimates the integrated GARCH(1,1) model.

(j) **regRLS**: Estimates the parameters using recursive least squares.

(k) **spectral_cycle_gtest**: Computes the g Fisher's test.

(l) **spectral_density_arfima**: Computes the spectral density of an ARFIMA process.

(m) **spectral_density_arma**: Computes the spectral density of an ARMA process.

(n) **spectral_density_bsm**: Computes the spectral density of a basic structural model.

(o) **spectral_density_cycle**: Computes the spectral density of a stochastic cycle model.

(p) **spectral_density_ll**: Computes the spectral density of a local linear model.

(q) **spectral_density_llt**: Computes the spectral density of a local linear trend model.

6. **Quantitative Strategies**

(a) **Barbell_Calibrate**: Calibrates a barbell strategy.

(b) **Bond_Compute_Duration**: Computes the duration of a bond.

(c) **Bond_Compute_mDuration**: Computes the modified duration of a bond.

(d) **Bond_Compute_Price**: Computes the price of a bond.

(e) **Bond_Compute_YTM**: Computes the yield-to-maturity.

(f) **BS_Call**: Computes the call option price.

(g) **BS_CallSpread**: Computes the call-spread option price.

(h) **BS_Put**: Computes the put option price.

(i) **BS_PutSpread**: Computes the put-spread option price.

(j) **BS_Straddle**: Computes the straddle option price.

(k) **BS_Straddle_Delta**: Computes the delta of a straddle option.

(l) **BS_Straddle_Gamma**: Computes the gamma of a straddle option.

(m) **BS_Straddle_Theta**: Computes the theta of a straddle option.

(n) **BS_Straddle_Vega**: Computes the vega of a straddle option.

(o) **BullSpread_Payoff**: Computes the payoff of a bull-spread strategy.

(p) **CallSpread_Payoff**: Computes the payoff of a call-spread option.

(q) **compute_average_volatility**: Computes the average volatility of the basket.

(r) **compute_basket_volatility**: Computes the volatility of the basket.

(s) **Compute_Dynamic_Delta_Hedging**: Computes the 1D dynamic delta hedging of a derivatives portfolio.

(t) **Compute_Dynamic_Delta_Hedging2**: Computes the 2D dynamic delta hedging of a derivatives portfolio.

(u) **compute_EMN_Portfolio**: Calibrates an equity market neutral portfolio using the ERC method.

(v) **compute_Implied_Correlation**: Computes the implied correlation of the basket.

(w) **compute_Risk_Contribution_LS**: Computes the risk contribution of a Long/Short portfolio.

(x) **Compute_Turnover**: Computes the turnover of a strategy.

(y) **core_satellite_cdf**: Computes the cdf of C_t in a core-satellite strategy.

(z) **core_satellite_compute**: Computes the terminal value of C_t in a core-satellite strategy.

(aa) **core_satellite_pdf**: Computes the pdf of C_t in a core-satellite strategy.

(ab) **core_satellite_simulate**: Simulates a core-satellite strategy.

(ac) **CoveredCall_Payoff**: Computes the payoff of a covered-call strategy.

(ad) **CoveredCall_Simulate**: Simulates a covered-call strategy.

(ae) **CoveredCall_mSimulate**: Simulates a covered-call strategy.

(af) **cppi_cdf**: Computes the cdf of C_t in a cppi strategy.

(ag) **cppi_compute**: Computes the terminal value of C_t in a cppi strategy.

(ah) **cppi_pdf**: Computes the pdf of C_t in a cppi strategy.

(ai) **cppi_simulate**: Simulates a CPPI strategy.

(aj) **NelsonSiegel_ForwardRate**: Compute the forward rate in the Nelson-Siegel yield curve model.

(ak) **NelsonSiegel_SpotRate**: Compute the spot rate in the Nelson-Siegel yield curve model.

(al) **NelsonSiegel_ZeroCoupon**: Compute the bond price in the Nelson-Siegel yield curve model.

(am) **PutSpread_Payoff**: Computes the payoff of a put-spread option.

(an) **Simulate_stopLoss_Strategy**: Computes a stop-loss strategy.

(ao) **SpreadOption_BS**: Computes the spread option price.

(ap) **Straddle_Payoff**: Computes the payoff of a straddle option.

(aq) **VarianceSwap_DailyPayoff**: Computes the daily payoff of a variance swap.

(ar) **VarianceSwap_Payoff**: Computes the payoff of a variance swap.

(as) **VolatilitySwap_Payoff**: Computes the payoff of a volatility swap.

1.4 Using Online Help

QAM library supports Windows Online Help. Before using the browser, you have to verify that the **QAM** library is activated by the `library` command.

Chapter 2

Quantitative Asset Management

see [1].

Chapter 3

Examples

Some programs require the following Gauss library:

- gWizard
- nnet
- option
- pde2d
- pf
- tsm

All these libraries (except TSM) are available in the web page :

`http://www.thierry-roncalli.com/#gauss`

Remark 1 *The QAM library contains a new version of the PF library which includes the Pitt-Shephard Auxiliary Particle Filter and the Liu-West Particle Filter. All these procedures have been implemented by Guillaume Weisang.*

3.1 Examples in the *backtest* directory

1. **backtest1.prg**
   Performs a backtest example (Chapter 1, Page 47, Tables 1 to 5).

2. **backtest2.prg**
   An example of adding managing and performance fees (Chapter 1, Page 52, Table 6).

3. **backtest3.prg**
   Impact of performance fees on the backtest (Chapter 1, Page 54, Figure 1).

4. **backtest4.prg**
   Impact of performance fees on risk, return and Sharpe ratio (Chapter 1, Page 55, Figure 2).

5. **backtest5.prg**
   An example of currency hedging (Chapter 1, Page 57, Figure 3).

6. **backtest6.prg**
   Computes the backtest of a leverage strategy (Chapter 1, Page 58, Table 7).
7. **backtest10.prg**
   Illustrates the volatility bias (Chapter 1, Page 64, Figure 4).
8. **backtest11.prg**
   Illustrates the volatility bias (Chapter 1, Page 64).
9. **backtest12.prg**
   Computes the maximum drawdown (Chapter 1, Page 66, Figure 5).
10. **backtest13.prg**
    Computes the loss function (Chapter 1, Page 66, Figure 6).
11. **backtest14.prg**
    Computes the reporting of a backtest (Chapter 1, Page 10, Tables 8 and 9).

3.2 Examples in the *optimization* directory

1. **bellman1.prg**
   Dynamic programming (Chapter 2, Page 144).
2. **bellman2.prg**
   Solves a dynamic program using Bellman principle (Chapter 2, Page 146, Figure 19).
3. **bellman3.prg**
   Solves the quadratic control program (Chapter 2, Page 149, Figure 20).
4. **bellman10.prg**
   Illustrates the long-term investment problem (Chapter 2, Page 156, Figure 21).
5. **bellman11.prg**
   Illustrates balanced funds (Chapter 2, Page 160).
6. **bellman12.prg**
   Calibrate risk aversion of balanced funds (Chapter 2, Page 162, Table 20).
7. **bellman13.prg**
   Calibrate risk aversion of balanced funds (Chapter 2, Page 162, Table 21).
8. **bellman14.prg**
   Calibrates the equity allocation of balanced funds (Chapter 2, Page 163, Figure 24).
9. **bellman15.prg**
   Computes equity and bond risk contributions in balanced funds (Chapter 2, Page 164, Figure 25).
10. **bellman20.prg**
    Computes the utility function in the liability-driven investment problem (Chapter 2, Page 159, Figure 22).
11. **bellman21.prg**
    Optimal solution of the liability-driven investment problem (Chapter 2, Page 159, Figure 23).

12. **bellman30.prg**
Computes the glide path of target date funds (Chapter 2, Page 165, Figure 26).

13. **bellman31.prg**
Computes the glide path of target date funds (Chapter 2, Page 166, Figure 27).

14. **lprog1.prg**
Estimation of skew-beta returns by quantile regression using linear programming (Chapter 2, Page 79, Figure 1).

15. **nonlin1.prg**
Illustrates the CRRA utility function (Chapter 2, Page 125, Figure 15).

16. **nonlin10.prg**
Solves a risk-budgeting allocation problem (Chapter 2, Page 129, Tables 13 and 14).

17. **nonlin11.prg**
Illustrates the diversification effect (Chapter 2, Page 131, Figure 16).

18. **nonlin12.prg**
Computes EW, MV, ERC and MDP portfolios (Chapter 2, Page 135, Tables 15-18).

19. **nonlin20.prg**
Computes the π vector of Black-Litterman problem (Chapter 2, Page 137).

20. **nonlin21.prg**
Solves the Black-Litterman problem (Chapter 2, Page 141, Table 19).

21. **nonlin22.prg**
Illustrates the solution of the Black-Litterman inverse problem (Chapter 2, Page 142, Figure 17).

22. **nonlin23.prg**
Illustrates the solution of the Black-Litterman inverse problem (Chapter 2, Page 142).

23. **nonlin24.prg**
Backtest of the Black-Litterman approach (Chapter 2, Page 143, Figure 18).

24. **qprog1.prg**
Interpretation of the Lagrange coefficients in the Qprog procedure (Chapter 2, Page 82, Figure 2).

25. **qprog2.prg**
Performs a Sharpe style regression (Chapter 2, Page 88, Table 2 and Figure 5).

26. **qprog3.prg**
Performs a Sharpe style regression (Chapter 2, Page 88, Table 3 and Figure 6).

27. **qprog4.prg**
Solves a minimum variance portfolio problem (Chapter 2, Page 93, Table 4).

28. **qprog5.prg**
Backtest of a minimum variance strategy on a basket of global asset classes (Chapter 2, Page 94, Figure 7).

29. **qprog6.prg**
    Solves a Markowitz allocation problem (Chapter 2, Page 96, Table 5 and Figure 8).

30. **qprog7.prg**
    Solves μ and σ-problems (Chapter 2, Page 97, Tables 6 and 7).

31. **qprog8.prg**
    Illustrates the capital market line (Chapter 2, Page 98, Figure 9).

32. **qprog9.prg**
    Computes the Sharpe ratio (Chapter 2, Page 100, Figure 10).

33. **qprog10.prg**
    Optimizes the Sharpe ratio (Chapter 2, Page 99, Table 8).

34. **qprog11.prg**
    Calibrates Long/Short portfolios with a volatility target (Chapter 2, Page 103, Table 9).

35. **qprog20.prg**
    Solves an enhanced indexing problem (Chapter 2, Page 105, Table 10).

36. **qprog21.prg**
    Computes the efficient frontier of the enhanced indexing problem (Chapter 2, Page 107, Figure 11).

37. **qprog22.prg**
    Computes the optimal information ratio (Chapter 2, Page 107, Figure 12).

38. **qprog23.prg**
    Computes the sampling of the S&P 500 stock index (Chapter 2, Page 110, Figure 13).

39. **qprog24.prg**
    In this program, we compare the efficiency of the sampling method on S&P 500 and CAC 40 indices (Chapter 2, Page 111, Figure 14).

40. **qprog25.prg**
    Solves a 130/30 indexing problem (Chapter 2, Page 113, Table 11).

41. **qprog30.prg**
    Impact of transaction costs (Chapter 2, Page 114).

42. **qprog31.prg**
    Solves a portfolio allocation problem with transaction costs (Chapter 2, Page 117, Table 12).

43. **qprog40.prg**
    Computes constrained portfolios (Chapter 2, Page 83, Table 1).

44. **qprog41.prg**
    Computes bi-linear and quadratic interpolations (Chapter 2, Page 85, Figure 3).

45. **qprog42.prg**
    Computes bi-linear and quadratic interpolations (Chapter 2, Page 85, Figure 4).

3.3 Examples in the *numerics* directory

1. **approxim1.prg**
   Computes a discretization of the simplex set (Chapter 3, Page 194).

2. **approxim2.prg**
   Solves a portfolio allocation problem using discretization of the simplex set (Chapter 3, Page 195, Table 5).

3. **approxim3.prg**
   Spline interpolation and smoothing of a GBM process (Chapter 3, Page 197, Figure 8).

4. **approxim10.prg**
   Approximates a covariance matrix using the square root matrix decomposition (Chapter 3, Page 198).

5. **approxim11.prg**
   Computes the nearest correlation matrix (Chapter 3, Page 200).

6. **approxim12.prg**
   Simulates a correlation matrix (Chapter 3, Page 246).

7. **approxim13.prg**
   Computes the basket option price (Chapter 3, Page 247, Figure 30).

8. **approxim20.prg**
   Knots and weights of Gauss-Legendre quadratures (Chapter 3, Page 203, Figure 9).

9. **approxim21.prg**
   Gauss-Legendre numerical integration (Chapter 3, Page 203, Figure 10).

10. **approxim22.prg**
    Gauss-Legendre numerical integration (Chapter 3, Page 204, Figure 11).

11. **approxim23.prg**
    Knots and weights of Gauss-Laguerre quadratures (Chapter 3, Page 204).

12. **approxim24.prg**
    Computation of knots and weights using eigenvalue decomposition (Chapter 3, Page 205).

13. **approxim25.prg**
    Computation of the spread option price using 1D and 2D numerical quadratures (Chapter 3, Page 208, Table 7).

14. **approxim30.prg**
    Solves ODE systems using Euler, Runge-Kutta, Adams-Basforth and Adams-Moulton algorithms (Chapter 3, Page 211, Figure 12).

15. **approxim31.prg**
    Comparison of numerical ODE solutions (Chapter 3, Page 212, Figure 13).

16. **approxim32.prg**
    Numerical errors of ODE systems (Chapter 3, Page 212).

17. **approxim33.prg**
    Solves non-Cauchy problems (Chapter 3, Page 213).

18. **approxim34.prg**
Solves non-Cauchy problems (Chapter 3, Page 214, Figure 14).

19. **approxim35-37.prg**
Solves chaotic systems (Chapter 3, Page 216, Figures 15-17).

20. **approxim38.prg**
Solves the BDFS yield curve model (Chapter 3, Page 219, Figure 18).

21. **approxim39.prg**
Solves a Ponzi system (Chapter 3, Page 221, Figure 19).

22. **approxim50.prg**
Solves numerically the PDE of the Vasicek model (Chapter 3, Page 228, Figure 20).

23. **approxim51.prg**
Comparison of the densities of the Wiener process obtained by Feynman-Kac and Fokker-Planck algorithms (Chapter 3, Page 228).

24. **approxim52.prg**
Comparison of the densities of the GBM process obtained by Feynman-Kac and Fokker-Planck algorithms (Chapter 3, Page 230, Figure 22).

25. **approxim53.prg**
Comparison of the densities of the OU process obtained by Feynman-Kac and Fokker-Planck algorithms (Chapter 3, Page 229, Figure 21).

26. **approxim54.prg**
Computation of the density in the Heston model by solving the 2D Fokker-Planck PDE (Chapter 3, Page 231).

27. **approxim55.prg**
Computation of the density in the Heston model by solving the 2D Fokker-Planck PDE (Chapter 3, Page 232, Figure 23).

28. **approxim56.prg**
Computation of the density in the SABR model by solving the 2D Fokker-Planck PDE (Chapter 3, Page 232, Figure 24).

29. **lapack1.prg**
Eigenvalue decomposition (Chapter 3, Page 171).

30. **lapack2.prg**
Cholesky decomposition (Chapter 3, Page 171).

31. **lapack3.prg**
Comparison of eigenvalue and cholesky decompositions for simulating Gaussian random vectors (Chapter 3, Page 172).

32. **lapack4.prg**
Principal component analysis of the yield curve (Chapter 3, Page 174).

33. **lapack5.prg**
Computes the first 3 factors of the yield curve (Chapter 3, Page 177, Figure 1).

34. **lapack6.prg**
Schur and complex schur decomposition (Chapter 3, Page 177).

35. **lapack7.prg**
Exponential, logarithm, sine and cosine of a matrix (Chapter 3, Page 178).

36. **lapack8.prg**
Computes the Markov generator of the transition probability matrix of fund ratings (Chapter 3, Page 181).

37. **lapack9.prg**
Estimates the Markov generator using [IRW-1] and [IRW-2] algorithms (Chapter 3, Page 182).

38. **lapack10.prg**
Computes the transition probabilities (Chapter 3, Page 183, Figure 2).

39. **lapack11.prg**
Computes the transition probabilities (Chapter 3, Page 183, Figure 3).

40. **lapack12.prg**
Computes the transition probabilities (Chapter 3, Page 184, Figure 4).

41. **lapack13.prg**
Computes transition probability matrices (Chapter 3, Page 182).

42. **lapack14.prg**
Computes the persistence times (Chapter 3, Page 185, Figure 5).

43. **lapack15.prg**
Linear dependency analysis (Chapter 3, Page 186).

44. **lapack20.prg**
Storage comparison of band and dense matrices (Chapter 3, Page 187).

45. **lapack21.prg**
Speed comparison of band and dense matrices (Chapter 3, Page 188).

46. **lapack22.prg**
Band matrices and flexible least squares (Chapter 3, Page 192, Figures 6 and 7).

47. **mc1.prg**
Simulation of uniform random numbers (Chapter 3, Page 233).

48. **mc2.prg**
Comparison of Matlab and Gauss random generators (Chapter 3, Page 234, Table 10).

49. **mc3.prg**
Lattice structure of LC generators (Chapter 3, Page 235, Figure 25).

50. **mc4.prg**
Comparison of exact and euler schemes for the GBM process (Chapter 3, Page 240, Figure 26).

51. **mc5.prg**
Density of MC estimators (Chapter 3, Page 240, Figure 27).

52. **mc6.prg**
Simulates a Brownian bridge (Chapter 3, Page 243, Figure 28).

53. **mc7.prg**
Simulates a constrained GBM process using Brownian bridges (Chapter 3, Page 244, Figure 29).

54. **mc8.prg**
Simulates correlation matrices (Chapter 3, Page 246).

55. **mc9.prg**
Computes an upper bound of the spread option price using simulation methods (Chapter 3, Page 247, Figure 30).

56. **mc10.prg**
Computation of π by simulations (Chapter 3, Page 249, Figure 31).

57. **mc11.prg**
Non-parametric density of MC estimators (Chapter 3, Page 250, Figure 32).

58. **mc12.prg**
Computation of π by simulations (Chapter 3, Page 249).

59. **mc13.prg**
Convergence of MC estimators (Chapter 3, Page 251, Figure 33).

60. **mc14.prg**
Antithetic simulation of GBM processes (Chapter 3, Page 254, Figure 34).

61. **mc15.prg**
Non-parametric density of MC and MC+AV estimators (Chapter 3, Page 254, Figure 35).

62. **mc20.prg**
Comparison of LCG, Hammersley, Halton and Faure random generators (Chapter 3, Page 256, Figure 36).

63. **mc21.prg**
Illustration of the Sobol random generator (Chapter 3, Page 257, Figure 37).

64. **mc22.prg**
Projection of the 3D Faure random generator (Chapter 3, Page 256).

65. **mc23.prg**
Projection of several random generators on a sphere (Chapter 3, Page 257, Figure 38).

66. **mc24.prg**
Computation of the Spread option using QMC (Chapter 3, Page 258, Table 11).

3.4 Examples in the *statistics* directory

1. **ann1.prg**
Sigmoid functions (Chapter 4, Page 320, Figure 8).

2. **ann2.prg**
Graphic representation of the artificial neural network (Chapter 4, Page 320, Figure 9).

3. **ann3.prg**
Structure of artificial neural networks (Chapter 4, Page 321, Figure 10).

4. **ann4.prg**
Graphic representation of the dense ann (Chapter 4, Page 325, Figure 11).

5. **ann5.prg**
Graphic representation of the constrained ann (Chapter 4, Page 326, Figure 12).

6. **ann6.prg**
Partial R^2 and omission costs analysis (Chapter 4, Page 328, Figure 13).

7. **ann7.prg**
Graphic representation of the optimal ann (Chapter 4, Page 328, Figure 14).

8. **ann8.prg**
The XOR problem (Chapter 4, Page 329, Figure 15).

9. **ann9.prg**
The T-C problem (Chapter 4, Page 330, Figures 16 and 17).

10. **ann11.prg**
An example of classification (Chapter 4, Page 332, Figure 18).

11. **ann12.prg**
A more complex example of classification (Chapter 4, Page 333, Figure 19).

12. **cov1.prg**
Estimates the 1F correlation model (Chapter 4, Page 302, Figure 7).

13. **cov2.prg**
Compares the 1F and the multi-factor correlation models (Chapter 4, Page 301).

14. **cov3.prg**
Estimates a factor model (Chapter 4, Page 305).

15. **cov10.prg**
Estimates the correlation matrix using random matrix theory (Chapter 4, Page 310).

16. **cov11.prg**
Estimates the correlation matrix using shrinkage methods (Chapter 4, Page 312).

17. **cov20.prg**
Estimates the correlation matrix using copula methods (Chapter 4, Page 316).

18. **mars1.prg**
The Mars example of Friedman in *Annals of Statistics* (Chapter 4, Page 336).

19. **mars2.prg**
An example of Mars-Logit (Chapter 4, Page 336).

20. **mars3.prg**
An example of Mars modellinbg with the Stoxx 50 index (Chapter 4, Page 337).

21. **reg1.prg**
Robust estimation of beta stocks (Chapter 4, Page 267, Figure 1).

22. **reg2.prg**
Illustration of the EM algorithm (Chapter 4, Page 275).

23. **reg3.prg**
MCMC estimation using Gibbs sampling (Chapter 4, Page 291, Figure 2).

24. **reg4.prg**
Computes a frequency histogram (Chapter 4, Page 296, Figure 3).

25. **reg5.prg**
Estimates the pdf using the Kernel method (Chapter 4, Page 296, Figure 4).

26. **reg6.prg**
Compares the pdf of the order statistics using non-parametric and parametric models (Chapter 4, Page 297, Figure 5).

27. **reg7.prg**
Linear and quantile non-parametric regressions (Chapter 4, Page 299, Figure 6).

3.5 Examples in the *time_series* directory

1. **arfima1.prg**
AR coefficients of a fractional process (Chapter 5, Page 413, Figure 32).

2. **arfima2.prg**
Comparison of the AR(1) process and the fractional process (Chapter 5, Page 413, Figure 33).

3. **arfima3.prg**
Spectral density of the fractional process (Chapter 5, Page 414, Figure 34).

4. **arfima4.prg**
R/S analysis of the VIX index and the S&P 500 index (Chapter 5, Page 417, Figure 36).

5. **arfima5.prg**
Estimates the Hurst exponent using R/S analysis and time-frequency regression (Chapter 5, Page 417).

6. **spectral1.prg**
Time representation of the cycle model (Chapter 5, Page 397, Figure 22).

7. **spectral2.prg**
Correlogram of the cycle model (Chapter 5, Page 397, Figure 23).

8. **spectral3.prg**
Spectral representation of the cycle model (Chapter 5, Page 398, Figure 24).

9. **spectral4.prg**
Spectral density of ARMA processes (Chapter 5, Page 400, Figure 25).

10. **spectral5.prg**
Spectral density of LL, LLT and BSM processes (Chapter 5, Page 404, Figure 26).

11. **spectral6.prg**
Spectral density of the stochastic cycle model (Chapter 5, Page 404, Figure 27).

12. **spectral7.prg**
Spectral density of the ARFIMA process (Chapter 5, Page 415, Figure 35).

13. **spectral8.prg**
Estimates the spectral density using the periodogram and smoothing techniques (Chapter 5, Page 407, Figure 28).

14. **spectral9.prg**
Estimates the covariogram function using the periodogram (Chapter 5, Page 407, Figure 29).

15. **spectral10.prg**
Comparison of the time-domain and frequency-domain log-likelihood function (Chapter 5, Page 408).

16. **spectral11.prg**
Comparison of the time-domain and frequency-domain maximum likelihood estimators (Chapter 5, Page 408, Figure 30).

17. **spectral12.prg**
Spectral coherency of bivariate processes (Chapter 5, Page 410, Figure 31).

18. **spectral13.prg**
Estimates the cycle model using the periodogram technique (Chapter 5, Page 419, Figure 37).

19. **spectral14.prg**
Decomposition of the signal (Chapter 5, Page 421, Figure 38).

20. **spectral15.prg**
Reconstruction of the signal (Chapter 5, Page 421, Figure 39).

21. **spectral16.prg**
Illustrates the Parseval theorem (Chapter 5, Page 422, Table 7).

22. **sv1.prg**
Estimates the stochastic volatility model using Kalman filter (Chapter 5, Page 383, Figure 15 and Table 5).

23. **sv2.prg**
Estimates the stochastic volatility model using Griddy-Gibbs algorithm (Chapter 5, Page 389).

24. **sv3.prg**
Estimates the stochastic volatility model using the Random Walk Metropolis algorithm (Chapter 5, Page 389).

25. **sv4.prg**
Estimates the stochastic volatility model using the Metropolis-Hastings algorithm (Chapter 5, Page 390).

26. **sv5.prg**
Estimates the stochastic volatility model using the Griddy-Gibbs Metropolis-Hastings algorithm (Chapter 5, Page 390).

27. **sv6.prg**
Plots the estimated stochastic volatilities (Chapter 5, Page 393, Figure 18).

28. **sv7.prg**
Estimates the posterior density function of the parameters (Chapter 5, Page 393, Figure 19).

29. **sv8.prg**
Estimates the parameters of the stochastic volatility model (Chapter 5, Page 392, Table 6).

30. **sv9.prg**
Estimates the stochastic volatility model using the Liu-West particle filter (Chapter 5, Page 394, Figures 20 and 21).

31. **ts1.prg**
Simulates a cointegrated process (Chapter 5, Page 343, Figure 1).

32. **ts2.prg**
Performs recursive least squares (Chapter 5, Page 354, Figure 2).

33. **ts3.prg**
Performs recursive least squares (Chapter 5, Page 353).

34. **ts10.prg**
Performs Kalman filter (Chapter 5, Page 364, Table 3).

35. **ts11.prg**
Estimates the alternative beta model using Kalman filter in the case $P_0 = \mathbf{0}$ (Chapter 5, Page 367, Figure 3).

36. **ts12.prg**
Estimates the alternative beta model using Kalman filter in the case $P_0 \neq \mathbf{0}$ (Chapter 5, Page 368, Figure 4).

37. **ts13.prg**
Estimates the alternative beta model using Kalman filter in the case $\varepsilon_t = 0$ (Chapter 5, Page 368, Figure 5).

38. **ts14.prg**
Performance attribution between traditional beta, alternative beta and alpha (Chapter 5, Page 369, Figures 6 and 7).

39. **ts20.prg**
Estimates a non-linear state-space model using particle filters (Chapter 5, Page 376, Figures 10 and 11).

40. **ts21.prg**
Computes the probability density function of the SIS estimator (Chapter 5, Page 377, Figure 12).

41. **ts22.prg**
An example of dynamic asset allocation (Chapter 5, Page 374, Figure 8).

42. **ts23.prg**
Estimates the dynamic asset allocation model using Kalman filter, particle filter, SIS algorithm and SIR algorithm (Chapter 5, Page 374, Figure 9).

43. **ts30.prg**
Monthly returns of the S&P 500 index (Chapter 5, Page 378, Figure 13).

44. **ts31.prg**
ACF and PACF of r_t and r_t^2 (Chapter 5, Page 382, Figure 14).

45. **ts32.prg**
Estimates the GARCH(1,1) model (Chapter 5, Page 383, Figure 15 and Table 5).

46. **wavelet1.prg**
Time-frequency localisation (Chapter 5, Page 425, Figure 40).

47. **wavelet2.prg**
Morlet wavelet function (Chapter 5, Page 426, Figure 41).

48. **wavelet3.prg**
Mirror filters (Chapter 5, Page 429, Figure 42).

49. **wavelet4.prg**
Wavelet analysis of a non-stationnary signal (Chapter 5, Page 431, Figure 43).

50. **wavelet5.prg**
Periodogram of a non-stationnary signal (Chapter 5, Page 431, Figure 44).

51. **wavelet6.prg**
Threshold filtering (Chapter 5, Page 432, Figure 45).

52. **wavelet7.prg**
Sub-bands coding (Chapter 5, Page 433, Figure 46).

53. **wavelet8.prg**
Threshold filtering (Chapter 5, Page 432).

54. **wavelet9.prg**
Signal denoising with hard and soft shrinkage (Chapter 5, Page 434, Figures 47 and 48).

55. **wavelet10.prg**
Reproduces the denoising example of Donoho and Johnson (1994) (Chapter 5, Page 435, Figure 49).

56. **wavelet11.prg**
Fractal estimation using wavelets (Chapter 5, Page 436).

57. **wavelet12.prg**
Fractal estimation using wavelets (Chapter 5, Page 436).

58. **wavelet13.prg**
Comparison of the GPH estimator and the wavelets estimator of the fractional differencing parameter (Chapter 5, Page 436, Figure 50).

59. **wavelet14.prg**
Scalogram of time series (Chapter 5, Page 438, Figure 51).

3.6 Examples in the *strategy* directory

1. **carry1.prg**
   FX carry trade (Chapter 6, Page 511, Figure 48).
2. **carry2.prg**
   Performs a quarter selection of currencies (Chapter 6, Page 513, Tables 4 and 5).
3. **cppi1.prg**
   Computes the guarantee rate (Chapter 6, Page 445, Figure 1).
4. **cppi2.prg**
   Computes the guarantee rate G^+ and the initial cushion C_0 (Chapter 6, Page 446, Table 1).
5. **cppi3.prg**
   Computes the cushion C_T at maturity (Chapter 6, Page 448, Figure 2).
6. **cppi4.prg**
   Computes the pdf of S_T, C_T and V_T (Chapter 6, Page 450, Figure 3).
7. **cppi5.prg**
   Computes C_T with respect to S_T, σ and T (Chapter 6, Page 451, Figure 4).
8. **cppi6.prg**
   Simulates a CPPI strategy with $m = 5$ (Chapter 6, Page 453, Figure 5).
9. **cppi7.prg**
   Simulates a CPPI strategy with $m = 7$ (Chapter 6, Page 453, Figure 6).
10. **cppi8.prg**
    Computes the gap risk with respect to σ (Chapter 6, Page 455, Figure 7).
11. **cppi9.prg**
    Computes the gap risk with respect to $\mathrm{d}t$ (Chapter 6, Page 454).
12. **cppi10.prg**
    Computes the gap risk with respect to $\mathrm{d}t$ (Chapter 6, Page 454).
13. **cppi11.prg**
    Computes the gap risk with respect to $\mathrm{d}t$ (Chapter 6, Page 454, Figure 8).
14. **cppi12.prg**
    Computes the optimal multiple (Chapter 6, Page 457, Table 2).
15. **cppi13.prg**
    Computes the pdf of C_T in a core-satellite strategy (Chapter 6, Page 459, Figure 9).
16. **cppi14.prg**
    Comparison of V_t and V_t^{LS} (Chapter 6, Page 460, Figure 10).
17. **cppi15.prg**
    Simulates a core-satellite strategy (Chapter 6, Page 460, Figure 11).
18. **emn1.prg**
    Calibrates the portfolio of an equity market neutral strategy (Chapter 6, Page 533, Table 12).

19. **ir1.prg**
Computes the sport and forward rates with the Nelson-Siegel model (Chapter 6, Page 516, Figure 49).

20. **ir2.prg**
Illustrates the movements of the yield curve (Chapter 6, Page 517, Figure 50).

21. **ir3.prg**
Price, YTM and sensibility of the bond (Chapter 6, Page 518, Table 6).

22. **ir4.prg**
Impact of an interest-rate variation on the bond price(Chapter 6, Page 518, Table 7).

23. **ir5.prg**
Impact of an interest-rate variation on the bond price(Chapter 6, Page 518).

24. **ir11.prg**
Excess return of the roll-down strategy (Chapter 6, Page 521, Figure 51).

25. **ir12.prg**
Computes the return of the roll-down strategy (Chapter 6, Page 519).

26. **ir13.prg**
Computes the breakeven of the roll-down strategy (Chapter 6, Page 521, Figure 52).

27. **ir14.prg**
Excess return of the roll-down strategy with a swap investment (Chapter 6, Page 526, Figure 54).

28. **ir15.prg**
Computes the carry and roll-down decomposition (Chapter 6, Page 520, Table 8).

29. **ir16.prg**
Computes the carry and roll-down decomposition (Chapter 6, Page 520, Table 8).

30. **ir17.prg**
Computes the carry and roll-down decomposition (Chapter 6, Page 520, Table 8).

31. **ir18.prg**
Computes the return of the roll-down strategy (Chapter 6, Page 520).

32. **ir21.prg**
Computes the portfolio weights of the barbell strategy (Chapter 6, Page 523, Tables 9, 10 and Figure 524).

33. **ir21.prg**
Computes the PnL of the barbell strategy (Chapter 6, Page 525, Table 11).

34. **momentum1.prg**
Examples of the exposure function (Chapter 6, Page 536, Figure 55).

35. **momentum2.prg**
Performs the backtest of the trend-following benchmarked strategy (Chapter 6, Page 539, Figure 56).

36. **momentum3.prg**
Performs the backtest of the total return strategy (Chapter 6, Page 539, Figure 57).

37. **momentum4.prg**
Performs the backtest of the absolute return strategy (Chapter 6, Page 540, Figure 58).

38. **momentum10.prg**
Mean-reverting properties of the Ornstein-Uhlenbeck process (Chapter 6, Page 541).

39. **momentum11.prg**
Mean-reverting properties of the Ornstein-Uhlenbeck process (Chapter 6, Page 542, Figure 59).

40. **momentum12.prg**
Illustration of the contrarian strategy (Chapter 6, Page 544, Figure 60).

41. **momentum13.prg**
Calibrates the exposure function of the contrarian strategy (Chapter 6, Page 544, Figure 61).

42. **momentum14.prg**
Computes the Sharpe ratio with respect to the parameter a (Chapter 6, Page 545, Figure 62).

43. **option1.prg**
Computes the payoff of a long position in a call option (Chapter 6, Page 462, Figure 12).

44. **option2.prg**
Computes the delta of a call option (Chapter 6, Page 463, Figure 13).

45. **option3.prg**
Computes the exposure of an option strategy (Chapter 6, Page 463, Figure 14).

46. **option4.prg**
Computes the implied strike of a trend-following strategy (Chapter 6, Page 464, Figure 15).

47. **option5.prg**
Computes the payoff of a strangle option strategy (Chapter 6, Page 465, Figure 16).

48. **option6.prg**
Computes the payoff of a mean-reverting strategy (Chapter 6, Page 469, Figure 17).

49. **option7.prg**
Computes the payoff of a trend-following strategy (Chapter 6, Page 469, Figure 18).

50. **option8.prg**
Estimates the pdf of Gamma costs (Chapter 6, Page 470, Figure 19).

51. **option10.prg**
Performs the backtest of call/put option strategies (Chapter 6, Page 472, Figures 20 and 21).

52. **option11.prg**
Normalized probability distribution of the returns of call/put option strategies (Chapter 6, Page 473, Figure 22).

53. **option20.prg**
Computes the payoff of a covered-call strategy (Chapter 6, Page 474, Figure 23).

54. **option21.prg**
Comparison of the PnL of covered-call and buy-and-hold strategies (Chapter 6, Page 476, Figure 24).

55. **option22.prg**
Comparison of the PnL of covered-call and buy-and-hold strategies (Chapter 6, Page 476, Figure 25).

56. **option23.prg**
Volatility of the PnL of the covered-call strategy (Chapter 6, Page 477, Figure 26).

57. **option24.prg**
Density of the PnL of the covered-call strategy (Chapter 6, Page 478, Figure 27).

58. **option24.prg**
Comparison of BXM and BXY indexes (Chapter 6, Page 480, Figure 28).

59. **option30.prg**
Computes the payoff of a bull-spread strategy (Chapter 6, Page 481, Figure 29).

60. **option31.prg**
Computes the probability distribution of a bull-spread PnL (Chapter 6, Page 482, Figure 30).

61. **option32.prg**
Computes the implied strike (Chapter 6, Page 483, Figure 31).

62. **option33.prg**
Marked-to-market of the bull-spread strategy (Chapter 6, Page 484).

63. **option34.prg**
Backtest of the bull-spread strategy (Chapter 6, Page 485, Figure 32).

64. **rotation1.prg**
Performance of the Stoxx 600 index and sub-indexes (Chapter 6, Page 548, Figures 63 and 64).

65. **rotation2.prg**
Performance of the Stoxx 600 index and sub-indexes (Chapter 6, Page 549, Table 13).

66. **rotation3.prg**
Backtest of the 5 best performer Stoxx 600 sub-indexes (Chapter 6, Page 550, Figure 65).

67. **volatility1.prg**
Computes the payoff of the straddle strategy (Chapter 6, Page 486, Figure 33).

68. **volatility2.prg**
Computes the greeks of the straddle option (Chapter 6, Page 487, Figure 34).

69. **volatility3.prg**
Computes the theoretical relationship between the score and the PnL of the straddle strategy (Chapter 6, Page 489 Figure 35).

70. **volatility4.prg**
Backtest of the straddle strategy (Chapter 6, Page 490, Figures 36 and 37).

71. **volatility5.prg**
Backtest of the straddle strategy (Chapter 6, Page 489).

72. **volatility10.prg**
Compute the PnL of the variance swap (Chapter 6, Page 493, Figure 38).

73. **volatility11.prg**
Illustrates the behavior of the variance swap (Chapter 6, Page 494, Figure 39).

74. **volatility12.prg**
Convergence of the calls portfolio to the variance swap (Chapter 6, Page 496, Figure 40).

75. **volatility13.prg**
Spread between historical and implied volatilities (Chapter 6, Page 499, Figure 41).

76. **volatility14.prg**
SGI Volatility Premium (Chapter 6, Page 499, Figure 42).

77. **volatility15.prg**
Mean-reverting property of the historical-implied spread (Chapter 6, Page 500, Figure 43).

78. **volatility16.prg**
Computes the implied correlation of the Eurostoxx 50 index (Chapter 6, Page 503, Figure 44).

79. **volatility17.prg**
Historical simulation of dispersion trading(Chapter 6, Page 505, Figure 45).

80. **volatility20.prg**
VIX index (Chapter 6, Page 507, Figure 46).

81. **volatility21.prg**
Relationship between variations in the VIX index and variations in the S&P 500 index(Chapter 6, Page 508, Figure 47).

3.7 Examples in the *scoring* directory

1. **scoring1.prg**
Computes the Ornstein-Uhlenbeck score (Chapter 7, Page 557, Figure 1).

2. **scoring2.prg**
Aggregates two ranking scores (Chapter 7, Page 560, Figure 2).

3. **scoring3.prg**
Aggregates two ranking scores (Chapter 7, Page 561, Figure 3).

4. **scoring4.prg**
Computes the value of σ_S (Chapter 7, Page 562, Table 1).

5. **scoring5.prg**
Computes the Shannon entropy (Chapter 7, Page 564, Figure 4).

6. **scoring6.prg**
Compares two scoring systems using the Shannon entropy (Chapter 7, Page 567, Figure 5).

7. **scoring7.prg**
   Displays the selection curve of the first scoring system (Chapter 7, Page 571, Figure 6).

8. **scoring8.prg**
   Displays the selection curve of the second scoring system (Chapter 7, Page 571, Figure 7).

9. **scoring9.prg**
   Computes the selection curve of a good scoring system (Chapter 7, Page 573, Figure 8).

10. **scoring10.prg**
    Computes the critical values of the Kolmogorov-Smirnov statistic (Chapter 7, Page 575, Figure 9).

11. **scoring11.prg**
    Computes the Lorenz curve (Chapter 7, Page 572).

12. **scoring12.prg**
    Computes the ROC curve and the Gini coefficient (Chapter 7, Page 576, Figure 10 and Table 3).

13. **scoring13.prg**
    Compares financial performance and scoring performance (Chapter 7, Page 576).

14. **scoring14.prg**
    Compares financial performance and scoring performance (Chapter 7, Page 577, Figure 11).

3.8 Examples in the *risk* directory

1. **data1.prg**
   Impact of the data on the backtest (Chapter 8, Page 593, Figure 4).

2. **data2.prg**
   Impact of the data on the backtest (Chapter 8, Page 594, Figure 5).

3. **liquidity1.prg**
   Histogram of the bid-ask spread (Chapter 8, Page 602, Figure 6).

4. **liquidity2.prg**
   Daily volume (Chapter 8, Page 605, Figure 8).

5. **liquidity3.prg**
   Computes the $\mathfrak{R}$ measure (Chapter 8, Page 606, Figure 9).

6. **stoploss1.prg**
   Calibrates a stop loss strategy (Chapter 8, Page 588).

7. **stoploss2.prg**
   Simulates a stop loss strategy (Chapter 8, Page 590, Figure 1).

8. **stoploss3.prg**
   Computes the reporting of a stop loss strategy (Chapter 8, Page 589, Tables 1 and 2).

9. **stoploss4.prg**
   Exposure of a volatility target strategy (Chapter 8, Page 591, Figure 2).

10. **stoploss5.prg**
Simulates a volatility target strategy (Chapter 8, Page 591, Figure 3).

11. **turnover1.prg**
Computes the turnover of portfolio (Chapter 8, Page 598).

12. **turnover2.prg**
Computes the turnover of a monthly strategy (Chapter 8, Page 598, Table 3).

13. **turnover3.prg**
Computes the turnover of a weekly strategy (Chapter 8, Page 599, Table 4).

14. **turnover4.prg**
Computes the turnover of a constant-mix strategy (Chapter 8, Page 599, Tables 5 and 6).

15. **turnover5.prg**
Computes the turnover of the absolute return strategy (Chapter 8, Page 603).

16. **turnover6.prg**
Performs the backtest of the absolute return strategy with transaction costs (Chapter 8, Page 603, Figure 7).

17. **turnover7.prg**
Computes the transaction cost of the absolute return strategy (Chapter 8, Page 603).

3.9 Examples in the *pricing* directory

1. **ap-commodity1.prg**
Term structure of the crude oil futures (Appendix, Page 634, Figure 4).

2. **ap-commodity2.prg**
Contango and backwardation effects (Appendix, Page 635, Figure 5).

3. **ap-commodity3.prg**
Comparison of rolling methods on commodity futures (Appendix, Page 635, Figure 6).

4. **ap-option1.prg**
Call option pricing (Appendix, Page 637, Figure 7).

5. **ap-option2.prg**
Dynamic delta hedging with a negative final PnL (Appendix, Page 641, Table 5).

6. **ap-option3.prg**
Dynamic delta hedging with a positive final PnL (Appendix, Page 642, Table 6).

7. **ap-option4.prg**
Density of the PnL ratio (Appendix, Page 643, Figure 8).

8. **ap-option5.prg**
Reproduces the hedging example presented in the book of John Hull (Appendix, Page 640).

9. **ap-option6.prg**
Reproduces the hedging example presented in the book of John Hull (Appendix, Page 640).

10. **ap-option7.prg**
    Computes the hedging efficiency measure $\sigma(\pi)$ (Appendix, Page 644, Figure 9).

11. **ap-option10.prg**
    Computes the Marked-to-Market pricing with constant and time-varying volatilties (Appendix, Page 646, Tables 7 and 8).

12. **ap-option11.prg**
    Computes the Delta coefficients (Appendix, Page 648, Figure 10).

13. **ap-option12.prg**
    Computes the Gamma coefficients (Appendix, Page 648, Figure 11).

14. **ap-option13.prg**
    Volatility smile (Appendix, Page 650, Figure 12).

15. **ap-option14.prg**
    Computes the density of the hedging measure π (Appendix, Page 652, Figure 13).

16. **ap-option15.prg**
    Risk-neutral distribution and volatility smile (Appendix, Page 654, Figure 14).

17. **ap-option20.prg**
    Pricing with the binomial CRR model (Appendix, Page 656, Figure 15).

18. **ap-option21.prg**
    Calibration of the local volatility model (Appendix, Page 656, Figure 16).

19. **ap-option22.prg**
    Volatility smile of the Heston model (Appendix, Page 657, Figure 17).

20. **ap-option23.prg**
    Volatility smile of the SABR model (Appendix, Page 659, Figure 18).

21. **ap-option24.prg**
    Sensibility of the SABR volatility smile to the α, ν and ρ parameters (Appendix, Page 659, Figure 19).

22. **ap-option25.prg**
    The $\beta - \rho$ calibration problem in the SABR model (Appendix, Page 660, Figure 19).

Bibliography

[1] Roncalli, T [2010], La Gestion d'Actifs Quantitative, Economica.