RAISE PARTNER

Wall Risk Engine

Wall Risk Engine® User Guide

April 7, 2014

Contents

List of Figures

1 Introducing Wall Risk Engine®

1.1 What is Wall Risk Engine®?

1.1.1 Overview

Wall Risk Engine® is a software component composed of 5 modules:

- Risk and expert modeling: estimating the risk (volatilities, correlations,...) and the performance (expected returns) of the components of your investment universe;
- Risk and performance analysis: analyzing (ex-ante and ex-post) the risk and performance properties of your financial instruments and portfolios;
- Robust portfolio optimization: designing tailored, dynamic and robust allocation strategies;
- Simulation and stress-tests: Simulating future values of financial instruments (simple and structured products) and portfolios, computing large-scale risk measures (VaR) based on Monte-Carlo approaches.
- Data preparation: outlier detection and data completion functionalities to prepare the data for the risk modeling and risk analysis.

1.1.2 Main features

Risk and performance modeling

This module provides advanced functionalities to build tailored risk and performance models, using both historical data and exogenous information such as anticipations, market-implied information or in-house scores. Monitoring and estimating the correlations (or covariances) across all asset classes is a very sensitive issue especially within a portfolio optimization or a pricing process. The risk models provided in Wall Risk Engine® rely on 3 main pillars:

- Computing reactive and dynamic historical estimations of the correlations and volatilities (or the variance/covariance matrix);
- Taking into account exogenous information (if any) about the correlations and the volatilities;

- Applying a relevant filter to ensure the robustness of the risk model (by removing the noise) and merge historical and exogenous information.

Risk and performance analysis

This module embeds a wide range of ex-ante and ex-post analysis designed to monitor the risk and performance of a given portfolio:

- Performance measures;
- Risk measures: Volatility, semi-volatility, tracking-error, robust VaR, drawdown,...;
- Ratios: Sharpe, Sortino, Information, Omega;
- 3rd and 4th moments of the distribution: skewness, kurtosis;
- Risk and performance contributions.

Robust portfolio optimization

This module provides optimization functions to design tailored portfolios:

- Minimization of the risk of the portfolio given some performance constraint;
- Risk Budgeting: maximization of the expected return of the portfolio given some volatility budget constraints;
- Market portfolio (with maximum Sharpe ratio);
- Off-the-shelf adaptive allocation model to build reactive and robust portfolios;

All these functionalities embed patented robust optimization techniques and provide reliable and pertinent portfolios, ensuring a smooth behaviour with respect to market changes. These allocation models are compatible with any type of linear constraints on the weights such as sectorial constraints, asset-level constraints.

Simulation and stress-testing

This module provides tools to:

- simulate financial instruments using factorial or dense simulation models,
- build scenario simulations based on exogenous factors,
- stress-test a financial instrument or a portfolio,
- compute risk measures (Value at Risk) on large-scale portfolios based on Monte-Carlo simulations.

Quant data preparation

This module provides functionalities to prepare the market data for the risk modeling and analysis:

- detecting outliers in multi-dimensional time series,
- completing historical time series with robust regression approaches.

1.2 Installation

1.2.1 Technical requirements

System requirements

All the functions are available for:

- Win32/Win64 architecture on Intel processors : Dynamic link library (DLL)
- Linux platforms - Shared library (SO)

The components are designed to be highly efficient on all types of scalar machines (PC's, workstations). The components can be used and integrated in different types of applications such as:

- Win32/Win64 applications (.NET, .COM, Visual C++ 6.0, Visual Basic 6.0),
- spreadsheets in Microsoft Office applications such as Excel and Access (VBA),
- web-based applications (ASP, PHP, VB script, Java),
- client applications (Java, VB),
- high-performance computing, computation servers (Fortran 77/90, C/C++),
- quant research tools: Matlab, Scilab.

1.2.2 Installation procedure

The installation procedure depends on the chosen API. Please refer to the *Getting started with Wall Risk Engine®* guide delivered with the API.

1.2.3 License

Each installation of the Wall Risk Engine® library requires a license which is generated from the expiration date and the MAC (or physical) address of the user's computer. To find this MAC address, you need to open a DOS command window, type "ipconfig /all" and read the MAC address corresponding to your network connexion.

1.3 Documentation

The available documentation for Wall Risk Engine® includes:

- The Wall Risk Engine® User Guide describes practical information for developers who use Wall Risk Engine®.
- The Wall Risk Engine® Quant Guide provides detailed information about the quant models.
- For each API, we provide a *Getting started* guide (pdf) driving you through your first experience with Wall Risk Engine®. You will find the detailed installation procedure as well as simple examples to get familiar with the functionalities and the API.

- A technical manual and integration examples for each language (Java, C/C++, Visual C++, VB/VBA) providing functions description, instructions and annotated sample programs.

1.4 Support

Any bug report or question regarding the use of Wall Risk Engine® should be sent to the consultant in charge of your project at Raise Partner.

2 How To's

We present hereafter examples explaining how to organize Wall Risk Engine® functionalities in order to build allocation, backtesting, stress-testing and simulation processes.

2.1 Portfolio allocation process

The figure 2.1 describes how to articulate Wall Risk Engine® functionalities to implement a portfolio allocation process, from the computation of historical returns to the ex-ante analysis of the optimal portfolio:

- Step 1: Computation of the arithmetic historical returns from the historical prices with the *NORMmodelingReturns* function
- Step 2: Construction of the risk model:
  - Computation of the historical estimation of the covariance matrix given the chosen estimation window, even if the historical dataset is incomplete (cf. Risk and Performance Modeling module).
  - Calibration of the covariance matrix with additionnal views (explicit or interval) on the volatilities or correlations.
- Step 3: Construction of the performance model:
  - Computation of the historical mean returns (historical trend approach) given the chosen estimation window.
  - This step can be completed or replaced by any in-house performance modeling process (based on a scoring approach for instance).
- Step 4: Computation of the optimal portfolio based on the risk and performance models, the chosen strategy (minimum volatility, risk budgeting...), the set of strategy parameters and the user-defined constraints.
- Step 5: Ex-ante analysis of the optimal portfolio with the ex-ante functions of the Risk and Performance module.

2.2 Back-testing of an investment strategy

The figure 2.2 describes how to implement a back-testing process to analyze an investment strategy on real past scenarios.

At each historical rebalancing date, the returns corresponding to the rolling estimation window are extracted and fed into the risk and performance models. Then the optimal portfolio at this date is computed, and the portfolio is left untouched until the next rebalancing date.

At the end of the back-test, the performances of the simulated portfolio are recomposed and compared to the underlying instruments (or to any other benchmark) via an ex-post risk and performance analysis.

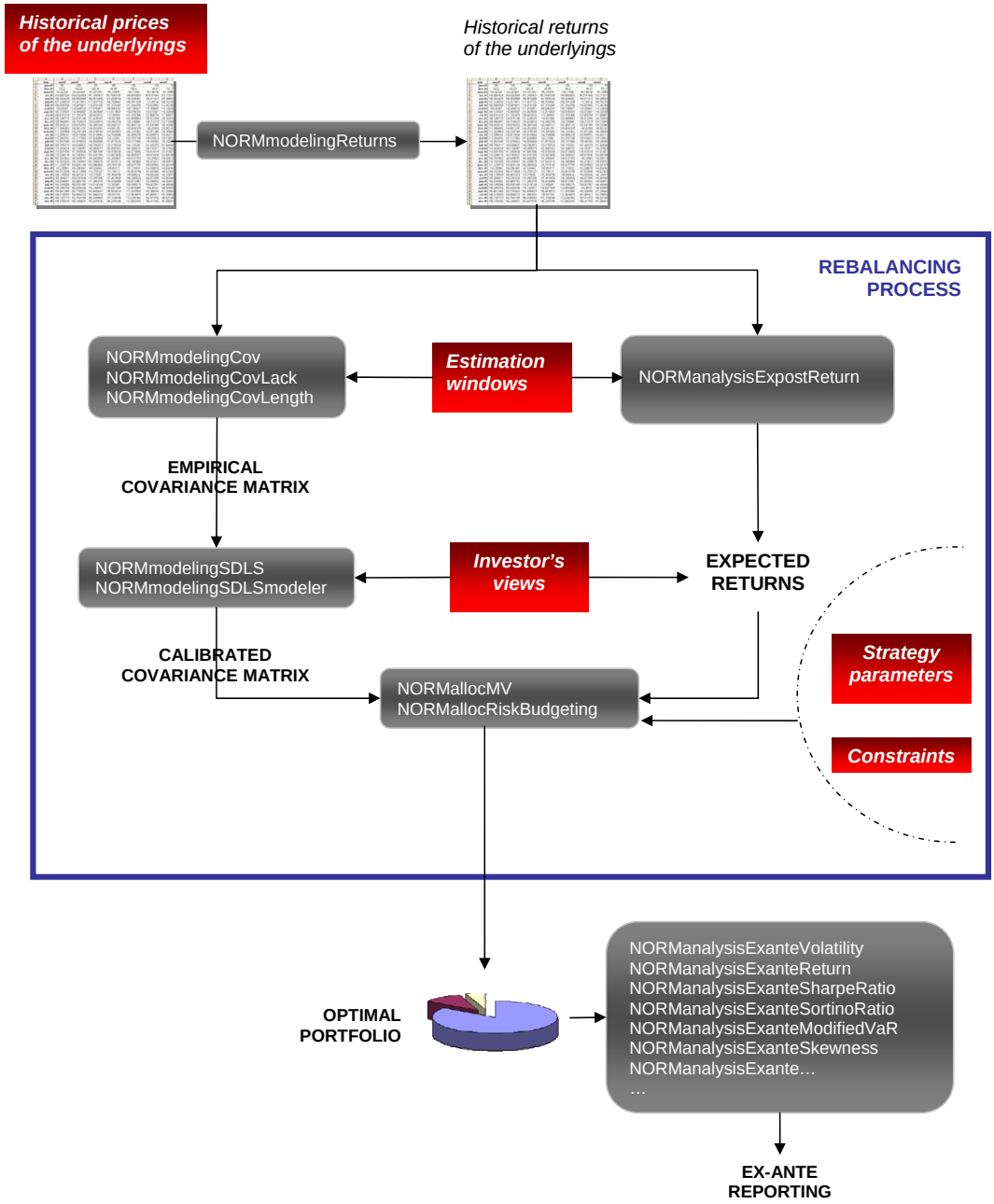

Figure 2.1: Portfolio rebalancing process with Wall Risk Engine® functions

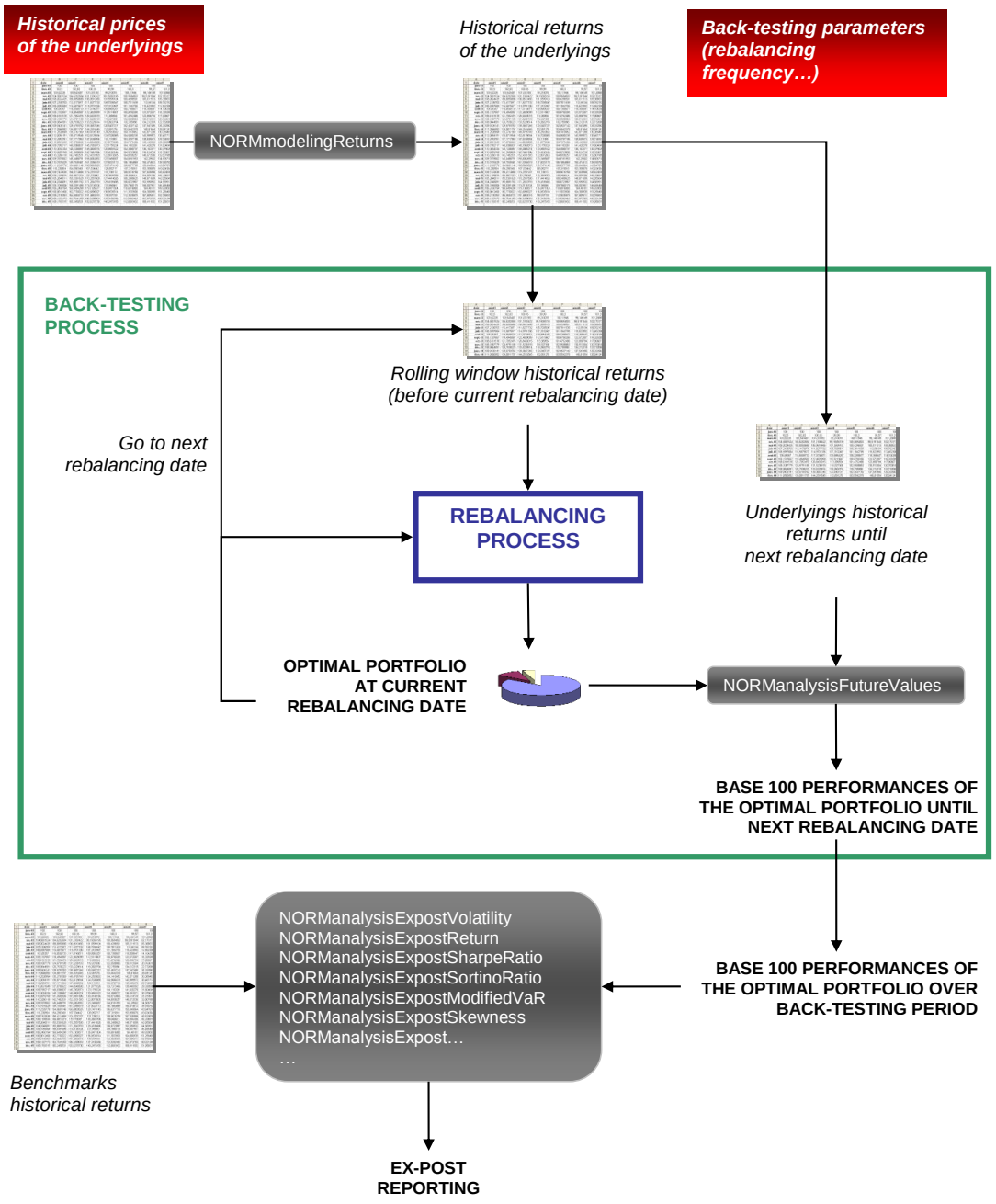

Figure 2.2: Back-testing process with Wall Risk Engine® functions

2.3 Simulating an analyze a complex financial instrument (such as a structured product)

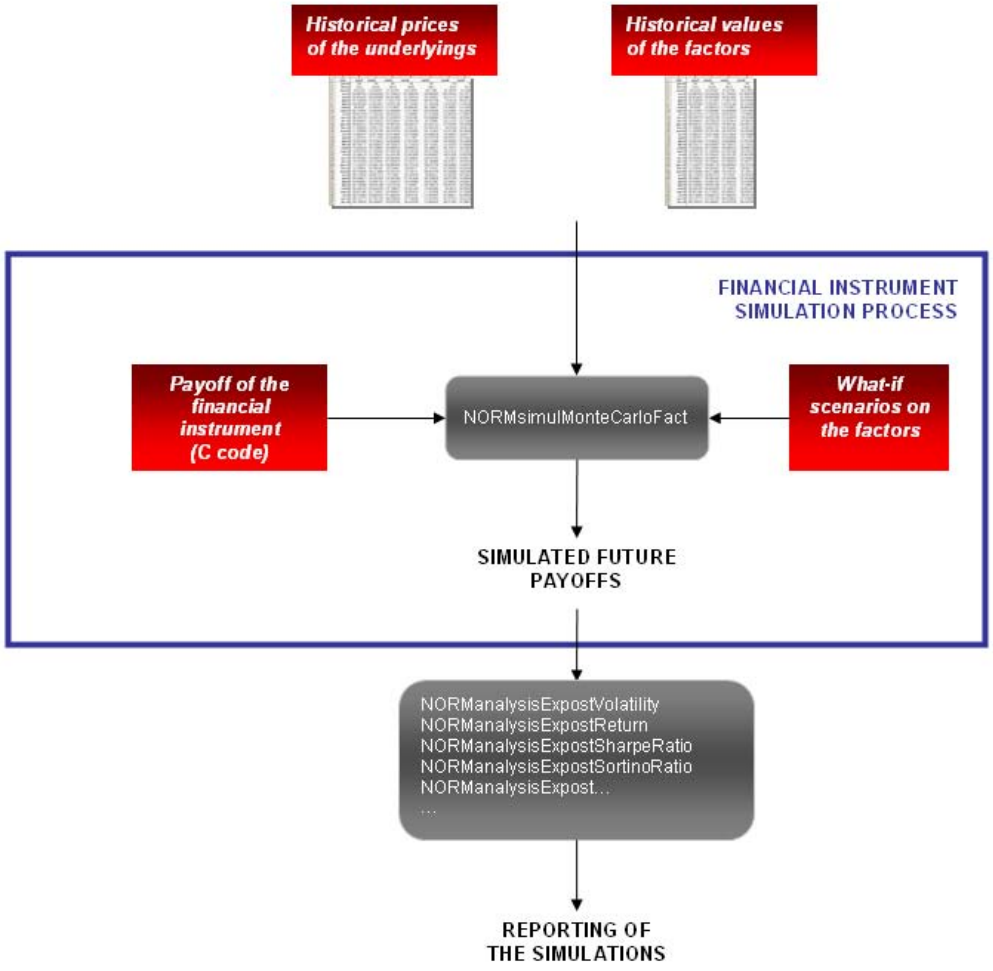

Figure 2.3: Financial instrument simulation process with Wall Risk Engine® functions

2.4 Building stress-testing scenarios for a portfolio

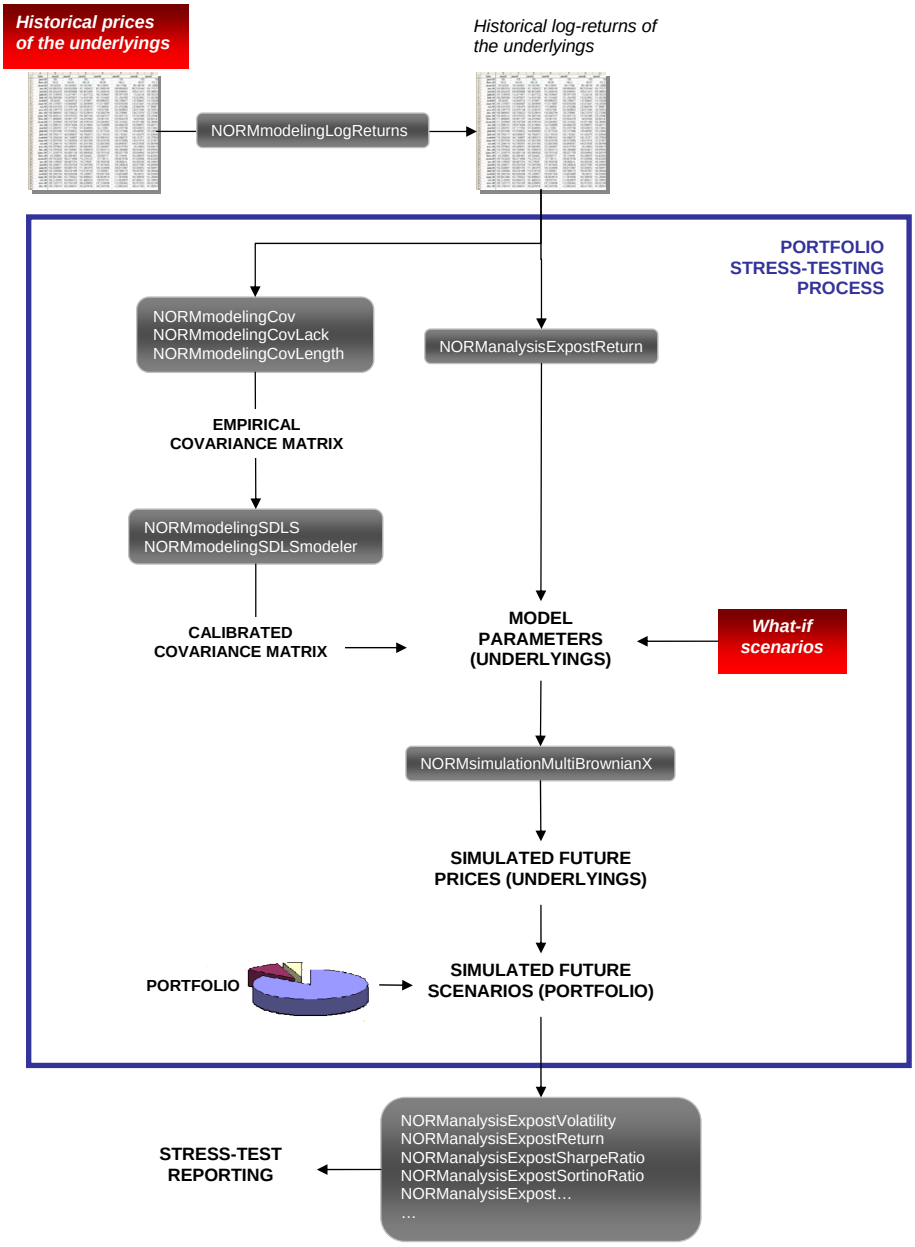

Figure 2.4: Portfolio stress-testing process with Wall Risk Engine® functions

Part I

Functions description

2.5 ENSIMAG module

2.5.1 WREallocIT - Index Tracking Minimization

Overview

This component computes the optimal portfolio which minimizing the tracking error vs. a benchmark.

Inputs

Variable	Type	Description
nbAssets	integer	number of asset(s) (¿1)
cov	double array of dimension nbAssets by nbAssets	covariance matrix
expectedReturns	double array of dimension nbAssets	mean return(s)
benchmarkCov	double array of dimension nbAssets	covariance of benchmark
benchmarkExpectedReturn	double	benchmark mean return supplied
nbEqConst	integer	number of equality constraints (¿=0)
nbIneqConst	integer	number of inequality constraints (¿=0)
C	double array of dimension nbAssets by nbEqConst+nbIneqConst	matrix of constraints. If no constraints are given, C=0
b	double array of dimension nbEqConst+nbIneqConst	vector of constraints. If no constraints are given, b=0
minWeights	double array of dimension nbAssets	lower bounds
maxWeights	double array of dimension nbAssets	upper bounds
relativeTargetReturn	double	target performance relative to the benchmark

Outputs

Variable	Type	Description
optimalWeights	double array of dimension nbAssets	optimal portfolio
info	integer array of dimension 1	diagnostic argument

2.5.2 WREallocMV - Mean-Variance (Markowitz)

Overview

This component computes the optimal portfolio with minimal volatility under a performance constraint (target return) and various linear constraints.

Inputs

Variable	Type	Description
nbAssets	integer	number of asset(s) (¿1)
cov	double array of dimension nbAssets by nbAssets	covariance matrix
expectedReturns	double array of dimension nbAssets	mean return(s)
nbEqConst	integer	number of equality constraints (¿=0)
nbIneqConst	integer	number of inequality constraints (¿=0)
C	double array of dimension nbAssets by nbEqConst+nbIneqConst	matrix of constraints. If no constraints are given, C=0
b	double array of dimension nbEqConst+nbIneqConst	vector of constraints. If no constraints are given, b=0
minWeights	double array of dimension nbAssets	lower bounds
maxWeights	double array of dimension nbAssets	upper bounds
targetReturn	double	target return

Outputs

Variable	Type	Description
optimalWeights	double array of dimension nbAssets	optimal portfolio
info	integer array of dimension 1	diagnostic argument

2.5.3 WREallocSharpeRatio - Maximum Sharpe Ratio

Overview

This component computes the optimal portfolio with maximum Sharpe Ratio under various linear constraints.

Inputs

Variable	Type	Description
nbAssets	integer	number of asset(s) (¿1)
cov	double array of dimension nbAssets by nbAssets	covariance matrix
expectedReturns	double array of dimension nbAssets	mean return(s)
nbEqConst	integer	number of equality constraints (¿=0)
nbIneqConst	integer	number of inequality constraints (¿=0)
C	double array of dimension nbAssets by nbEqConst+nbIneqConst	matrix of constraints. If no constraints are given, C=0
b	double array of dimension nbEqConst+nbIneqConst	vector of constraints. If no constraints are given, b=0
minWeights	double array of dimension nbAssets	lower bounds
maxWeights	double array of dimension nbAssets	upper bounds
riskFreeRate	double	the risk-free rate (with the same frequency as expected returns)

Outputs

Variable	Type	Description
optimalWeights	double array of dimension nbAssets	optimal portfolio
info	integer array of dimension 1	diagnostic argument

2.5.4 WREanalysisExanteModifiedVaR - Ex-ante Modified Value-at-Risk

Overview

The modified Value-at-Risk (VaR) adjusts the traditional Gaussian VaR with the Skewness and Kurtosis of the distribution.

Inputs

Variable	Type	Description
nbValues	integer	number of values (¿=4)
nbAssets	integer	number of asset(s) (¿=1)
weights	double array of dimension nbAssets	portfolio's weight(s)
assetsReturns	double array of dimension nbValues by nbAssets	asset(s) returns
cov	double array of dimension nbAssets by nbAssets	covariance matrix
probabilityLevel	double	probability level (¿0 and ¡1) - if exanteModifiedValueAtRisk ¡ 0: probability that the loss will be lower to exanteModifiedValueAtRisk - if exanteModifiedValueAtRisk ¿ 0: probability that the gain will be upper to exanteModifiedValueAtRisk

Outputs

Variable	Type	Description
exanteModifiedValueAtRisk	double array of dimension 1	portfolio's ex-ante modified Value-at-Risk
info	integer array of dimension 1	diagnostic argument

2.5.5 WREanalysisExanteNormalVaR - Ex-ante Value-at-Risk (Gaussian)

Overview

This component computes the Gaussian ex-ante Value-at-Risk (VaR) with the parametric Variance-Covariance method.

Inputs

Variable	Type	Description
nbAssets	integer	covariance matrix size (¿=1)
cov	double array of dimension nbAssets by nbAssets	covariance matrix
expectedReturns	double array of dimension nbAssets	mean return(s)
weights	double array of dimension nbAssets	portfolio's weight(s)
probabilityLevel	double	probability level (¿0 and ¡1) - if exanteNormalValueAtRisk ¡ 0: probability that the loss will be lower to exanteNormalValueAtRisk - if exanteNormalValueAtRisk ¿ 0: probability that the gain will be upper to exanteNormalValueAtRisk

Outputs

Variable	Type	Description
exanteNormalValueAtRisk	double array of dimension 1	portfolio's ex-ante Normal Value-at-Risk
info	integer array of dimension 1	diagnostic argument

2.5.6 WREanalysisExanteReturn - Ex-ante Mean Return

Overview

This function computes the ex-ante mean return (performance) of a given portfolio.

Inputs

Variable	Type	Description
nbAssets	integer	number of asset(s) (¿=1)
expectedReturns	double array of dimension nbAssets	mean return(s)
weights	double array of dimension nbAssets	weight(s)

Outputs

Variable	Type	Description
exanteReturn	double array of dimension 1	portfolio's mean return
info	integer array of dimension 1	diagnostic argument

2.5.7 WREanalysisExanteVolatility - Ex-ante Volatility

Overview

This component computes the ex-ante volatility (standard deviation) of a portfolio.

Inputs

Variable	Type	Description
nbAssets	integer	covariance matrix size (¿=1)
cov	double array of dimension nbAssets by nbAssets	covariance matrix
weights	double array of dimension nbAssets	weight(s)

Outputs

Variable	Type	Description
exanteVolatility	double array of dimension 1	portfolio's ex-ante volatility
info	integer array of dimension 1	diagnostic argument

2.5.8 WREanalysisExpostReturn - Ex-post Mean Return

Overview

This function computes the ex-post mean return (performance).

Inputs

Variable	Type	Description
nbValues	integer	number of points (¿=1)
portfolioReturns	double array of dimension nbValues	portfolio's return(s)

Outputs

Variable	Type	Description
expostReturn	double array of dimension 1	ex-post mean return
info	integer array of dimension 1	diagnostic argument

2.5.9 WREanalysisExpostVolatility - Ex-post Volatility

Overview

The volatility (or standard deviation) is a statistical measure of the historical returns. The ex-post volatility is usually computed using daily or monthly returns.

Inputs

Variable	Type	Description
nbValues	integer	number of points (¿1)
portfolioReturns	double array of dimension nbValues	portfolio's return(s)

Outputs

Variable	Type	Description
expostVolatility	double array of dimension 1	ex-post volatility
info	integer array of dimension 1	diagnostic argument

2.5.10 WREanalysisGaussianKernel - Density Estimation by Gaussian Kernel Method

Overview

Inputs

Variable	Type	Description
n	integer	number of return(s) (¿=2)
m	integer	number of grid point(s) (¿=1)
x	double array of dimension n	return(s)
y	double array of dimension m	grid (increasing order)

Outputs

Variable	Type	Description
z	double array of dimension m	Gaussian kernel estimation
info	integer array of dimension 1	diagnostic argument

2.5.11 WREmodelingCorr - Correlation Matrix

Overview

Inputs

Variable	Type	Description
nbValues	integer	number of points (¿1)
nbAssets	integer	number of variables (¿=1)
assetsReturns	double array of dimension nbValues by nbAssets	returns matrix

Outputs

Variable	Type	Description
corr	double array of dimension nbAssets by nbAssets	correlation matrix
info	integer array of dimension 1	diagnostic argument

2.5.12 WREmodelingCov - Covariance Matrix

Overview

Inputs

Variable	Type	Description
nbValues	integer	number of points (¿1)
nbAssets	integer	number of variables (¿=1)
assetsReturns	double array of dimension nbValues by nbAssets	returns matrix

Outputs

Variable	Type	Description
cov	double array of dimension nbAssets by nbAssets	covariance matrix
info	integer array of dimension 1	diagnostic argument

2.5.13 WREmodelingLogReturns - Logarithmic return(s)

Overview

Inputs

Variable	Type	Description
nbValues	integer	number of historical points (¿=2)
nbAssets	integer	number of variables (¿=1)
assetsValues	double array of dimension nbValues by nbAssets	matrix of values (¿0)
horizon	integer	investment horizon (0 ¿ horizon ¡ nbValues)

Outputs

Variable	Type	Description
assetsReturns	double array of dimension nbValues-horizon by nbAssets	matrix of returns
info	integer array of dimension 1	diagnostic argument

2.5.14 WREmodelingReturns - Simple Net return(s) (Arithmetic)

Overview

Inputs

Variable	Type	Description
nbValues	integer	number of historical points (¿=2)
nbAssets	integer	number of variables (¿=1)
assetsValues	double array of dimension nbValues by nbAssets	matrix of values (¿0)
horizon	integer	investment horizon (0 ¿ horizon ¡ nbValues)

Outputs

Variable	Type	Description
assetsReturns	double array of dimension nbValues-horizon by nbAssets	matrix of returns
info	integer array of dimension 1	diagnostic argument

2.5.15 WREmodelingSDLS - Semi-Definite Least Square (SDLS) optimization

Overview

Inputs

Variable	Type	Description
p	integer	size of the matrix (¿=1)
Q	double array of dimension p by p	input matrix
nbEqConst	integer	number of equality constraints (¿=0)
Ceq	double array of dimension p by nbEqConst*(p)	symmetric matrices of equality constraints
bEq	double array of dimension nbEqConst	vector of equality constraints
nbIneqConst	integer	number of inequality constraints (¿=0)
Cineq	double array of dimension p by nbIneqConst*p	symmetric matrices of inequality constraints
bLowerIneq	double array of dimension nbIneqConst	vector of lower constraints
bUpperIneq	double array of dimension nbIneqConst	vector of upper constraints
constPrecision	double	constraints precision (¿0)
minEigenValue	double	minimum level of eigenvalue desired in the output matrix (¿=0)

Outputs

Variable	Type	Description
X	double array of dimension p by p	corrected matrix
info	integer array of dimension 1	diagnostic argument

2.5.16 WREmodelingSDLScorr - Semi-Definite Least Square (SDLS) optimization for a correlation matrix

Overview

Inputs

Variable	Type	Description
p	integer	size of the matrix (¿=1)
Q	double array of dimension p by p	input matrix
constPrecision	double	constraints precision (¿0)
minEigenValue	double	minimum level of eigenvalue desired in the output matrix (¿=0 and ¡1)

Outputs

Variable	Type	Description
X	double array of dimension p by p	corrected correlation matrix
info	integer array of dimension 1	diagnostic argument

2.5.17 WREsimulGeometricBrownianX - Simulation of a Multidimentional Geometric Brownian Motion

Overview

Inputs

Variable	Type	Description
p	integer	number of assets (¿1)
T	integer	time (days, months, years) (¿0)
N	integer	number of sub-division(s) (¿1)
S	double array of dimension p	initial values
mu	double array of dimension p	process drifts
cov	double array of dimension p by p	covariance matrix

Outputs

Variable	Type	Description
y	double array of dimension N by p	generated process
info	integer array of dimension 1	diagnostic argument

3 Troubleshooting

This chapter provides the user with instructions on how to access and interpret the error handling contained within the Wall Risk Engine® library. If these suggestions do not help evaluate specific difficulties, we suggest that the user review the "bug report" section (how to report a problem) and contact the Raise Partner support at support@raisepartner.com.

3.1 Common errors

We list in this section the most common programming errors when calling Wall Risk Engine® functions. These errors may cause the function to report a failure or they may lead to wrong results:

- wrong number of arguments,
- arguments in the wrong order,
- an argument of the wrong type (especially double and integer arguments of the wrong precision),
- wrong dimension for an array argument,
- bad value for an argument,
- insufficient space.

3.2 Bug report

All requests should be sent to the consultant in charge of your project at Raise Partner. The first questions we will ask are the following:

- the client's name and/or reference code,
- the type of machine on which the function was run (e.g. Intel based),
- Operating system (e.g. Windows XP Professional),
- the program language used, e.g. Visual Basic 6.0,

- the name of the Wall Risk Engine® function, e.g. NORMallocIT,
- the value of the function return code (res),
- the value of the diagnostic argument (info),
- if appropriate, a copy of the input file(s) and output file(s).

3.3 Error codes

Code	Description
0	Successful Termination
1i	Error in the ith argument size - computation performed. Ex. code = 102, the second argument has bad size.
2	Memory allocation problem - no computation performed.
3i	Illegal value(s) of the ith argument - no computation performed. Ex. code = 301, the first argument has bad size.
4*	License has expired or bad license - no computation performed. 41 or 42: license date expired. 43: MAC address not found. 44: MAC address access denied.
5	Numerical problem in computation - no computation performed. For further information see the diagnostic argument *info* <0.
6	Warning - computation performed (but not optimal). For further information see the diagnostic argument *info* >0.

3.3.1 Exit code *res*

Every function of Wall Risk Engine® returns an integer exit code that should be tested by the calling program. The exit code is to be interpretable as follows:

3.3.2 Diagnostic argument *info*

All routines have a diagnostic argument info that indicates the success or failure of the numerical computation (in case the error code is 5 or 6) for each function, as follows:

- info = 0 : successful termination, no problem occurred.
- info = * : computation problem defined for each routine - cf. table hereafter for the diagnostic argument interpretation.
    - If info > 0: Warning - computation performed (but not optimal)
    - If info < 0: Problem in computation - no computation performed

3.3.3 Arithmetic errors

Error Code	Warning	Description
-1		Division by zero - Divisor is zero and dividend is a finite nonzero number (ex. 10.8/0.0).
-2		Not enough data/value(s) for computation.
-3		Underflow - Operation produces a result that is too small to be represented as a normal number.
-4		Overflow - Operation produces a result that exceeds the range of exponent.
-5		Invalid operand - Operation with mathematically invalid operand.
6	x	Inexact - Operation produces a result that cannot be represented with finite precision.

3.3.4 Asset allocation errors

Error Code	Warning	Description
-100		The constraints $C_{inf} <= w <= C_{sup}$ or $C <= b$ are incompatibles.
-101		The target return is too large (ex. Superior to the maximum of the underlying specified returns).
-102		The probability confidence level a is not in [0%, 100%].
-103		A quote/price value is too small (or zero).
-104		The variance/volatility is too small (or zero).
-105		The Tracking Error is too small (or zero).
106	x	The number of maximum iterations is reached without finding an optimal portfolio.
107	x	The TE/VaR constraint is too small The portfolio with minimum TE/VaR is computed.
-108		Covariance matrix must be definite positive (use matrix correction function).
-109		The input matrix (Risk Budgeting objective function) is not definite positive.
110	x	A risk budget constraint is too small (the dual solution of this constraint is equal to 1.E+15).
111	x	A risk budget constraint is to high, maximum number of iterations reached without saturation of this risk budget constraint (the dual solution of this constraint is equal to 0).
-112		A risk budget Index tracking constraint is negative, should be positive.
-113		Max. Sharpe ratio unattainable - No problem solution
-114		The risk-free rate is too large. No problem solution.

3.3.5 Optimization errors

Error Code	Warning	Description
1001	x	There are no admissible optimal points with these specified constraints.
-1002		No inferior bound supplied.
-1003		Degenerate point with infinite cycle.
-1004		Problem of bound, too large gap between two successive iterations.
-1005		Incorrect input(s) data.
-1006		Incompatibility of equality constraints.
-1101		Error in the simulation/objective function (non-linear optimization).
-1102		An input parameter is badly initialized (non-linear optimization).
-1103		Quasi-Newton matrix is no definite positive (non-linear optimization).
1104	x	Maximum number of iterations reached without finding an admissible point (non-linear optimization).
1105	x	Maximum number of "simulator" call reached without finding an admissible point.
1106	x	Maximum precision reached without finding an admissible point (non-linear optimization).
-1107		Error in the Hessian matrix factorization (non-linear optimization).
-1108		The problem have no solution (info = 1106 with dxmin < 1.E-15)
-1109		At least one equality constraint unverified.
-1110		Problem with constraints, failed to pass checks
-1201		Problem in the dichotomy function.
1202	x	Maximum number of iterations reached without finding an admissible point (dichotomy).
5001	x	Absolute tolerance reached (SOCP).
5002	x	Relative tolerance reached (SOCP).
5003	x	Target value achieved (SOCP).
5004	x	Maximum number of iterations reached (SOCP).
5006	x	If feasible original problem is unbounded (SOCP).
-5001		Incorrect input(s) data (SOCP).
-5002		Error in the initialization of the size of the constraints (must be greater than 0) (SOCP).
-5003		Dual constraints no verified (SOCP).
-5004		Primal constraints not respected, return (SOCP).
-5005		Primal constraints no verified, unfeasible problem (SOCP).
-5006		The matrix A is not full ranked (SOCP).
-5007		Problem with lower/upper bounds (SOCP).
-5008		Target value unachievable (SOCP).
-5009		The constraint matrix not a symmetric matrix (SOCP).
-5010		The constraint matrix not a strictly positive matrix (SOCP).

-5011		None inverse matrix (SOCP).
-5014		The lambda parameter is negative (SOCP).
-5016		The linear primal constraints are not respected (SOCP).
-5019		Problem with the computation of pseudo-inverse. Constraints badly initialized (SOCP).
-5021		Problem (kappa > 0) amplitude of the ellipsoid is strictly negative (SOCP).

3.3.6 Matrix calibration errors

Error Code	Warning	Description
-1300		The input parameter α is too large, the constraint $\alpha \leq n$*Trace(C) is not verified (SDLS with trace constraint).
1301		The % of explain variance is too low.
1302		The % of explain variance is too large.
1310		No correction. The minimum eigenvalue is equal the maximum eigenvalue.
-2001		The input parameter α is not in [0, 1], X $\geq \alpha$*Id (SDLS for correlation matrix).
-2003		Specified variance not in [0, 1] (Kato correction).
-2004		Trace(X) is too small or zero (Kato correction).

3.3.7 Risk factor model errors

Error Code	Warning	Description
-4001		Bad index value (not equal to 11, 12, 13, 21, 22 or 23).
-4002		Number of point(s) used in iteration is too small or zero.
-4003		Incoherent number of tracking factor(s).
-4006		Numbers of iteration is too small or zero (Tracking Factor).
-4008		L2 norm is too small or zero (Tracking Factor).
4009	x	The covariance matrix Kato-structure is completely changing.

3.3.8 Simulation errors

Error Code	Warning	Description
-3001		The investment horizon is too small (or zero).
-3002		Initial price(s) is too small (or zero).
-3003		Number of simulation(s) is too small (or zero).
-3101		Bad process order (ARCH, GARCH).
-3102		Condition of process stationary not yields (ARCH, GARCH).

3.3.9 Pricing errors

Error Code	Warning	Description
-3201		First moment is negative
-3202		Second moment is negative
-3203		Underlying price is negative or null
-3204		Strike is negative or null
-3205		Time to maturity is negative
-3206		Barrier is negative or null
-3207		Initial value is negative (Brownian bridge)
-3208		Final value is negative or null (Brownian bridge)
-3209		Nominal is negative or null
-3210		Capital protection is negative or null
-3211		Yield to maturity is lower than -1 (formula: rym = ((N-C0)/(CP*N))**(T) -1)
-3212		Barrier is lower than financial level (ex mini future)
-3213		Price of the underlying is lower or equal to Barrier (ex mini future)
-3214		Barrier is greater than or equal to strike
-3215		Leverage is negative or null
-3216		Coupon is greater than 1
-3217		Number of observation dates has to be >= 1
-3218		Negative price got in the pricing process (ex. Exchangeable certificate, call at the emission)

3.3.10 Portfolio simulation errors

Error Code	Warning	Description
-3301		Threshold1-threshold2 is equal to zero in ConfidScore
-3302		Sum of underlying weights is null
-3303		Unrecognized type of product
3304	x	One or more than one product has experienced an error when pricing
-3305		Currency index is out of limit (must be <= nbcrncy and >= 1)
-3306		Time to maturity is negative or null
-3307		Bond last price St is negative or null
-3308		Nominal is negative or null (ex. Bonds)
-3309		One or more of bond historical prices are negative or null
-3310	x	Coefficient of diversification experienced a problem (possibly, correlations calculus)
-3311		All the instruments have experienced a problem when pricing, no calculation is done
3312	x	Error occurred while calculating VaR contributions
3313	x	Error occurred while calculating Performance contributions
-3314		Error occurred while calculating the VaR
-3315		Error occurred while calculating the confidence score
-3316		Error occurred while calculating the portfolio ex-post mean return
-3317		Error were occurred when adjusting bond returns (division by 0)
-3318		Product underlyings must be of the same currency (except straight certificate)
-3319		FX rate return of the product is less or equal than -1 (devision by 0)
-3320		Simulated FX is negative or null (devision by 0)
-3321		Product must have underlying currency index (see Ucrncies and Underlyings parameters)
-3322	x	Coefficient of diversification of the equi-weighted Ptf experienced a problem
-3323		Implicit volatility calculation problem
-3324		S and/or A are less then its threshold
-3325		NaN value detected in the outputs (maybe internal error or problem with data)
3326	x	
-3327		Market to theo problem

3.3.11 Data management errors

Error Code	Warning	Description
-9001		All data are missing for a date.
-9003		No enough observed data.

3.3.12 Data management errors

Error Code	Warning	Description
-301		Argument 1 has bad value (bad link to the source of data)

Copyright