PROCEEDINGS OF SCIENCE

Grid Based Full Portfolio Revaluation for VaR Computation

Gianluca Fusai *
Dipartimento SEMEQ,
Università del Piemonte Orientale
Via Perrone, 18 - 28100 Novara
E-mail: gianluca.fusai@eco.unipmn.it

Roberto Chinelli
Avanade Italy
Corso Venezia, 46 - Milano
E-mail: robertoch@avanade.com

Daniele De Martini, Giovanni Longo
Dipartimento SEMEQ,
Università del Piemonte Orientale
Via Perrone, 18 - 28100 Novara
E-mail: daniele.demartini@eco.unipmn.it
E-mail: giovanni.longo@eco.unipmn.it

Marina Marena
Dipartimento di Statistica e Matematica Applicata Diego de Castro,
Università degli Studi di Torino
E-mail: marina.marena@econ.unito.it

Laura Mariano, Roberto Cappuccio, Raffaele Sgherri, Luca Regini, Werther Chierici, Ernesto Cocca
Avanade Italy
Corso Venezia, 46 - Milano

Full revaluation of a portfolio based on Monte Carlo simulation can be very burdensome from the computational point of view. In this paper we develop a full revaluation of a middle-size Italian Bank portfolio for Value at Risk (VaR) computation on Avanade Grid Architecture (AGA.NET) addressing performance and trustworthiness issues. Moreover, we compare the grid implementation with the delta-gamma approximation based on the Fast Fourier algorithm.

Grid Technology for Financial Modeling and Simulation
3-4 february 2006
Palermo, Italy

*Speaker.

PoS(GRID2006)011

1. Introduction

Institutions are frequently faced with problems of data processing that request a very high computing power that is not matched by any individual available computing system. Nevertheless, such processing power limit could often be reached and even exceeded coordinating several systems, even already available at their sites, through a Grid Computing Architecture.

A Grid computing Architecture is characterized by large-scale sharing and cooperation of dynamically distributed resources, such as CPU cycles, communication bandwidth, and data, to constitute a computational environment. A grid's dynamic environment consists of procedures that can be split into a high number of tasks which can be executed independently on different computing nodes.

The aim of the present paper is consider a standard financial problem such as the computation of the Value-at-Risk (VaR) of a portfolio using a full Monte Carlo . This problem is computationally very intensive, because it consists in the re-valuation of hundreds of assets under several different market scenarios.

Therefore, we will examine how the computations can be based on Avanade Grid Based Architecture (AGA.NET), an application architecture which is able to facilitate the implementation and execution of distributed processing. AGA.NET splits applications into a high number of tasks that can be executed independently on different computing nodes (see Figure 2).

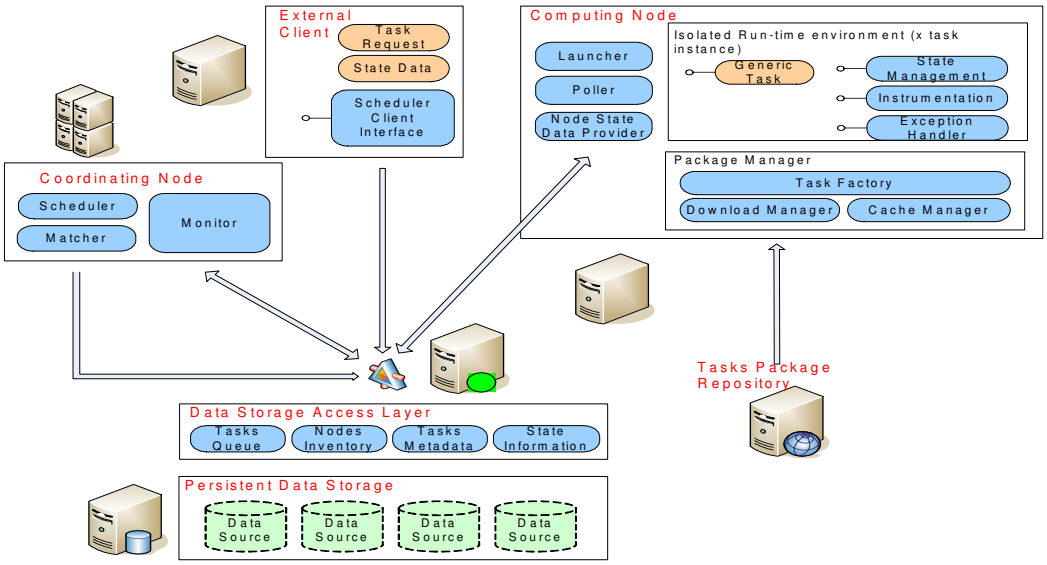

Figure 1: Avanade Grid Architecture

The intrinsically parallel aspect of Monte Carlo applications makes them an ideal fit for the grid-computing paradigm. In general such an application, divided in tasks (basic unit of work) can be submitted to the scheduler service and assigned to the best computing node chosen according to an heuristic routine called the matching algorithm. A central service (named monitor) will be constantly monitoring the execution of the tasks at the computing nodes by periodically checking

their status to verify that everything is going as expected. In particular, this service will check that processing time does not exceed a limit assigned by the scheduler for a given type of task, that every computing node is sending its own heart beat (polling) with the expected frequency, that the time since the last checkpoint in the task (checkpoints should be defined for complex long running task) does not exceed a specified limit, and finally that computing nodes take ownership of tasks that are assigned to them within a specified amount of time since the assignment by the scheduler. The definition of checkpoints within tasks and the persistence of tasks' state at every checkpoint enable to resume the tasks from a last good known checkpoint rather than from the beginning, thus saving time and computing power.

In Section 2 we formulate the problem and the two solutions that are commonly used in practice: Monte Carlo simulation and delta-gamma approximation with Fast Fourier Inversion (FFT). In Section 3 we present the grid computing technology. In Section 4 we describe the Monte Carlo Full Evaluation Application Architecture. In Section 5 we illustrate preliminary results and possible future researchs.

2. Value at Risk computation

In this section we will discuss the implementation of the Monte Carlo simulation with the aim of measuring the risk of a complex portfolio over a time horizon dt. The risk measures we are interested in are the loss probability given a thresold x and the portfolio's Value at Risk (VaR) at a given confidence level α, $VaR_\alpha(t,t+dt)$. The loss probability refers to the probability that the portfolio loss will be greater than x. The VaR of a portfolio is defined as the loss which, with a certain probability α, will not be exceeded. If the probability distribution $F_L(x)$ of the loss is known, then the VaR is its $1-\alpha$ quantile

$$1-\alpha = F_L\left(VaR_\alpha(t,t+dt)\right),$$

and then, assuming that F_L is strictly increasing

$$VaR_\alpha(t,t+dt) = F_L^{-1}(1-\alpha) \tag{2.1}$$

The time horizon dt and the confidence level α are the two major parameters that should be chosen in an appropriate way in relation to the overall goal of risk measurement. The time horizon can differ from a few hours for an active trading desk to a year for a pension fund. When the primary goal is to satisfy external regulatory requirements, such as bank capital requirements, the quantile is typically very small (for example, 0.1% or 1% of worst outcomes). However for an internal risk management model used by a company to control the risk exposure the typical number is around 5%. The two measures can be used by financial institutions to assess their risks or by a regulatory committee to set margin requirements. In either case, VaR is used to ensure that financial institutions can still be in business after a large market crash.

VaR provides a common measure of risk across different positions and risk factors. It can be applied to any type of portfolio and enables us to compare the risks across different portfolios, such

as fixed-income and equity. Traditional methods are more limited: duration and convexity measures apply only to fixed-income positions, sensitivities such as the Greeks apply only to derivatives positions, portfolio measures apply to equity and similar positions such as commodity, and so forth. Additional important aspects that makes the VAR an important tool in risk management are: it allows one to aggregate the risks of positions, taking account of the ways in which risk factors correlate with each other, it takes full account of all driving risk factors, whilst for example Greek measures look at risk factors only one at a time. It focuses assessment on a complete portfolio and not just the individual positions in it[1]. A general introduction to VaR can be found in Pearson (1996), Jorion (1997) and Dowd (1998).

The estimation of VaR requires the knowledge of the loss distribution. This can be simulated using Monte Carlo scenarios. However, a financial institution may have thousands of outstanding loans and therefore the simulation of a large number of risk factors and revaluation of the portfolio for each simulation, computation of the loss distribution and detection of the desired percentile is usually reputed expensive. Avanade Grid Architecture makes possible to perform a Full Monte Carlo portfolio revaluation: since the calculations that occur in each simulation are independent, the Monte Carlo method can exploit parallel computation very effectively. However, the procedure will still be computationally burdensome for some complex derivatives that cannot be valued analytically and require intensive numerical computations[2].

The accuracy of the resulting VaR estimate is primarily a function of the number of Monte-Carlo trials performed, although elegant methods to increase accuracy are available, such as control variates and importance sampling; for a review see Glasserman (2000).

The portfolio under examination represents a sample of real trading positions of a middle-sized Italian bank and contains assets with several exotic features. Our sample is composed of 200 assets, depending on 40 risk factors, 18 for interest rate sensitive products and 22 stock prices. In particular, we can distinguish:

a. 100 corporate bonds with different optional characteristics (caps, floors, swaptions, constant maturity swaps, callable, etc.) and different maturities, ranging from a few months to up to 30 years.

b. 100 exotic products written on 22 different assets belonging to the EuroStox index.

All the assets in the portfolio are priced in the Black-Scholes framework. The Monte Carlo procedure for full revaluation consists of three steps: Setup, Simulation and Analysis of the results. The assumed time frame is 10 days: after simulating the risk factors we reprice the entire portfolio. The three steps are descripted here below.

Setup

1. Identification of the risk factors. The relevant risk factor have been identified in spot rates with different maturities (3m, 6m, 9m, 1yr., 2yrs., 3yrs., 4yrs., 5yrs., 6yrs., 7yrs., 8yrs., 10yrs., 12yrs., 15 yrs., 20 yrs, 25 yrs and 30 yrs) and in 22 different assets belonging to the EuroStox index.

[1] However, the VaR measure does not satisfy some natural property, such as subadditivity. This means that the risk of the total position is less than or equal to the sum of the risk of individual portfolios. Other measures have been proposed in the literature to face such problems, such as expected shortfall, i.e. the average loss in states beyond the VaR level. For a discussion see Artzner et al. (1999). See also Meucci (2005) for a more in-depth analysis.

[2] This is the case for example of American options on a basket of assets.

2. Estimation of the 40x40 covariance matrix of the absolute variations of the risk factors. In particular, we have at first estimated separately the sub-matrices with reference to the two set of risk factors. The covariance matrix in the stock market has been estimated shrinking the sample covariance matrix towards the identity matrix, as suggested in Ledoit and Wolf (2003) and in Meucci (2005)[3]. Then, we have estimated the covariances relative to the two set of risk factors assuming a constant correlation. This correlation has been estimated with reference to the EuroStox index and to an index constructed as arithmetic average of spot rates with different maturities.

3. Given the short horizon of VaR calculations, it is a standard assumption that the risk factors jointly evolve according to a multivariate Brownian process

$$d\mathbf{f}(t) = \mu\Delta t + \mathbf{S}d\mathbf{W}(t), \tag{2.2}$$

where $d\mathbf{f}(t)$ is the 40x40 vector representing the stochastic shocks to the different risk-factors, μ is the expected instantaneous variation of the risk-factors, $\mathbf{S}$ is the covariance matrix estimated according to the procedure described above, and $d\mathbf{W}(t)$ is the vector containing the independent Brownian motion increments. An additional and standard simplification, given the usual short horizon, consists in setting the drift vector μ equal to $\mathbf{0}$.

Simulation

The steps involved in the simulation are the following:

1. Computation of the Cholesky factorization of the covariance matrix $\mathbf{S}$, i.e. find a lower triangular matrix $\mathbf{A}$ such that $\mathbf{A}^T\mathbf{A} = \mathbf{S}$;

2. Generation of n independent Gaussian random variables, ε.

3. Generation of correlated random numbers setting $\eta = \mathbf{A}\varepsilon$

4. Update the values of the risk factors using

$$\mathbf{f}(t+\Delta t) = \mathbf{f}(t) + \eta\sqrt{\Delta t}$$

5. Revaluate the entire portfolio and compute the portfolio loss over the time frame $(t, t+\Delta t)$

$$L = \Pi(t,\mathbf{f}) - \Pi(t+\Delta t, \mathbf{f}+d\mathbf{f}).$$

where $\Pi(t,\mathbf{f})$ is the portfolio value at time t. In the revaluation step, we also include a calibration of the simulated term structure of spot rates using the Nelson-Siegel parametric functional form.

6. Repeat steps 2-5 N times and denote by L^i, $i = 1,...,N$, the portfolio loss generated in the i-th simulation.

Analysis

Given the results of the simulations, we estimate the portfolio's Value at Risk at a given confidence level α, $VaR_\alpha(t, t+dt)$.

1. The portfolio Value at Risk can be computed ordering in increasing order the realised loss values L^i, so that we compute the order statistics of the samples

$$L^{(1)} \leq L^{(2)} \leq ... \leq L^{(N)}$$

[3]This procedure consists in taking a weighted average of the sample covariance $\hat{\mathbf{S}}$ and a preassigned target matrix. As target matrix we have considered the matrix $\bar{\lambda}\mathbf{I}_n$, where $\mathbf{I}_n$ is the identity matrix of order n, $n = 40$, and $\bar{\lambda}$ is given by $tr\left(\hat{\mathbf{S}}\right)/n$, and $tr\left(\hat{\mathbf{S}}\right)$ represents the trace of the matrix $\hat{\mathbf{S}}$. The optimal shrinkage weight is given in Meucci (2005, pag. 208).

with $L^{(1)}$ being the sample minimum and $L^{(N)}$ the sample maximum. The VaR estimate is

$$\hat{V}aR_\alpha(t,t+dt) = L^{([(1-\alpha)N])},$$

where $[x]$ is the integer part of x.

2. We are also interested in quantifying the accuracy of our VaR estimate. We can use a asymptotic result[4] from the statistical literature (e.g., Cox and Hinkley, 1974, Appendix 2). Let us denote by $\zeta_{1-\alpha}$ the true $1-\alpha$ quantile of the loss distribution. Then $\hat{V}aR_\alpha$ is asymptotically normal with mean $\zeta_{1-\alpha}$ and variance $\alpha(1-\alpha)/\left(N[f(\zeta_{1-\alpha})]^2\right)$, that is

$$\hat{V}aR_\alpha \sim \mathcal{N}\left(\zeta_{1-\alpha}, \frac{\alpha(1-\alpha)}{Nf^2(\zeta_{1-\alpha})}\right),$$

where $f(x)$ represents the loss density. This result requires the knowledge of the density function at the quantile $\zeta_{1-\alpha}$, i.e. $f(\zeta_{1-\alpha})$. To this aim we have applied a two step procedure. At first, we estimate the density f with a locally adaptive smooth estimator, assigning a probability mass of $1/(N+1)$ to each interval $(L^{(i)}, L^{(i+1)})$, see Collings and Hamilton (1988) and De Martini (2000). Then we denote by $\hat{f}_{L,N}$ and $\hat{F}_{L,N}$ the empirical density and distribution functions, and we let $\hat{\xi}_{1-\alpha} = \hat{F}^{-1}_{L,N}(1-\alpha)$ to be the percentile estimated through the smoothing technique and we obtain for a given q:

$$\int_{\hat{\xi}_{1-\alpha-q}}^{\hat{\xi}_{1-\alpha+q}} \hat{f}(x)\,dx = 2q\frac{N}{N+1}.$$

Then, we define a further smoothed density $\tilde{f}$ by setting:

$$\int_{\hat{\xi}_{1-\alpha-q}}^{\hat{\xi}_{1-\alpha+q}} \hat{f}(x)\,dx = \int_{\hat{\xi}_{1-\alpha-q}}^{\hat{\xi}_{1-\alpha+q}} \tilde{f}\left(\hat{\xi}_{1-\alpha}\right)dx = \tilde{f}\left(\hat{\xi}_{1-\alpha}\right)(\hat{\xi}_{1-\alpha+q} - \hat{\xi}_{1-\alpha-q}),$$

and we finally have

$$\tilde{f}\left(\hat{\xi}_{1-\alpha}\right) = \frac{N}{N+1}\frac{2q}{(\hat{\xi}_{1-\alpha+q} - \hat{\xi}_{1-\alpha-q})}.$$

We apply this formula setting $q = 0.2\,(1-\alpha)$. However, the normal approximation may not be sufficiently accurate for estimating tail quantiles or probabilities if the number of simulation is not large or if the loss distribution is highly skewed. Moreover, for values of α close to 1, a relatively large number of simulations is required to attain an acceptably accurate estimate of the desired quantile. For this reason, we consider a different procedure that computes a confidence interval for ζ_α using a Bootstrap Percentile Method, see for example Efron and Tibshirani (1993). This consists in drawing with replacement a sample of size N from the original sample, and in computing the estimate of the percentile of interest, say $\hat{\xi}^*_{1-\alpha}$, on the basis of this re-sample. Then, the confidence interval of level $1-\gamma$ for $\zeta_{1-\alpha}$, is given by the percentiles $\gamma/2$ and $1-\gamma/2$ of the distribution of $\hat{\xi}^*_{1-\alpha}$. In practice, by applying a Monte Carlo technique, we repeat this sampling m times (in practice $m = 5000$), correspondingly obtaining the m estimates $\hat{\xi}^{*b}_\alpha$, $b = 1,...,m$. Finally, the confidence interval of level $1-\gamma$ for $\zeta_{1-\alpha}$ is given by the percentiles $\gamma/2$ and $1-\gamma/2$ of the empirical distribution of $\hat{\xi}^*_{1-\alpha}$.

[4]For a general review of asymptotic approaches to quantile estimation see Ma and Robinson (1998).

2.1 Delta-Gamma approximation

We compare Monte Carlo simulation estimates with the results coming from the delta-gamma approximation. This second alternative is based on approximating the changes of portfolio values using a second order Taylor expansion

$$-L(t,t+\Delta t) = \Pi(t+\Delta t, \mathbf{f}+\Delta\mathbf{f}) - \Pi(t,\mathbf{f}) \approx \Theta\Delta t + \delta^T \Delta\mathbf{f} + \frac{1}{2}\Delta\mathbf{f}^T \Gamma \Delta\mathbf{f} \equiv Q$$

,

where $\Theta = \partial V/\partial t$ is the first order sensitivity of the portfolio value with respect to time (theta), Δ is the vector with components $\delta_i = \partial V/\partial f_i$ and Γ is the square matrix with components $\Gamma_{ij} = \partial^2 V/(\partial f_i \partial f_j)$. In other words Δ and Γ collect the first and second order sensitivities of the portfolio value with respect to changes in the underling risk factors, i.e. the vector of portfolio deltas and the matrix of gammas[5]. The characteristic function of the quadratic approximation Q, given the assumption of normality of risk factors, can be computed in closed form, see for example Mathai and Provost (1992) or Glasserman (2000). Then an application of the FFT algorithm[6] yields the entire distribution of the portfolio loss, so that we can compute the loss probability and the VaR. This approximation is expected to perform well for portfolios with stable "Greeks" delta and gamma. This is not the case, with short maturity ATM options or options with discontinuous payoffs like digitals and barriers. Indeed, for these options the Greeks can change abruptly near the discontinuity points. In our implementation the sensitivities were computed numerically. Important contributions on the delta-gamma approximations include Cardenas et al. (1997) and Rouvinez (1997).

3. Grid Computing Technology

The introduction of Grid Technology has enabled the approach to numerical intensive computing procedures at a reasonable cost. At the same time, the acceptance of collaborative computing and Web based technologies has opened the door to a wide number of industry specific applications and servers, many of which running on Intel based low cost machines. As a consequence it is quite common to see large Institutions with hundreds or even thousands Intel based servers: it means availability of large, but often heterogeneous and fragmented, computing power.

The ability to fully leverage the available computing power is relevant for a Grid technology in order of resulting performances. Nevertheless performances can also be affected by the application itself: efficiency of the implementation or quality of code may impact performances adding overhead on a single batch of work to be distributed across multiple machines.

[5]The computation of the Greeks can be done analytically for many contracts. However, if our portfolio include path-dependent exotic options, having American or Asian features, the computation of the sensitivities can become quite expensive and require Monte Carlo simulation. For a discussion on the numerical computation of the Greeks see Glasserman (2000).

[6]The FFT algorithm has been implemented using the fractional Fast Fourier algorithm, with 2^{16} points. By this approach both the grid spacing of the density function and the sampling frequency of the characteristic function can be chosen independently. For details see Chourdakis (2004).

Distributing pieces of work across systems on a complex scenario is an orchestrated operation where achieving expected results at the appropriate time is fundamental. In an orchestrated environment a slower computing component could slow the end operation of an entire batch of work. It becomes therefore critical the ability to perform tasks in the expected time by fully exploiting the underlying Grid infrastructure, both at hardware and software level.

From the architecture point of view, it should be distinguished between central services functionalities and computing node functionalities. Central services functionalities can be set from the central administration console and have direct impact on the global behavior of the grid: matching algorithm, ranking algorithm, machine group virtual partitioning, task tuning, resuming from last known good checkpoint.

Computing node functionalities, instead, must be set on a per node basis and allow configuration of local policies on nodes: node responsiveness, node tuning, node virtualization, node manageability.

Through the matching algorithm the right task is being run at the right computing node: when one specific task is required from within the admin console a specific hardware platform, proper software requirements can be specified to find only those nodes capable to run it.

The maximum number of tasks running at a node and the maximum task load for a specific node should be key parameters in every grid administration console. An evaluation of every single task weight must necessarily be accomplished to train the task load mechanism.

Data is gathered from computing nodes by sending their status to the central scheduler service on a regular basis: the scheduler then executes a query back on the node information central database filtering out only relevant information. Computing node services typically perform two main duties: gathering data on their status and sending them to central services, checking central services for newly assigned task. The frequency by which a computing node executes this service task is adjustable and is a key parameter to influence the responsiveness of the whole scheduling process.

Finally, mechanisms for sending control signals to out of control or too much demanding tasks should be always provided in non-dedicated production environment to guarantee the fully manageability of grid system in all operating conditions, and also CPU scheduling priority of tasks should also be set to prevent impact on normal working activities.

In the very common scenario where machines of different hardware classes coexist in the same physical Grid, the possibility of virtual partitioning into groups of similar performing computing nodes is a very efficient way to optimize the execution of tasks. Geographic distribution and computing power are typical criteria in a virtual sub-grid scenario.

In a large and heterogeneous grid environment it may happen that some of the nodes have a failure: reassigning the task from failing nodes is an intrinsic feature in every grid technology but the possibility of saving what has already been done can save a huge amount of CPU cycles. Saving processing state should be possible in the central database so that, in the case of one node failure, the newly assigned computing node can resume from the last known good checkpoint state successfully saved by the failing node.

It is strongly advisable that grid task programming techniques are equipped by a powerful set of APIs to manage the whole life-cycle of tasks.

A full portfolio revaluation method adds more levels of complexity that impact performances by orders of magnitude. Properly orchestrating the distribution of work requires an efficient distribution strategy and application architectural implementation.

4. MCM Full Revaluation Application Architecture

MCM Full Revaluation presented here is the .NET application based on the AGA framework. Although AGA is available for both Linux and Windows systems, the solution presented here is the SOA architecture implemented on the Microsoft platform. The conceptual design is shown in Figure (2).

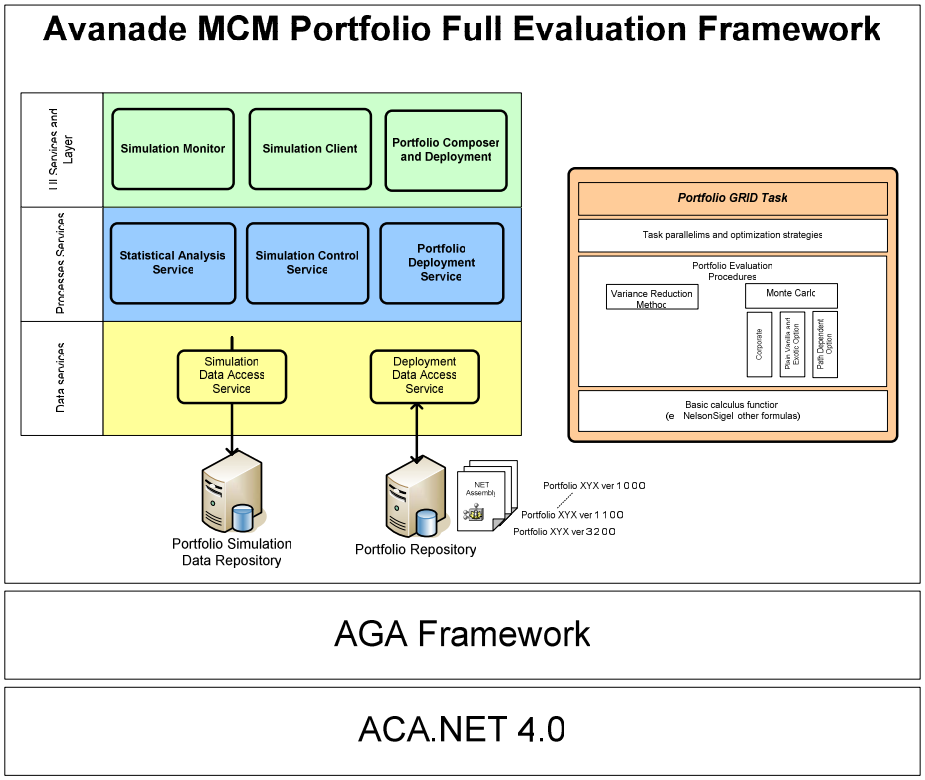

Figure 2: Avanade Grid Architecture

4.1 UI Service and presentation layer

Simulation monitor:

This web based application provides the functionalities to monitor the status of a simulation required by the user.

Simulation client (on the same web portal front):

This web application provides functionalities to start a simulation (revaluation) over a portfolio.

Portfolio Composer and Deployment:

This Windows Based client provides a rich UI to define a portfolio, including portfolio risk factors (rates, assets etc.).

4.2 Process Services Layer

Simulation Control Service:

This service provides simulation functionalities control: it runs a new simulation, it checks for completed simulation, it aborts a running simulation, it collects and retrieves data.

Statistical Analysis Service:

This service provides all statistical functionalities required to create the final simulation report based on simulation data: it includes the VaR calculus algorithm.

Portfolio Deployment Service:

This service provides functionality to compile and deploy a high performance configured portfolio into the system.

4.3 Data Services Layer

This layer supports all data access required for the system

4.4 Portfolio GRID Task

This component manages the Monte Carlo simulation based on the following modules:

Parallelism and workload strategies

This module provides functionalities to support multi CPU and multi machine environment in order to split the workload of a simulation on all resources available on the grid.

Portfolio evaluation procedures

This module provides the functionalities for the main computation of the portfolio value and includes the following blocks:

`Monte Carlo:`

Module for Monte Carlo implementation for running the simulation and saving the partial data.

`Pricing:`

Application modules for the evaluation of specific class of portfolio instruments, such as: Corporate (Fixed Income and Structured Note), Plain Vanilla and Exotic Options, Path Dependent Option (Asiatic).

`Variance Reduction Method`

This block provides functionalities to reduce the variance of the evaluation in order to reduce the number of simulations required.

Basic calculus function library

These modules include all basic financial evaluation function such as Nelson-Siegel, Black formulas or other pricing functions.

4.5 MCM Full Evaluation Use Cases

From the user's point of view, MCM Full Evaluation supports three main use-cases, Portfolio Configuration and Deployment, Portfolio Evaluation, Portfolio Simulation Data Analysis.

4.5.1 Portfolio Configuration and Deployment

This use-case includes all functionalities for the configuration and deployment process. Inputs for a portfolio configuration are financial instruments attributes, risk factors, risk measurement attributes. Portfolio data are saved as XML.

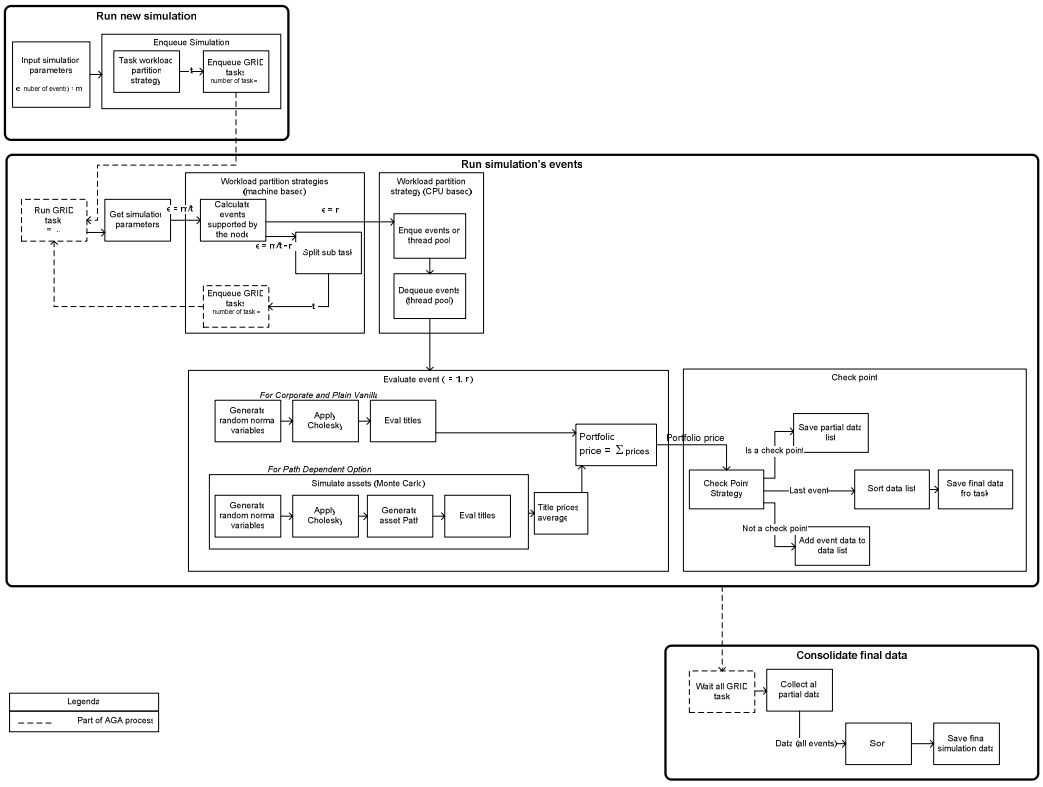

Figure 3: Processes overview.

The deployment process reads the portfolio XML configuration and generates the package binary code. The generator selects the evaluation formula for each portfolio instrument from the price model library and produces the module source code.

The module contains the portfolio data: risk factors, pre-processed input such as the Cholesky matrix from the variance-covariance matrix and the instruments evaluation functions.

Finally, the generator module compiles the source code and produces a .NET assembly (the package binary) ready to be deployed in the Portfolio Repository.

Portfolio evaluation versioning is supported, meaning that a single portfolio can be evaluated with different instruments/risk factors attributes and composition.

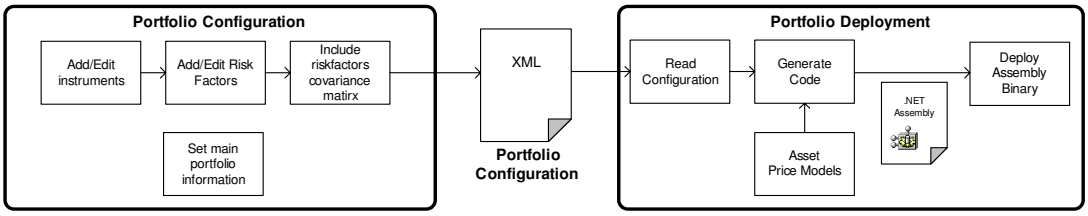

Figure 4: Portfolio configuration and deployment.

4.5.2 Portfolio Evaluation

This use-case is composed of three main processes, *run new simulation*, *run simulation's events*, *consolidate final data*.

Run new simulation Input simulation parameters are:

Calibration for Nelson-Siegel Model, *number of simulations* ($e_{\max}$), *workload strategy tuning parameters*, *min number of events per subtask* ($e_{\min}$), *max number of subtasks* (T_m), *checkpoint every events (Cp)*, *expected execution time* (T_e).

Outputs are: enqueued grid tasks where proper workload strategies should be applied.

Enqueue simulation strategy The strategy use the number of available grid nodes as initial task number:

The number of events assigned to a task is:

$$e_i = \frac{e_{\max}}{tasks}$$

where tasks = number of available nodes.

Run simulation events The following processes are executed on the grid computing node:

Workload strategies A workload strategy is applied for the task to calculate the number of events supported by the task on the machine while the task is running

$$\bar{e}_i = N_{cpu} \cdot \frac{Te}{\frac{\sum_{j=1}^{m} W_j}{CPU_{clock}}},$$

where W_j is the estimated number in mega cycles needed to calculate the j^{th} titles in the portfolio, CPU_{clock} is the CPU speed in MHz of the computing node and N_{cpu} is the number of CPU on the computing node.
The number of events per task is:

$$\hat{e}_i = \min\left(e_{i,} \max\left(e_{\min}, \bar{e}_i\right)\right).$$

If the number of events calculated by the strategy for the current task is less than the number of total event passed as input, than the remaining events are splitted in subtasks. The number of subtasks is calculated using the following split strategy:

$$Nsubtask = \max\left(1, \min\left(\frac{e_i - \bar{e}_i}{e_{\min}}, Tm\right)\right)$$

Each subtask receives: $e_k = (e_i - \bar{e}_i)/Nsubtask$ events (rounded up).

This strategy is shown in Figure (5):

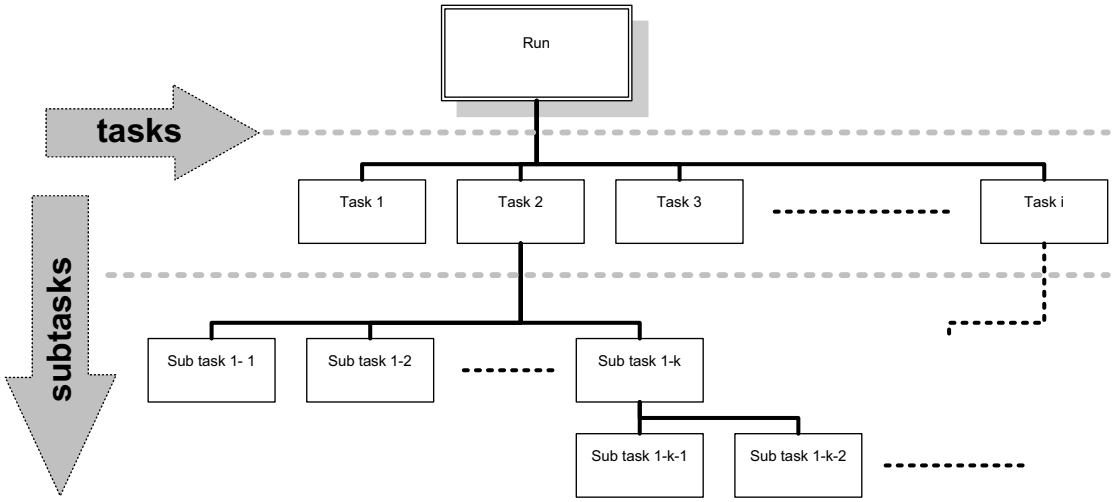

Figure 5: Workload strategy.

Execution Strategy The execution task uses a thread pool to parallelize the computation over multi CPU (multi core), and each single event is set on a queue spooled by a thread pool.

A thread on the threads pool dequeues the events and evaluates the portfolio.

Evaluate event (portfolio evaluation) To evaluate a portfolio for a specific event the task engages different pricing modules, such as Corporate (Fixed Income, Structured), Plain Vanilla, Exotic Options, Path Dependent Option, or eventually other modules.
For each module the same process is performed:
1. Generate random normal variables

The architecture is designed to choose the best random generator according to the environment where it runs (grid, process, thread).
2. Apply Cholesky factor to variables

The Cholesky factor is part of portfolio configuration and is deployed with it.
3. Apply price function

The applied pricing function depends on the calculus module and the instrument type and configuration. The function used is part of portfolio configuration and is deployed with it.

Checkpoint strategy During each task evaluation a checkpoint is performed by the AGA. The checkpoint feature is useful in case of failure: it allows the task to save partial data used to resume from the last know good execution point (last event successfully evaluated and saved). The check point strategy works as follows:
Save partial data: Each evaluated event result (value of portfolio calculated for the event) is saved in memory. If the number of evaluated events from the last checkpoint is equal to *Cp*, than the task saves all the calculated values for all completed events in theAGA data structure to support restart in case of failure and in the portfolio repository as partial data.
Save final data: All completed events are saved in the portfolio repository, marked as non-partial data, and sorted.

Consolidate final data The portfolio evaluation process ends when all tasks posted on grid, including subtasks, are finished. To consolidate data the process collects all data produced by the workload strategy application, sorts the entire collection of data and saves data for analysis.

4.5.3 Portfolio Simulation Data Analysis

The last use-case includes all functionalities to analyze the data produced by the simulation. The analysis is performed on the sorted portfolio prices produced by the process Consolidate Final Data. The analysis applies statistics functions to calculate VaR according to different confidence levels.

5. Results and Conclusions

Our Grid environment test is composed, on purpose, of a quite heterogeneous set of machines for memory size and clock speed, as shown in Table 1. The multiprocessor machine has been configured as a 4 single CPU virtual node.

Thousands of financial instruments may be priced in seconds while processing a single instrument with path-dependency features written on a basket of several assets may require days. The availability of the appropriate information becomes therefore relevant for a proper capacity planning. To achieve an acceptable response time, an implementation of a full portfolio revaluation

Name	CPU Number	Clock (Mhz)	RAM (Mb)
SEMEQ-Server	4	2666	1024
SEMEQ-22	1	2144	512
AV01100526	1	1995	1024
AV01100525	1	1995	1024
AV01100516	1	1995	1024
AV01100515	1	1994	1024
AV01100522	1	1994	1024
AV01100520	1	1994	1024
AV01100523	1	1994	1024
AV01100524	1	1994	1024
AV01100083	1	851	512
AV01100042	1	751	384
AV01100046	1	747	384
AV01100043	1	747	256

Table 1: Machines involved in the Grid simulation.

method may require few to hundreds of systems. Therefore it is truly important to achieve the right balance between available computing power and complexity of the algorithm required to evaluate a single financial instrument.

Response time is less relevant. Indeed, it is the application architecture that enables an optimal distribution of work based on the CPU power available. The overhead added by the grid infrastructure and by the portfolio full revaluation application (i.e. subtask dependencies) when distributing work is in the order of 1.2 second per task, decreasing with computer nodes.

Financial instruments included in our portfolio are grouped in two main categories, Corporate and Options. The first portfolio contains interest-rate derivatives, such as zero-coupon bond, corporate bond and constant maturity swap (CMS) with maturities ranging from 1 year up to 30 years. The second portfolio contains plain vanilla and exotic options on a single asset[7] or on several underlyings[8]. Table 2 reports the average clock cycles at the given frequency of 35.7 MHz of 200 different financial instruments .

The execution time is measured on a non dedicate Laptop (Dell D600 1 CPU Intel Centrino, 1,6 GHz, 1GBytes RAM) while other applications are running (e.g. Word, SQL Server, MS .NET development environment). The measures are done by using an high resolution built-it hardware timer with a frequency of 35.79545 MHz, every measure is repeated for $10\hat{4}$ times.

The sixth column in Table 2 gives the CPU usage of every financial instrument respect to the simplest one, a zcb with one year maturity. We can appreciate the different CPU usage for the different groups of instruments. In our current work we have listed instruments that are taking CPU up to 572 more times: if one second is needed for a certain instrument, up to 572 seconds are needed for a different type of instrument. The impact could be even bigger if we look at a full

[7]Cash or nothing, Asset or Nothing, Gap, Forward Start, Supershare, Simple Chooser options

[8]Exchange options, two asset cash-or-nothing option, two asset correlation option, option on the maximum or the minimum of two risky asset, complex chooser option.

Financial Instruments	Portfolio Percentage	Maturity (years)	Number of Coupons	Average Clock Cycles at Freq.35MHz	Times over base
ZCB	0.50%	1	1	3,625	1,00
Corporate (Floater & Fixed)	20.00%		2-34	5,031- 58,313	1,39-16,09
Corporate (Floater with options)	13.00%		3-94	28,683-1130,226	7,91-311,79
CMS	8.00%	$<=$ 15y	3-38	49,891-2076,248	13,76-572,76
Corporate CMS	7.00%	$>$15y	9-19	379,453- 1545,454	104,68-426,33
Plain Vanilla and Exotic	37.00%	0.90-15		4,601-20,942	1,27-5,78
Exotic on several underlyings	14.50%	0.50-2		25,934-253,514	7,15-69,93

Table 2: Execution time for different derivative contracts.

spectrum of instruments.

The complexity degree of the algorithm used for a specific instrument evaluation impacts on performance. If we consider a sample instrument and assume that a typical 60,000 instruments portfolio is composed by simple derivative contracts, the processing of a 60,000 elements portfolio may be performed in a short time on few machines. If we consider an Asian Option on a portfolio of 22 assets with daily monitoring, portfolio revaluation can take even hours. Therefore, the mix of financial instruments contained in the portfolio is a key factor influencing performances. This also shows us how it becomes relevant to optimize the computation algorithms in order to reduce response time and increase the quality of mathematical analysis. Grid computing offers a way to achieve great capabilities but not infinite capabilities.

Figure (6) represents the distribution of the portfolio losses. Standard statistical tests reject the normality assumption. Tables (3), (4), and (5) report the results of the Monte Carlo simulation and comparison with the VaR measure computed according to the delta-gamma approximation. The figures in the Tables refer to a time horizon of 10 days. Similar results have been obtained over a 1 month horizon, although the delta-gamma approximation deteriorates sligthly: higher probability of larger moves in the risk factors can make the second order approximation less reliable. We have computed the VaR measure for three different portfolios. The third portfolio is just the sum of the two components. We found after aggregating the two portfolios a diversification effect: the portfolio VaR is lower than the VaR of its components. Tables report also the VaR computed according to the delta-gamma approximation. We observe that it always falls outside the 95% confidence interval. The delta-gamma approximation should provide results even less accurate as we consider non normal shocks, such as the ones generated by fat-tails distributions, or as we include in our portfolio path-dependent assets. Indeed, for these derivatives the computation of accurate delta and gamma sensitivities contracts present both theoretical and practical challenges, (see the discussion in Glasserman (2000)).

Corporate Portfolio (Initial Value: 96.237, Time horizon: 10 days)			
Percentile	0.999	0.99	0.95
AN	(3.492, 3.493)	(2.600, 2.603)	(1.790,1.800)
Bootstrap	(3.473, 3.510)	(2.600, 2.603)	(1.790,1.800)
FFT	4.180	2.588	1.788

Table 3: AN: confidence interval using asymptotic normality. Bootstrap: confidence interval using the bootstrap methodology; FFT: VaR estimate using the analytical delta gamma approximation.

Exotic Portfolio (Initial Value: 266.735, Time horizon: 10 days)			
Percentile	0.999	0.99	0.95
AN	(14.280, 14.283)	(11.076, 11.088)	(8.229, 8.263)
Bootstrap	(14.208, 14.364)	(11.110, 11.051)	(8.227, 8.262)
FFT	13.770	10.701	7.977

Table 4: AN: confidence interval using asymptotic normality. Bootstrap: confidence interval using the bootstrap methodology; FFT: VaR estimate using the analytical delta gamma approximation.

Global Portfolio (Initial Value: 362.972, Time horizon: 10 days)			
Percentile	0.999	0.99	0.95
AN	(13.502, 13.505)	(10.510, 10.522)	(7.770,7.804)
Bootstrap	(13.446, 13.588)	(10.486, 10.543)	(7.770,7.803)
FFT	13.642	10.546	7.790

Table 5: AN: confidence interval using asymptotic normality. Bootstrap: confidence interval using the bootstrap methodology; FFT: VaR estimate using the analytical delta gamma approximation.

References

[1] Artzner, P., F. Delbaen, J. M. Eber, and D. Heath. (1999). Coherent measures of risk. *Mathematical Finance*. **9**(3): 203-228.

[2] Cardenas J., Fruchard E., Picron J.-F., Reyes C., Walters K., Yang W. (1999). Monte Carlo within a Day, Risk **12** February, 55-59.

[3] Boyle P., Broadie M. and P. Glasserman (1997). Monte Carlo methods for Security Pricing. *Journal of Economic Dynamics and Control* **21**, 1267-1321.

[4] Chourdakis K. (2004). Option Pricing Using the Fractional FFT. *Journal of Computational Finance*, **8**, n. 1-2, 1-18, winter.

[5] De Martini, D. (2000). Smoothed Bootstrap Consistency through the Convergence in Mallows Metric of Smooth Estimates. *Journal of Nonparametric Statistics*, **12**, pp. 819-835.

[6] Dowd, P. (1998). *Beyond Value at Risk: The New Science of Risk Management*. Wiley & Sons.

[7] Efron, B., Tibshirani, R.J., (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York.

[8] Foster I., Kesselman C. (2004). *The Grid 2: Blueprint for a New Computing Infrastructure*. The Morgan Kaufmann Series in Computer Architecture and Design.

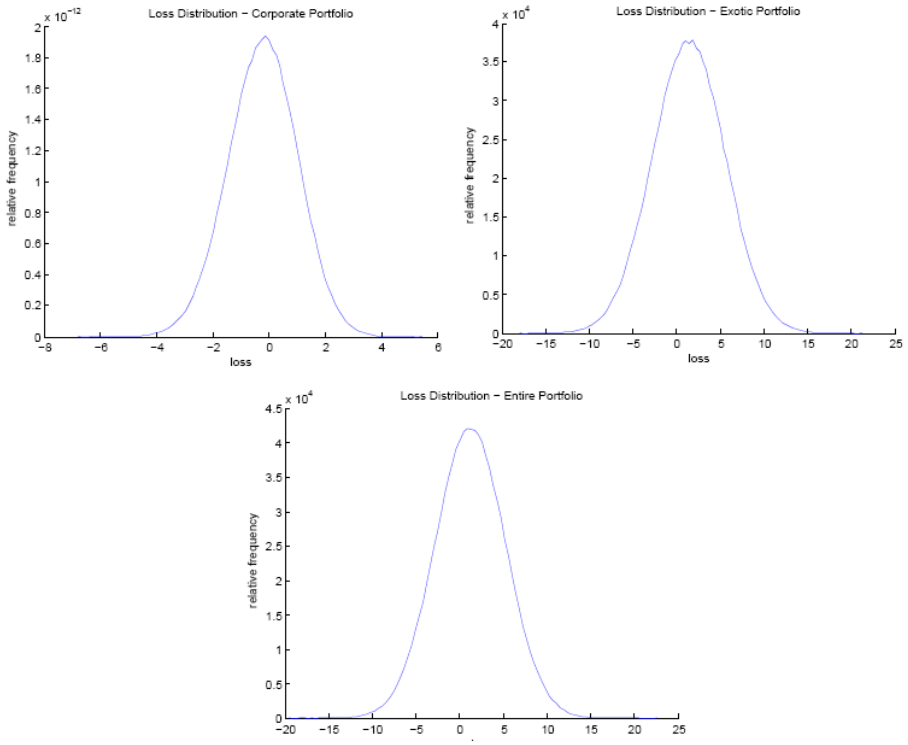

Figure 6: Loss Distribution for the three Portfolios.

[9] Glasserman P. (2000). *Monte Carlo Methods in Financial Engineering*. Springer.

[10] B.J., Hamilton, M.A., (1988). Estimating the Power of the Two Sample Wilcoxon Test for Location Shift. *Biometrics*, **44**, 847–860.

[11] Jorion, P. (1997). *Value at Risk, the New Benchmark for Controlling Market Risk*. McGraw-Hill.

[12] Ledoit, O. and Wolf, M. (2003). Improved Estimation of the Covariance Matrix of Stock Returns with an Application to Portfolio Selection. *Journal of Empirical Finance* **10**, 603-621.

[13] Ma, C., Robinson J. (1998). *Approximations to Distributions of Sample Quantiles, Order Statistics: Theory and Methods*. Handbook of Statistics **16**, 463-484, Elsevier Science B.V., Amsterdam

[14] Mathai A. M., Provost S. B. (1992). *Quadratic Forms in Random Variables: Theory and Applications*. New York, Marcel Dekker.

[15] Meucci A., (2005). *Risk and Asset Allocation*. Springer.

[16] Rouvinez C. (1997). Going Greek with VAR. Risk **10**, February, 57-65.