A financial engineering benchmark for performance analysis of grid middlewares

Viet Dung Doan, Abhijeet Gaikwad, Mireille Bossy, Françoise Baude, Frédéric Abergel

INRIA

A financial engineering benchmark for performance analysis of grid middlewares

Viet Dung Doan — Abhijeet Gaikwad — Mireille Bossy — Françoise Baude — Frédéric

Abergel

N° 365

Thèmes COM et NUM

Rapport technique

A financial engineering benchmark for performance analysis of grid middlewares

Viet Dung Doan[*] , Abhijeet Gaikwad[†], Mireille Bossy[*], Françoise Baude[*],
Frédéric Abergel[†]

Abstract: Pricing and hedging of higher order derivatives such as multidimensional (up
to 100 underlying assets) European and first generation exotic options represent mathe-
matically complex and computationally intensive problems. The power of Grid computing
promises to give the capability to handle such intense computations. It has been an at-
tractive cost-effective solution for high performance scientific computing for the last decade.
However, non-functional features of grid computing such as support for heterogeneity, fault
tolerance, deployabiliy, load balancing and efficient resource utilization have not been widely
applied in the computational finance domain. In this report we present our work that ex-
plores such issues and aims to demonstrate benefits from the application of grid techniques
to computational finance. Furthermore, with several grid middleware solutions available, it
is cumbersome to select an ideal candidate to develop financial applications that can cope
up with time critical computational demand for complex pricing requests. Another contri-
bution of our report is to present a financial benchmark suite to evaluate and quantify the
effects of the overhead imposed by grid middleware on both the throughput of the system
and on the turnaround times of a financial grid application. This approach is a step to-
wards producing a middleware independent, comparable, reproducible and fair performance
analysis of grid middlewares. The proposed benchmarks are self-sufficient for evaluating
any grid middleware with respect to its aforementioned non-functional aspects. The result
of such performance analysis can be used by middleware vendors to find the bottlenecks
and problems in their design and implementation of the system. Beside the performance
analysis, we also provide the mathematics proofs and numerical results in order to help the
financial application developers to verify the implementation of their financial algorithms.
Moreover, these benchmarks are derived from the real market data, hence, can also be used
by the computational finance community to propose novel parallel algorithms for pricing

[*] INRIA Sophia Antipolis – Université de Nice – CNRS - I3S
[†] Laboratoire de Mathématiques Appliquées aux Systèmes – Ecole Centrale de Paris

and hedging of high dimensional options more efficiently. In this technical report we explain the motivation and the details of the proposed benchmark suite. To our knowledge, this is the first attempt to make such benchmarks publicly available to both the communities. As a proof of concept, such benchmark suite was successfully used in an International Grid Programming contest, the 2008 Grid Plugtest and Contest which was organized by INRIA and finally we demonstrate the result of initial experiments during this event.

Key-words: financial benchmark suite, grid middlewares, performance analysis, option pricing, Monte Carlo methods

A financial engineering benchmark for performance analysis of grid middlewares

Résumé : Pas de résumé

Mots-clés : Pas de motclef

1   Introduction

Over the past few decades financial engineering has become a critical discipline and have gained strategic reputation for its own. Financial mathematicians keep coming up with novel and complex financial products and numerical computing techniques which often increase volume of data or computational time while posing critical time constraints for transactional processing. Generally, Monte Carlo (MC) simulations based methods are utilized to overcome typical problems like "curse of dimensionality" (e.g. integration over high dimensional space) [4]. Despite the ease of numerics, MC simulations come at the cost of tremendous computational demand in addition to slower convergence rates. However, advances in computer architectures like multi–core, many–cores, General Purpose Graphics Processing Units (GPGPUs) and their macro forms like clusters and federated Grids have made such MC simulations a handy tool for financial engineers [7]. Financial institution are using Grid computing to perform more time critical computations for competitive advantage. With this unprecedented computational capacity, running overnight batch processes for risk management or middle-office functions to re-evaluate whole product of portfolios have almost been out of fashion.

Grid middleware is what makes Grid computing work and easier to work with. It provides abstractions for core functionalities like authentication across large number of resources, authorization, resource matchmaking, data transfer, monitoring and fault–tolerance mechanisms in order to account for failure of resources . . . Any robust financial service operation cannot be achieved without paying a great attention to such issues. Current Grid middleware had its beginning in the Condor Project[1] and the Globus Alliance[2]. Recently, we have seen an upsurge of academic and commercial middleware providers such as gLite[3], ProActive/GCM Parallel Suite[4], Alchemi .NET Grid computing framework[5], Unicore[6] and KAAPI/TakTuk[7]. . . Now the question is which middleware to choose for gridifying financial applications? An obvious way is to devise a set of benchmarks and put different implementations through their paces. The middleware that results in the fastest computation could be declared as a winner. For this, one would need a standard well defined benchmark which would represent a wide set of financial algorithms, for instance MC based methods, and could also generate enough load on the middleware in test.

Benchmarks provide a commonly accepted basis of performance evaluation of software components. Performance analysis and benchmarking, however, is relatively young area in Grid computing compared to benchmarks designed for evaluating computer architecture. Traditionally, performance of parallel computer systems has been evaluated by strategically creating benchmark induced load on the system. Typically, such benchmarks comprise of

---

[1]http://www.cs.wisc.edu/condor/
[2]http://www.globus.org/
[3]http://glite.web.cern.ch/glite/
[4]http://proactive.inria.fr/
[5]http://sourceforge.net/projects/alchemi/
[6]http://www.unicore.eu/
[7]http://kaapi.gforge.inria.fr/

codes, workloads that may represent varied computations and are developed with different programming paradigms. Some examples are STREAM[8], LINPACK[9] and MPI Benchmarks[10], SPEC[11] and most popular NAS Parallel benchmark[12]. A key issue, however, is whether these benchmarks can be used "as is" for the Grid settings. The adoption of these benchmarks may raise several fundamental questions about their applicability, and ways of interpreting the results. Inherently, Grid is a complex integration of several functionally diverse components which may hinder evaluation of any individual component like middleware. Furthermore, in order to have fair evaluation, any benchmark would have to account for heterogeneity of resources, presence of virtual organizations and their diverse resource access policies, dynamicity due to inherent shared nature of the Grid. Such issues in turn have led to broader implications upon methodologies used behind evaluating middlewares as discussed in [1, 13]. In our work, however, for the sake of simplicity we assume the benchmark are run on dedicated Grid nodes. Thus, we primarily focus on quantifying performance of financial applications, achievable scalability, ease of deployment across large number of heterogeneous resources and their efficient utilization.

The goal of our work presented in this report is to design and develop SuperQuant Financial Benchmark Suite, a tool for researchers that wish to investigate various aspects of usage of Grid middlewares using well-understood benchmark kernels. The availability of such kernels can enable the characterization of factors that affect application performance, the quantitative evaluation of different middlewares, scalability of financial algorithms . . .

The rest of this report is organized as follows: in Section 2 we discuss the motivation behind SuperQuant Financial Benchmark Suite and propose guidelines for designing such benchmark. In Section 3 we describe the components of the benchmark suite. Section 4 presents the preliminary benchmark usage in a Grid Programming Contest. We conclude in Section 5.

2 SuperQuant Financial Benchmark suite

2.1 Motivation

In order to produce verifiable, reproducible and objectively comparable results, any middleware benchmark must follow the general rules of scientific experimentation. Such tools must provide a way of conducting reproducible experiments to evaluate performance metrics objectively, and to interpret benchmark results in a desirable context. The financial application developer should be able to generate metrics that quantify the performance capacity of Grid middleware through measurements of deployability, scalability, and computational capacity etc. Such metrics can provide a basis for performance tuning of application or the middleware. Alternatively, the middleware providers could utilize such benchmarks to make

---

[8]http://www.gridstream.org/
[9]http://www.netlib.org/benchmark/
[10]http://hcl.ucd.ie/project/mpiblib
[11]http://www.spec.org/mpi2007/press/release.html
[12]http://www.nas.nasa.gov/Resources/Software/npb.html

necessary problem specific software design changes. Hence, in order to formalize efforts to design and evaluate any Grid middleware, we designed a financial benchmark suite named SuperQuant.

2.2   Desired Properties

Some other considerations for the development of this benchmarks are described below and significantly follow the design guidelines of NAS benchmarks suite [2],

- Benchmarks must be conceptually simple and easy to understand for both financial and Grid computing community.

- Benchmarks must be "generic" and should not favor any specific middleware. Many middlewares provide different high level programming constructs such as tailored APIs or inbuilt functionalities like provision for parallel random number generators etc.

- The correctness of results and performance figures must be easily verifiable. This requirement implies that both input and output data sets must be limited and well defined. Since we target financial applications, we also need to consider real world trading and computation scenarios and data involved therewith. The problem has to be specified in sufficient detail and the required output has to be brief yet detailed enough to certify that the problem has been solved correctly.

- The problem size and runtime requirements must be easily adjustable to accommodate new middlewares or systems with different functionalities. The problem size should be large enough to generate considerable amount of computation and communication. In the kernel presented in this report, we primarily focus on the computational load while future benchmark kernels may impose communication as well as data volume loads.

- The benchmarks must be readily redistributable.

The financial engineer implementing the benchmarks with a given Grid middleware is expected to solve the problem in the most appropriate way for the given computing infrastructure. The choice of APIs, algorithms, parallel random number generators, benchmark processing strategies, resource allocation is left open to the discretion of this engineer. The languages used for programming financial systems are mostly C,C++ and Java. Most of the Grid middlewares are available in these languages and the application developers are free to utilize language constructs that, they think give the best performance possible or any other requirements imposed by the business decisions, on the particular infrastructure available at their organization.

3   Components of SuperQuant Financial Benchmark Suite

Our benchmark suite consists of three major components :

- An embarrassingly parallel kernel

- The input/output data and Grid metric descriptors

- An output evaluator

Each of these components are briefly described in the following sections.

3.1　Embarrassingly Parallel Kernel

We have devised a relatively "simple" kernel which consists of a batch of high dimensional vanilla and barrier options. The objective is to compute price and Greeks of maximum number of options with acceptable accuracy and within definite time interval using MC based methods. The algorithm, pseudocodes and an exemplary parallel version of MC based pricing method are provided along with the benchmark suite and are available on our website[13].

The kernel is based on computationally intensive financial problems, pricing and hedging of high dimensional European options. Such European option pricing and hedging have widespread applications related to both financing and investing decisions in financial as well as commodity markets since their first trades in CBOE[14] in 1973. In financial engineering, Monte Carlo methods has been widely applied in option pricing because of its well adaptability for various payoffs and particularly for high dimensional underlying assets. The definitions of financial terms in this section can be found in common financial engineering textbooks [8, 14], although reader may find the following information self–explanatory.

3.1.1　European Option Pricing

The Black–Scholes (BS) model is a simple model to describe the evolution of a basket of assets price through a system of stochastic differential equations (SDEs) [10],

$$dS_t^i = S_t^i(r - \delta_i)dt + S_t^i \sigma_i dB_t^i, \ i = 1, \dots, d, \text{ where} \tag{1}$$

- $S = \{S^1, \dots, S^d\}$ is a basket of d assets.

- r is the constant interest rate for every maturity date and at any time.

- $\delta = \{\delta_1, \dots, \delta_d\}$ is a constant dividend vector.

- $B = \{B^1, \dots, B^d\}$ is a correlated d-dimensional Brownian Motion (BM).

- $\sigma = \{\sigma_1, \dots, \sigma_d\}$ is a constant volatility vector.

---

[13]http://www-sop.inria.fr/oasis/plugtests2008/ProActiveMonteCarloPricingContest.html
[14]http://www.cboe.com/ - The Chicago Board Options Exchange

A European option is a contract which can be exercised only at a fixed future date T with a fixed price K. A call (or put) option gives option holder right (not the obligation) to buy (or sell) underlying asset at the date T. At T, exercised option contract will pay to the option holder a position payoff $\Phi(f(S_T))$ which depends only on the underlying asset price at the maturity date S_T (for Vanilla option) or $\Phi\big(f(S_t, t \in [0,T])\big)$ which depends on the entire underlying asset trajectories price S_t (for Barrier option). The definition of $f(\cdot)$ is given by the option's payoff type (Arithmetic Average, Maximum, or Minimum) [8, 14]. According to the Arbitrage Pricing Theory [10], the fair price V for the option contract is given by the following expression

$$V(S_0, 0) = \mathbb{E}\big[e^{-rT}\Phi\big(f(S_t, t \in [0,T])\big)\big].$$

This expectation value is approximated by using MC simulation based methods [7]. We have such that

$$V(S_0, 0) \simeq \frac{1}{nbMC} \sum_{j=1}^{nbMC} e^{-rT}\Phi\big(f(S_t^{(j)}, t \in [0,T])\big) \qquad (2)$$

where $\big(S_t^{(j)}\big)$ are independent trajectories of the solution of (1) and $nbMC$ is the number of Monte Carlo simulations. The law of large number of Monte Carlo methods implies that

$$\lim_{nbMC \to \infty} \frac{1}{nbMC} \sum_{j=1}^{nbMC} e^{-rT}\Phi\big(f(S_t^{(j)}, t \in [0,T])\big) \to V(S_0, 0)$$

with the probability 1. To illustrate an option pricing application using MC methods, we consider the following pseudo–code for a call Geometric Average (GA) option pricing in Algorithm 1. The Grid based approach for such option pricing using MC methods can be found in [5].

3.1.2   European Greeks Hedging

The Greeks represent sensitivities of option price with respect to market parameters like asset price, time remained to maturity, volatility, or interest rate. Usually Greeks are derivatives of first or second order that are computed using finite difference methods [7]. Greeks are not observed in the real time market but, are informations that needs to be computed with accuracy. We refer to [8, 14] for complete and detail explanations of Greeks such as Delta (Δ), Gamma (Γ), Rho (ρ) and Theta (θ). First and second order derivative are approximated respectively by using finite difference methods as follows :

$$\mathrm{Greek}^{(1)}(x) = \frac{V(S_0, 0)|_{x+\epsilon_x} - V(S_0, 0)|_{x-\epsilon_x}}{2x\epsilon_x}$$

$$\mathrm{Greek}^{(2)}(x) = \frac{V(S_0, 0)|_{x+\epsilon_x} - 2V(S_0, 0) + V(S_0, 0)|_{x-\epsilon_x}}{(x\epsilon_x)^2}$$

where x can consists one among the market parameters mentioned above. To be more clarified, we consider the GA option pricing in Algorithm 1 above, let us denote $V(S_0, 0)|_{S_0^i \pm \epsilon_S}$

Algorithm 1 Pricing a call GA option of d assets

Require: S_0^i, d, r, δ_i, σ_i, N_T and number of simulations $nbMC$

1: **for** $j = 1$ to $nbMC$ **do**

2:     $f(S_T) = 1$

3:     **for** $i = 1$ to d **do**

4:         $S_T^i = S_0^i \exp\left(\left((r - \delta_i) - \sigma_i^2/2 \right)T + \sqrt{T}\sigma_i B_T^i \right)$

5:         $f(S_T) = f(S_T) \times S_T^i$

6:     **end for**

7:     $f(S_T) = \sqrt[d]{f(S_T)}$

8:     $C_j = e^{-rT}\left(f(S_T) - K \right)^+$

9:     *In order to calculate the variance, we also compute $C_j{}^2$*

10:     $C_j{}^2 = \left(e^{-rT}\left(f(S_T) - K \right)^+ \right)^2$

11: **end for**

12: return $\widehat{C} = \dfrac{C_1 + \cdots + C_{nbMC}}{nbMC} \equiv V(S_0, 0)$

13: return $\widehat{C}^2 = \dfrac{C_1^2 + \cdots + C_{nbMC}^2}{nbMC}$

---

the option prices with the respect to the change of the asset price S^i, $V(S_0, 0)|_{r \pm \epsilon_r}$ and $V(S_0, 0)|_{\tau \pm \epsilon_\tau}$ the option prices with respect to the change of the interest rate r and of the time remained to maturity τ. Algorithm 2 below presents the pseudo code for the Greeks hedging by using finite difference methods. The vector of Δ and the matrix of Γ respectively are the first and second order derivatives of $V(S_0, 0)$ with respect to the change of each asset price among the basket. To simplify the computation, we only compute the diagonal of the matrix Γ. The two last Greeks ρ and θ are the first order derivatives of $V(S_0, 0)$ with respect to the change of r and τ.

3.1.3   The Composition of the Benchmark Kernel

The core benchmark kernel consists of a batch of 1000 well calibrated *TestCase*s. Each *TestCase* is a multi–dimensional European option with up to 100 underlying assets with necessary attributes like spot prices, payoffs types, time to maturity, volatility, and other market parameters. In order to constitute an option, the underlying assets are chosen from a pool of companies listed in the equity S&P500 index[15], while volatility of each asset and its dividend rate are taken from CBOE. In order to balance the computational time, the composition of the batch is as follows,

- 500 *TestCase*s of 10–dimensional European options with 2 years time to maturity

- 240 *TestCase*s of 30–dimensional European options with 9 months time to maturity

---

[15] http://www2.standardandpoors.com

Algorithm 2 Delta, Gamma, Rho and Theta hedging for a call GA option of d assets

---

Require: $V(S_0, 0)$, $V(S_0, 0)|_{S_0^i \pm \epsilon_S}$, $V(S_0, 0)|_{r \pm \epsilon_r}$ and $V(S_0, 0)|_{\tau \pm \epsilon_\tau}$

Ensure: Δ, Γ, ρ, θ

1: **for** $i = 1$ to d **do**

2: $\quad \Delta^i = \dfrac{V(S_0, 0)|_{S_0^i + \epsilon_S} - V(S_0, 0)|_{S_0^i - \epsilon_S}}{2 S_0^i \epsilon_S}$

3: $\quad \Gamma^{ii} = \dfrac{V(S_0, 0)|_{S_0^i + \epsilon_S} - 2 V(S_0, 0) + V(S_0, 0)|_{S_0^i - \epsilon_S}}{(S_0^i \epsilon_S)^2}$

4: **end for**

5: $\rho = \dfrac{V(S_0, 0)|_{r + \epsilon_r} - V(S_0, 0)|_{r - \epsilon_r}}{2 r \epsilon_r}$

6: $\theta = \dfrac{V(S_0, 0)|_{\tau + \epsilon_\tau} - V(S_0, 0)|_{\tau - \epsilon_\tau}}{2 \tau \epsilon_\tau}$

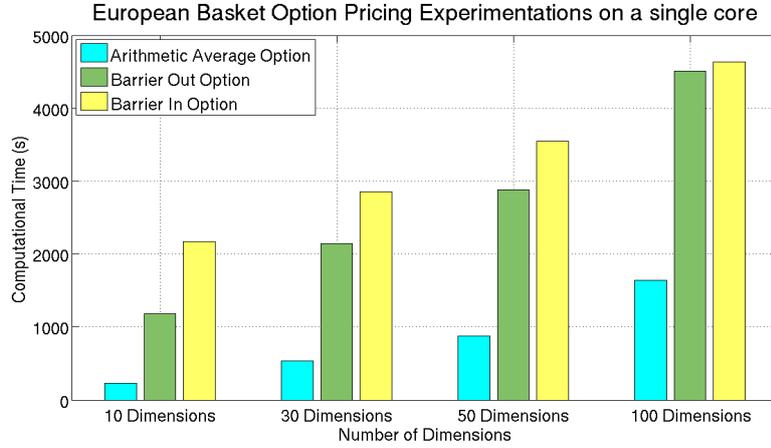7: **return** Δ, Γ, ρ, θ

Figure 1: Computational time of several European basket option pricings on a single core (Intel Xeon(R), E5335, 2.00GHz, 2G RAM, JDK 1.6)

- 240 *TestCase*s of 50–dimensional European options with 6 months time to maturity

- 20 *TestCase*s of 100–dimensional European options with 3 months time to maturity

In Figure 1 we present the computational time on a single core for each type of option within the benchmark suite. Thus, the objective of the benchmark is pricing and hedging of maximum number of *TestCase*s by implementing the algorithms using a given Grid middleware.

3.2   Input/Output Data and Grid Metrics Format

To facilitate processing, exchanging and archiving of input data, output data and Grid related metrics, we define relevant XML data descriptors. The *TestCases* required by the kernel and the "reference" results are also included in the benchmark suite.

- Input ***AssetPool :*** represents the database of 250 assets required to construct a basket (collection) option of assets

- Input ***CorrelationMatrix :*** defines a correlation matrix of the assets in *AssetPool*. The provided matrix is positive–definite with diagonal values 1 and correlation coefficients in the interval of $[-1, 1]$. The calibration of a historical correlation matrix is described in Appendix C.

- Input ***TestCases :*** defines a set of *TestCase*s, input parameters, needed by the pricing and hedging algorithm discussed above. Each *TestCase* includes parameters such as an option, which is a subset of *AssetPool*, a submatrix of *CorrelationMatrix*, type of payoff, type of option, barrier value if needed, interest rate, maturity date . . .

- Output ***Results :*** defines a set of *Result* which consists of *Price* and *Greeks* of individual *TestCase* and time *Metrics* required to compute each output values.

- Output Grid ***Metrics :*** defines the total time required for the entire computation.

3.3   Output Evaluator

The output evaluator is a tool to compare the results computed by different implementations of the benchmark kernel *TestCases* with "reference" results provided in the suite.

3.3.1   Evaluation Criteria

In order to measure the precision, the results are estimated with a confidence interval of 95% [7]. Consider a call option pricing in Algorithm 1. The estimator $\widehat{C}$ is unbiased, in the sense that its expectation is the target quantity

$$\mathbb{E}\big[\widehat{C}\big] = V(S_0, 0) \equiv \mathbb{E}\big[\exp(-rT)\big(f(S_T) - K\big)^+\big]$$

The estimator is strongly consistent, in the sense that

$$\widehat{C} \rightarrow V(S_0, 0) \text{ with probability 1, when nbMC} \rightarrow \infty$$

Hence, for a fixed number of MC simulations the Central Limit Theorem gives us the rate of convergence such that

$$\frac{nbMC(\widehat{C} - V(S_0, 0))}{\widehat{s}_C} \rightarrow N(0, 1) \text{ when nbMC} \rightarrow \infty$$

where the estimator $\widehat{s}_C$ for the standard deviation is computed as

$$\widehat{s}_C = \sqrt{\left(\frac{1}{nbMC} \sum_{j=1}^{nbMC} C_j^2\right) - \left(\frac{1}{nbMC} \sum_{j=1}^{nbMC} C_j\right)^2}.$$

The value $\widehat{C}$ is obtained with a 95% confidence within the following interval $\left[\pm 1.96 \frac{\widehat{s}_C}{\sqrt{nbMC}}\right]$. We decide the tolerable relative error in computing the results is 10^{-3} which produces a number of MC simulations of $\frac{10^6}{\widehat{s}_C}$. Since the accuracy of the computed results relies on the spot prices of the underlying assets, we consider relative errors with respect to the "reference" results (see below). These "reference" results are computed with sufficiently large number of MC simulations in order to achieve lower confidence interval. The **Output Evaluator** employs a point based scheme to grade the results and also provides a detailed analysis of points gained per *TestCase*. Thus we have outlined the following points based system :

- The main evaluation criteria is the total number of finished testcases, say M, that are priced during the assigned time slot. For each price computed, the team gets $+10$ points. Thus a team can earn up to $+10 \times M$ points.

- If the computed price is within the expected precision, the team gains $+5$ points

- If the computed price is above the expected precision, the team gains $+10$ points

- If the computed price is below the expected precision, the team is penalized with -10 points.

- For each Greek letter, namely Delta, Gamma, Rho and Theta that is precisely computed, the team will get $+2$ points per Greek letter. The Greek letters must be computed by a finite difference method with a fixed step size. Note that if non–precise, the values will not be given any points.

- For each minute saved out of the assigned time slot, the team will gain $+1$ point.

3.3.2 "Reference" Results Validation

The "reference" results provided in the benchmark suite are not analytical results and are computed by using MC based methods. It is well known in BS model that there only exists analytical solution for European option pricing in case of one dimension otherwise in case of multi–dimension we have to use other approximation approaches such as MC methods.

We observed that in some very specific cases we can analytically reduce a basket of assets into a one–dimensional "reduced" asset, further details of the reduction technique are given in Appendix B.1 and B.2. The exact option price on this "reduced" asset can be computed by using BS formula [6]. Thus in such particular cases, we can validate the "reference" results bases on the relative error with the exact results. These "reference" results will be validated once they achieve better accuracy than the tolerable relative error 10^{-3} otherwise we can calibrate the number of MC simulations until they satisfy such condition. This calibration helps us to have an idea about a large enough number of MC simulations for other "reference" results which do not have any analytical solution for comparison. To highlight the usefulness of this approach, we provide below a numerical example.

Numerical Example : Consider a call/put GA option of 100 independent assets ($d = 100$) with prices modeled by SDEs (1). The parameters are given as $S_0^i = 100, i = 1, \ldots, 100$, $K = 100$, $r = 0.0$, $\delta_i = 0.0$, $\sigma_i = 0.2$ and $T = 1$ year. The basket option is simulated by using 10^6 MC simulations by using Algorithm 1. The "reduced" asset is $\Sigma_t = \prod_{i=1}^d S_t^{i \frac{1}{d}}, i = 1, \ldots, 100$ and it is the solution of the one–dimensional SDE: $d\Sigma_t/\Sigma_t = \left(\widetilde{\mu}dt + \widetilde{\sigma}dZ_t\right)$ where $\widetilde{\mu} = r + \frac{\sigma^2}{2d} - \frac{\sigma^2}{2}, \widetilde{\sigma} = \frac{\sigma}{\sqrt{d}}$ and Z_t is a Brownian Motion. The parameters of Σ are given as $\Sigma_0 = 100, \widetilde{\mu} = 0.0198, \widetilde{\sigma} = 0.02$. We are interested in comparing the estimated option price V of d assets with the analytical "reduced" one $\widetilde{V}$ on Σ. We denote the absolute error $\Delta V = |V - \widetilde{V}|$, then the relative error η is computed as $\eta = \frac{\Delta V}{\widetilde{V}}$.

In Table 1, the first column represents the estimated option prices using MC methods and their 95% confidence interval. The second column gives the analytical exact option prices. The last two columns show the absolute and relative errors. As it can be observed, the relative error in case of put option pricing is less than 10^{-3}, therefore such put price is validated. Meanwhile the call price is not, thus to validate the call option price we have to increase the number of MC simulations. The Table 2 shows that such call option price is validated with 10^8 MC simulations. Bases on this calibration analysis, we can find, for each "reference" result in the benchmark suite, an optimal number of MC simulations which can produce a relative error less than 10^{-3} (e.g. for any call option pricing on upto 100 underlying assets, we should consider at least 10^8 MC simulations).

In this example, we also consider the Delta hedging for both MC based option pricing and analytical option pricing. In the first case, the Delta hedging produces a vector of 100 first order derivatives of the option price $\left(\Delta^i, i = 1, \ldots, 100\right)$ with respect to the change of each asset price among 100 assets. Following the initial parameters, every assets have the same spot price and volatility rate, hence Δ^i, $\forall i$ are uniform. Such Delta values are computed by using finite difference methods as described in Algorithm 2. In the later case, the Delta value $\widetilde{\Delta}$ is computed by using an analytical formula [8]. The relation between Δ^i

Table 1: Call/Put price of a GA of 100 assets option using 10^6 MC simulations and of the "reduced" option

Call MC Price V (95% CI)	"Reduced" Call Price $\widetilde{V}$	Absolute error	Relative error
0.16815 (0.00104)	0.16777	3.8×10^{-4}	2.2×10^{-3}
Put MC Price V (95% CI)	"Reduced" Put Price $\widetilde{V}$	Absolute error	Relative error
2.12868 (0.00331)	2.12855	1.3×10^{-4}	6.1×10^{-5}

Table 2: Call price of a GA of 100 assets option using 10^8 MC simulations and of the "reduced" option

Call MC Price V (95% CI)	"Reduced" Call Price $\widetilde{V}$	Absolute error	Relative error
0.16793 (0.00011)	0.16777	1.6×10^{-4}	9.5×10^{-4}

Table 3: Delta values of the Call/Put GA of 100 assets and of the "reduced" one

Basket MC Call Delta Δ^i	"Reduced" Call Delta $\widetilde{\Delta}$
0.00160	0.16030
Basket MC Put Delta Δ^i	"Reduced" Put Delta $\widetilde{\Delta}$
-0.00818	-0.82010

and $\widetilde{\Delta}$ is described as follows,

$$\prod_{i=1}^{d} \Delta^i = \left(\frac{1}{d}\widetilde{\Delta}\right)^d.$$

Further details of this relation are given in Appendix B.3 and B.4. In Table 3, we present the numerical results of the Delta hedging for both options.

Though such numerical example, we are able to validate our "reference" results in the benchmark suite and also to verify the option pricing implementation.

4 Proof of Concept : The V Grid Plugtest and Contest

As a proof of concept, we used the SuperQuant Benchmark Suite for the **2008 SuperQuant Monte Carlo Challenge** organized as a part of **V GRID Plugtest**[16] at INRIA Sophia Antipolis. The details of the contest and the benchmark input data can be found on the Plugtest and Challenge website[17]. Each participant was given an exclusive one hour ac-

---

[16] http://www.etsi.org/plugtests/GRID2008/About_GRID.htm
[17] http://www-sop.inria.fr/oasis/plugtests2008/ProActiveMonteCarloPricingContest.html

cess for evaluating the benchmark on two academic Grids, Grid'5000[18] and InTrigger[19], which combined consisted around 5000 computational cores geographically distributed across France and Japan. The description of Grid'5000 resources which were provided during the contest is given in Table 4.

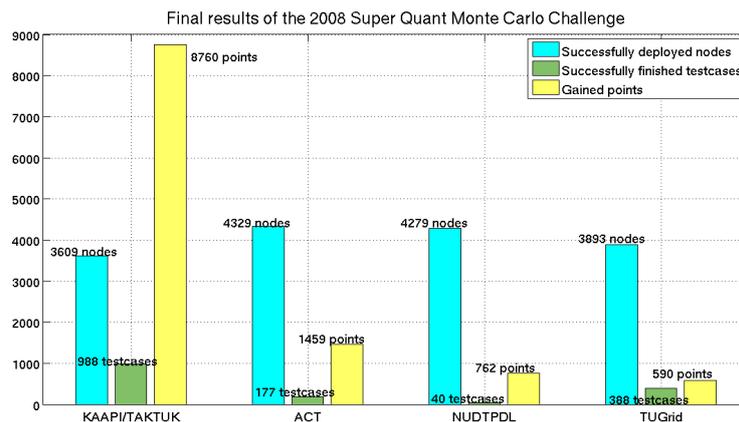**Challenge Results :** Figure 2 presents the final results of the Challenge. The partic-

Figure 2: Final results of the 2008 SuperQuant Monte Carlo challenge

ipants primarily used two middlewares, ProActive, an open source Java based Grid middleware and KAAPI/TAKTUK, which couples KAAPI , a Parallel Programming Kernel and TAKTUK, a middleware for adaptive deployment. As we can see in Figure 2, the KAAPI/TAKTUK team was successful in computing 98.8% of total number of *TestCases* and was also able to deploy application on a significantly large number of nodes. The other teams used ProActive to implement the benchmark kernel. Both middlewares implement the Grid Component Model (GCM) deployment model, recently standardized by the ETSI GRID[20] technical committee for deploying the application over large number Grid nodes [3]. The infrastructure descriptors and application descriptors required by GCM deployment were bundled with the benchmark suite. From Figure 2, we can observe that the benchmarks were not only useful to quantitatively compare two middleware solutions, but also gave the opportunity to evaluate different benchmark implementations using the same middleware. Such comparison is useful not only to middleware providers but also to Grid application developers.

---

[18]https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home
[19]https://www.intrigger.jp/wiki/index.php/InTrigger
[20]http://www.etsi.org/WebSite/Technologies/GRID.aspx

Table 4: Grid'5000 configuration for the "2008 SuperQuant Monte Carlo" challenge. Total number of machines is 1244

Location	# of machines	CPU	# of CPUs	OS	Gflops	JVM
Grid5000 Bordeaux	48	Opteron 248	2	Fedora Core 4	19.4	Sun 1.5.0_08 amd64
Grid5000 Lille	15	Opteron 252	2	Red Hat EL3	29.7	Sun 1.5.0_08 amd64
	53	Opteron 248	2	Red Hat EL3	29.7	Sun 1.5.0_08 amd64
Grid5000 Lyon	56	Opteron 246	2	Debian 3.1	20.9	Sun 1.5.0_08 amd64
	70	Opteron 250	2	Debian 3.1	14.6	Sun 1.5.0_08 amd64
Grid5000 Nancy	47	Opteron 246	2	Debian testing	15.6	Sun 1.5.0_08 amd64
Grid5000 Orsay	216	Opteron 246	2	Debian testing	unspecified	Sun 1.5.0_08 amd64
	126	Opteron 250	2	Debian testing	unspecified	Sun 1.5.0_08 amd64
Grid5000 Rennes	64	Xeon IA32 2.4Ghz	2	Debian testing	unspecified	Sun 1.5.0_08 i586
	99	Opteron 246	2	Debian testing	unspecified	Sun 1.5.0_08 amd64
	64	Opteron 248	2	Debian 3.1	26.6	Sun 1.5.0_08 amd64
	64	G5 2Ghz	2	OS X	unspecified	Sun 1.5.0_06 PowerPC
Grid5000 Sophia	105	Opteron 246	2	Rocks Linux	36.5	Sun 1.5.0_08 amd64
	56	Opteron 275	2	Rock Linux	25.9	Sun 1.5.0_08 amd64
Grid5000 Toulouse	58	Opteron 248	2	Fedora Core 3	unspecified	Sun 1.5.0_08 amd64
Grid5000 Grenoble	103	Itanium 2 1Ghz	2	Red Hat EL3	unspecified	Sun 1.5.0_08 i586

5 Conclusion and Perspectives

In this report we have presented SuperQuant Financial Benchmark Suite for performance evaluation and analysis of Grid middlewares in the financial engineering context. We described the preliminary guidelines for designing the benchmark. We also described the benchmark constituents along with a brief overview of the embarrassingly parallel benchmark kernel. As a proof of concept, we also utilized this benchmark in a Grid Programming Contest. Although this is a preliminary proposal for this benchmark, the specification of more complex kernels that can induce inter-cluster communication, high speed I/O requirements, or data processing, is necessary for truly understanding the overhead imposed by Grid middlewares in financial applications.

A   Simulation of correlated Brownian Motions

Consider a basket of d assets whose prices are typically driven by the Black–Scholes model

$$dS_t^i = S_t^i r dt + S_t^i \sigma_i dB_t^i, \ i = 1, \ldots, d; \ t \in [0, T], \tag{A-1}$$

given in (1). We complete the model description with the correlation matrix $\left(\rho_{ij}, \ i, j = 1, \ldots, d \right)$ of the Brownian Motion B, such that $\rho_{ij} = \frac{\mathbb{E}(B_t^i, B_t^j)}{t}$ so $Covariance(B_t^i, B_t^j) = \rho_{ij} t$. The calibration of ρ_{ij} will be discussed in Appendix C. We define the $d \times d$ covariance matrix Cov by,

$$Cov_{ij} = \sigma_i \sigma_j \rho_{ij}. \tag{A-2}$$

We aim to rewrite equation (A-1) by the following equation (A-3),

$$dS_t^i = S_t^i r dt + S_t^i \sum_{k=1}^{d} a_{ik} dW_t^k \tag{A-3}$$

where $\left(a_{ik}, i, k = 1, \ldots, d \right) = A$, such that $AA^t = Cov$, thus

$$Cov_{ij} = \sum_{k=1}^{d} a_{ik} a_{jk}, \ i, j = 1, \ldots, d. \tag{A-4}$$

Note that A exists, as Cov is always a positive–definite matrix. By applying Itô Lemma [10] for (A-3), we have

$$S_t^i = S_0^i \exp \left((r - \frac{1}{2} \sigma_i^2) t + \sum_{k=1}^{d} a_{ik} dW_t^k \right)$$

A trajectory realization of asset prices at discrete dates $0 < t_1 < t_2 \ldots < T$ is obtained by setting

$$S_{t_{n+1}}^i = S_{t_n}^i \exp \left((r - \frac{1}{2} \sigma_i^2)(t_{n+1} - t_n) + \sqrt{(t_{n+1} - t_n)} \sum_{k=1}^{d} a_{ik} Z_{t_{n+1}}^k \right) \tag{A-5}$$

where $\left(t_n = \frac{n}{N_T}, n = 0 \ldots, N_T \right)$ is a regular partition of the interval $[0, T]$, $\left(Z_n = \left(Z_n^1, \ldots, Z_n^d \right), n = 0 \ldots, N_T \right)$ is a familly of independent Gaussian variables of law $N(0, Id)$ and $\left(S_{t_n}^i, i = 1, \ldots, d; \ n = 1, \ldots, N_T \right)$ is a realizations of the assets price trajectories.

A.1 Construction of a correlated d–dimensional Brownian Motion with a standard q–dimensional one

Consider a standard q–dimensional Brownian Motion $W = (W^1, \ldots, W^q)$, where each W^i is independent to each other. Consider a matrix $(\alpha_{ik}, i = 1, \ldots, d; \ k = 1, \ldots, q)$ be a constant $d \times q$ matrix such that $\sigma_i = \left[\sum_{k=1}^q \alpha_{ik}^2 \right]^{\frac{1}{2}}$ and suppose that $\sigma_i > 0, \ \forall i = 1, \ldots, d$. We define processes $B^i, i = 1, \ldots, d$ by

$$B_t^i = \sum_{k=1}^d \int_0^t \frac{\alpha_{ik}}{\sigma_i} dW_u^k. \tag{A-6}$$

We are going to show that $(B^1, \ldots, B^d)$ is a d–dimensional correlated Brownian Motion with correlation matrix $\left(\rho_{ij} = \frac{1}{\sigma_i \sigma_j} \sum_{k=1}^q \alpha_{ik} \alpha_{jk}, i, j = 1, \ldots, d \right)$. Since $B_0 \equiv 0$ and B_t^i has continuous paths, it suffices to show that $dB_t^i dB_t^i = dt$ in order to prove that B^i is a Brownian Motion according to the Paul Levy calculation of Brownian Motion [10]. Indeed, from (A-6) we have

$$dB_t^i = \sum_{k=1}^q \frac{\alpha_{ik}}{\sigma_i} dW_t^k = \frac{1}{\sigma_i} \sum_{k=1}^q \alpha_{ik} dW_t^k \tag{A-7}$$

As W^i is a standard Brownian Motion, we have

$$\mathbb{E}\left[W_t^i W_t^i \right] = t$$
$$\mathbb{E}\left[W_t^i W_t^j \right] = 0, \ i \neq j$$

Hence, by (**??**) and Itô Lemma

$$\mathbb{E}\left[B_t^i B_t^i \right] = \sum_{k=1}^q \frac{\alpha_{ik}^2}{\sigma_i^2} \mathbb{E}\left[W_t^k W_t^k \right] = \frac{1}{\sigma_i^2} \sum_{k=1}^q \alpha_{ik}^2 t = t \tag{A-8}$$

We also compute the correlation of (B^i, B^j), we have

$$\mathbb{E}\left[B_t^i B_t^j \right] = \sum_{k=1}^q \frac{\alpha_{ik} \alpha_{jk}}{\sigma_i \sigma_j} \mathbb{E}\left[W_t^k W_t^k \right] = \frac{1}{\sigma_i \sigma_j} \sum_{k=1}^q \alpha_{ik} \alpha_{jk} t \tag{A-9}$$

From (A-8) and (A-9), we proved that $(B^1, \ldots, B^d)$ is a d–dimensional correlated Brownian Motion. Now we come back to the Black–Scholes model A-1. For a given correlation matrix ρ_{ij}, we compute a_{ij} such that $\sigma_i \sigma_j \rho_{ij} = \sum_{k=1}^d a_{ik} a_{ik}$. From A-7 by changing α with a, we can identify $(B_t^1, \ldots, B_t^d)$ as $B_t^i = \frac{1}{\sigma_i} \sum_{k=1}^d a_{ik} W_t^k$. Replacing B_t^i in A-1 we get A-3.

B   Reduction of the dimension in basket option pricing

In this section, we discuss about the reduction dimension problem used in order to validate the application of the 2008 "SuperQuant Monte Carlo" Challenge. Consider a probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t, t > 0), \mathbb{P})$, equipped with a d–dimensional standard Brownian Motion $W_t = (W_t^1, ..., W_t^d)$. Consider a free risk asset $S_t^0 = e^{rt}$, with fixed interest rate r, and a basket of d assets $S_t = (S_t^1, ..., S_t^d)$, solution of the SDEs

$$dS_t^i = S_t^i r dt + S_t^i \sum_{k=1}^{d} a_{ik} dW_t^k, \ i = 1, \ldots, d \tag{B-10}$$

where $a_{ik}, i, k = 1, \ldots, d$ are constant.

B.1   Payoff function as product of assets

We consider an European option on a basket of d assets S_t, with a payoff function $\Phi(\Sigma_t)$ depending only on the "reduced" variable $\Sigma_t = f(S_t)$, where

$$f(x) = \prod_{i=1}^{d} x_i^{\alpha_i}, \forall x = (x_1, ..., x_d) \tag{B-11}$$

with a given set $(\alpha_1, ..., \alpha_d) \in \mathbb{R}_+$. We aim to find the SDE satisfied by the "reduced" asset Σ_t in dimension one. To do so, we apply the multi–dimensional Ito Lemma to $f(S_t)$, we get

$$df(S_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial s_i}(S_t) dS_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial s_i \partial s_j}(S_t) S_t^i S_t^j \sum_{k=1}^{d} a_{ik} a_{jk} dt \tag{B-12}$$

With $f(\cdot)$ defined in (B-11), we compute the first term of (B-12),

$$\sum_{i=1}^{d} \frac{\partial f}{\partial s_i}(S_t) dS_t^i = \prod_{i=1}^{d} (S_t^i)^{\alpha_i} \sum_{i=1}^{d} \frac{\alpha_i}{S_t^i} dS_t^i$$

by using the definition of dS_t^i in (B-10), we have

$$\sum_{i=1}^{d} \frac{\partial f}{\partial s_i}(S_t) dS_t^i = \prod_{i=1}^{d} (S_t^i)^{\alpha_i} \sum_{i=1}^{d} \frac{\alpha_i}{S_t^i} S_t^i \left(r dt + \sum_{k=1}^{d} a_{ik} dW_t^k \right)$$

$$= \prod_{i=1}^{d} (S_t^i)^{\alpha_i} \sum_{i=1}^{d} \alpha_i (r dt + \sum_{k=1}^{d} a_{ik} dW_t^k)$$

We compute the second term of (B-12), we get

$$\prod_{i=1}^{d} (S_t^i)^{\alpha_i} \sum_{i,j=1}^{d} \left[\left((\frac{\alpha_i \alpha_j}{S_t^i S_t^j})_{i \neq j} + (\frac{\alpha_i(\alpha_i - 1)}{S_t^i S_t^j}))_{i=j} \right) S_t^i S_t^j \sum_{k=1}^{d} a_{ik} a_{jk} \right] dt$$

Hence, by identifying $\Sigma_t = f(S_t) = \prod_{i=1}^{d}(S_t^i)^{\alpha_i}$ then Equation (B-12) becomes:

$$\frac{d\Sigma_t}{\Sigma_t} = \left(\sum_{i=1}^{d} \alpha_i r + \frac{1}{2} \sum_{i,j=1}^{d} \left((\alpha_i \alpha_j)_{i \neq j} + (\alpha_i(\alpha_i - 1))_{i=j} \right) \sum_{k=1}^{d} a_{ik} a_{jk} \right) dt$$
$$+ \sum_{i,k=1}^{d} \alpha_i a_{ik} dW_t^k \tag{B-13}$$

Considering the process X_t defined by $X_t = \sum_{i,k=1}^{d} \alpha_i a_{ik} W_t^k$, then Equation (B-13) reduces to

$$\frac{d\Sigma_t}{\Sigma_t} = \underbrace{(r - \widehat{\delta})}_{\widehat{\mu}} dt + dX_t \tag{B-14}$$

where

$$\widehat{\delta} = r - \left(\sum_{i=1}^{d} \alpha_i r + \frac{1}{2} \sum_{i,j=1}^{d} \left((\alpha_i \alpha_j)_{i \neq j} + (\alpha_i(\alpha_i - 1))_{i=j} \right) \sum_{k=1}^{q} a_{ik} a_{jk} \right)$$

could be viewed as the dividend yield by the "reduced" asset Σ.

B.2 The particular case of Geometric Average of d assets

We consider the particular case $\alpha_i = \dfrac{1}{d}$, $i = 1, \ldots, d$ and $S^i, i = 1, \ldots, d$ are independent assets. This means that $a_{ik} = 0$, for $i \neq j$ and for a given $\sigma > 0$ we set $a_{ii} = \sigma$, $\forall i = 1, \ldots d$. Now we get

$$f(x) = \prod_{i=1}^{d} x_i^{\frac{1}{d}} \tag{B-15}$$

Start from (B-14)

$$\frac{d\Sigma_t}{\Sigma_t} = \underbrace{(r - \widehat{\delta})}_{\widehat{\mu}} dt + dXt$$

with now

$$\widehat{\delta} = \left(\frac{\sigma^2}{2} - \frac{\sigma^2}{2d} \right)$$

and

$$X_t = \sum_{i=1}^{d} \sum_{k=1}^{d} \alpha_i a_{ik} W_t^k = \sum_{i=1}^{d} \frac{1}{d} a_{ii} W_t^i = \frac{1}{d} \sigma \sum_{i=1}^{d} W_t^i.$$

Define $Z_t = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} W_t^i$. Then standard computations show that $\mathbb{E}\left[Z_t Z_s \right] = t \wedge s$ which implies Z_t as a standard BM on the probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t, t > 0), \mathbb{P})$. Finally, the

equation (B.2) becomes,

$$\frac{d\Sigma_t}{\Sigma_t} = \left(r + \frac{\sigma^2}{2d} - \frac{\sigma^2}{2}\right)dt + \frac{\sigma}{\sqrt{d}}dZ_t \tag{B-16}$$

The asset Σ_t is said to follow a geometric Brownian Motion. By applying the Ito Lemma for the function $F(\Sigma_t) = \log(\Sigma_t)$ and assuming that $\Sigma_t > 0, \forall t$, we have

$$d\log\Sigma_t = \left(\underbrace{r + \frac{\sigma^2}{2d} - \frac{\sigma^2}{2}}_{\widetilde{\mu}} - \underbrace{\frac{\sigma^2}{2d}}_{\frac{1}{2}\widetilde{\sigma}^2}\right)dt + \underbrace{\frac{\sigma}{\sqrt{d}}}_{\widetilde{\sigma}}\,dB_t$$

$$\log\Sigma_t = \log\Sigma_0 + \left(\widetilde{\mu} - \frac{\widetilde{\sigma}^2}{2}\right)t + \widetilde{\sigma}B_t$$

which leads to the explicit solution

$$\Sigma_T^{\Sigma_t,t} = \Sigma_t \exp\left(\left(\widetilde{\mu} - \frac{\widetilde{\sigma}^2}{2}\right)(T-t) + \widetilde{\sigma}B_T\right), \forall t \in [0,T]. \tag{B-17}$$

B.3 Option price formula for one–dimensional BS European option

We recall shortly some basic explicit formulas of financial engineering. Consider a call European option on the asset Σ_t modeled by Equation (B-17). We can compute such option value by using the Black Scholes formula. The call option value at time t is,

$$\widetilde{V}(\Sigma_t, t) = \mathbb{E}\big[\Phi(\Sigma_T^{\Sigma_t,t})\big]$$

with $\Phi(x) = (x - K)^+$. Then a simple computation leads to

$$\widetilde{V}(\Sigma_t, t) = \Sigma_t N(d_1) - K\exp(-r(T-t))N(d_2) \tag{B-18}$$

where $d_1 = \dfrac{\log\left(\frac{\Sigma_t}{K}\right) + (\widetilde{\mu} + \frac{1}{2}\widetilde{\sigma}^2)(T-t)}{\widetilde{\sigma}\sqrt{(T-t)}}$, $d_2 = d_1 - \widetilde{\sigma}\sqrt{(T-t)}$ and $N(\cdot)$ is the cumulative distribution function of the Gaussian law N(0,1),

$$N(d_1) = \int_{-\infty}^{d_1} \frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}\,du$$

and we have,

$$\frac{\partial N(d_1)}{\partial d_1} = \frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}$$

$$\frac{\partial N(d_2)}{\partial d_2} = \frac{1}{\sqrt{2\pi}}e^{-\frac{d_2^2}{2}} = \frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\frac{\Sigma_t}{K}e^{r(T-t)}$$

B.3.1 Delta $\widetilde{\Delta}$ hedging for the option price $\widetilde{V}$

The delta $\widetilde{\Delta}_t$ of the option price $\widetilde{V}$ is defined by

$$\widetilde{\Delta}_t = \frac{\partial \widetilde{V}(\Sigma_t, t)}{\partial \Sigma_t} \tag{B-19}$$

Remember that d_1 is a function of Σ_t and we have

$$\frac{\partial d_1}{\partial \Sigma_t} = \frac{\partial d_2}{\partial \Sigma_t} = \frac{1}{\Sigma_t \widetilde{\sigma} \sqrt{T-t}}$$

Hence, a classical computation gives,

$$\widetilde{\Delta}_t = N(d_1) \tag{B-20}$$

B.3.2 Gamma $\widetilde{\Gamma}$ hedging for the option price $\widetilde{V}$

The gamma $\widetilde{\Gamma}_t$ of the option price $\widetilde{V}$ is defined by

$$\widetilde{\Gamma}_t = \frac{d^2 \widetilde{V}(\Sigma_t, t)}{\partial \Sigma_t^2} = \frac{\partial \widetilde{\Delta}}{d \Sigma_t} \tag{B-21}$$

Hence, we get

$$\widetilde{\Gamma}_t = \frac{\partial N(d_1)}{\partial d_1} \frac{\partial d_1}{\partial \Sigma_t} = \frac{e^{-\frac{d_1^2}{2}}}{\sqrt{2\pi}} \frac{1}{\Sigma_t \widetilde{\sigma} \sqrt{T-t}} \tag{B-22}$$

B.3.3 Theta $\widetilde{\Theta}$ hedging for the option price $\widetilde{V}$

Denote $\tau = T - t$ time to maturity. The theta $\widetilde{\Theta}_t$ of the option price $\widetilde{V}$ is defined by

$$\widetilde{\Theta}_t = -\frac{\partial \widetilde{V}(\Sigma_t, t)}{\partial \tau} \tag{B-23}$$

Hence,

$$\widetilde{\Theta}_t = -\Sigma_t \frac{\sigma}{2\sqrt{\pi}} N(d_1) - rKe^{-r\tau} N(d_2) \tag{B-24}$$

B.3.4 Rho $\widetilde{\rho}$ hedging for the option price $\widetilde{V}$

The rho $\widetilde{\rho}_t$ of the option price $\widetilde{V}$ is defined by

$$\widetilde{\rho}_t = \frac{\partial \widetilde{V}(\Sigma_t, t)}{\partial r} \tag{B-25}$$

Hence,

$$\widetilde{\rho}_t = (T - t)Ke^{-r(T-t)} N(d_2) \tag{B-26}$$

B.4   Option price formula for basket option based on the reduction technique

We aim to compute the price $V(S_0, 0)$ of a call European Geometric Average option on the basket of d assets, where S_0 implies the basket of assets price at initial time of the contract. The pseudo–code of such option pricing uisng MC methods was described in Algorithm 1. However, this call option price $V(S_0, 0)$ with payoff function $\Phi(x) = (x - K)^+$ is also given by $\widetilde{V}$ in (B-18), where $\Sigma_t = f(S_t) = \prod_{i=1}^d S_t^{i\frac{1}{d}}$.

B.4.1   Delta Δ hedging for the option price V

The vector of Deltas $\left(\Delta_t^i, i = 1, \ldots, d\right)$ is the first order derivative with respect to the change of the underlying asset prices $\left(S_t^1, \ldots, S_t^d\right)$ of the basket option price $V(S_0, 0)$ is computed as follows,

$$\Delta_t^i = \frac{\partial}{\partial S_t^i}\big(V(S_0, 0)\big) = \frac{\partial}{\partial S_t^i}\big(\widetilde{V}(\Sigma_t, t)\big) = \frac{\partial}{\partial S_t^i}\Sigma_t \times \frac{\partial}{\partial \Sigma_t}\widetilde{V}(\Sigma_t, t) = \frac{\partial}{\partial S_t^i}f(S_t) \times \widetilde{\Delta}_t \quad \text{(B-27)}$$

where $\widetilde{\Delta}_t$ is given in (B-20). By replacing $f(S_t)$ in (B-15) we get

$$\prod_{i=1}^d \Delta^i = \left(\frac{1}{d}\widetilde{\Delta}\right)^d \frac{1}{\displaystyle\prod_{i=1}^d S_t^i}\left(\sqrt[d]{\prod_{i=1}^d S_t^i}\right)^d = \left(\frac{1}{d}\widetilde{\Delta}\right)^d. \tag{B-28}$$

B.4.2   Gamma Γ hedging for the option price V

Gamma is the second derivative with respect to the change in the underlying prices $\left(S_t^1, \ldots, S_t^d\right)$ of the basket option price $V(S_0, 0)$. Therefore the matrix of Gamma $\left(\Gamma_t^{ij}, i, j = 1, \ldots, d\right)$ is computed as follows :

$$
\begin{aligned}
\Gamma_t^{ij} &= \frac{\partial^2}{\partial S_t^i S_t^j}\big(V(S_0, 0)\big),\ i, j = 1, \ldots, d \\
&= \frac{\partial}{\partial S_t^j}\left(\frac{\partial}{\partial S_t^i}\Sigma_t \times \frac{\partial}{\partial \Sigma_t}\widetilde{V}(\Sigma_t, t)\right) \\
&= \frac{\partial^2}{\partial S_t^i S_t^j}\Sigma_t\widetilde{\Delta} + \frac{\partial}{\partial S_t^i}\Sigma_t\frac{\partial}{\partial S_t^j}\Sigma_t\frac{\partial^2}{\partial \Sigma_t^2}\widetilde{V}(\Sigma_t, t) \\
&= \frac{\partial^2}{\partial S_t^i S_t^j}\Sigma_t\widetilde{\Delta}_t + \frac{\partial}{\partial S_t^i}\Sigma_t\frac{\partial}{\partial S_t^j}\Sigma_t\widetilde{\Gamma}_t
\end{aligned}
\tag{B-29}
$$

where $\widetilde{\Delta}_t$ and $\widetilde{\Gamma}_t$ are given respectively in (B-20) and (B-22).

B.4.3 Theta Θ hedging for the option price V

$$\Theta_t = \frac{\partial}{\partial \tau}\big(V(S_0,0)\big) = \frac{\partial}{\partial \tau}\widetilde{V}(\Sigma_t,t) = \widetilde{\Theta}_t, \text{ given in (B-24)} \tag{B-30}$$

B.4.4 Rho ρ hedging for the option price V

$$\rho_t = \frac{\partial}{\partial r}\big(V(S_0,0)\big) = \frac{\partial}{\partial r}\widetilde{V}(\Sigma_t,t) = \widetilde{\rho}_t, \text{ given in (B-26)} \tag{B-31}$$

C Calibration of a correlation matrix

C.1 Calibration of the historical correlation matrix

Consider an pool of d asset prices, $\big(S_t^i, i = 1,\ldots,d; t = 1,\ldots,N\big)$. We need to define a correlation matrix $\big(\rho_{ij}, i,j = 1,\ldots,d\big)$. First we compute the return value of an asset S^i over a time scale Δt (e.g. a business day or less if we have enough data),

$$X_i(t) = \log\big(S_{t+\Delta t}^i - S_t^i\big).$$

Here the increment $\big(S_{t+\Delta t}^i - S_t^i\big)$ is supposed to be independent of t. We then define a normalized return as

$$x_i(t) = \frac{X_i(t) - \langle X_i \rangle}{\sigma_i}$$

where $\sigma_i = \sqrt{\langle X_i^2 \rangle - \langle X_i \rangle^2}$ is the standard deviation of X_i and $\langle X_i \rangle = \frac{1}{N-1}\sum_{n=1}^{N-1} X_i(n)$. Then the correlation matrix ρ is constructed as

$$
\begin{aligned}
\rho_{ij} &= \langle x_i(t) x_j(t) \rangle \\
&= \left\langle \left(\frac{X_i(t) - \langle X_i \rangle}{(\langle X_i^2 \rangle - \langle X_i \rangle^2)^{\frac{1}{2}}}\right)\left(\frac{X_j(t) - \langle X_j \rangle}{(\langle X_j^2 \rangle - \langle X_j \rangle^2)^{\frac{1}{2}}}\right) \right\rangle \\
&= \left\langle \left(\frac{X_i(t)X_j(t) - X_j(t)\langle X_i \rangle - X_i(t)\langle X_j \rangle + \langle X_i \rangle \langle X_j \rangle}{(\langle X_i^2 \rangle - \langle X_i \rangle^2)^{\frac{1}{2}}(\langle X_j^2 \rangle - \langle X_j \rangle^2)^{\frac{1}{2}}}\right) \right\rangle \\
&= \frac{\langle X_i X_j \rangle - \langle X_j \rangle \langle X_i \rangle - \langle X_i \rangle \langle X_j \rangle + \langle X_i \rangle \langle X_j \rangle}{(\langle X_i^2 \rangle - \langle X_i \rangle^2)^{\frac{1}{2}}(\langle X_j^2 \rangle - \langle X_j \rangle^2)^{\frac{1}{2}}} \\
&= \frac{\mathbb{E}(X_i X_j) - \mathbb{E}(X_i)\mathbb{E}(X_j)}{\big(\mathbb{E}(X_i^2) - \mathbb{E}(X_i)^2\big)^{\frac{1}{2}}\big(\mathbb{E}(X_j^2) - \mathbb{E}(X_j)^2\big)^{\frac{1}{2}}}
\end{aligned}
\tag{C-32}
$$

with $\mathbb{E}(X_i) \simeq \frac{1}{N-1}\sum_{n=1}^{N-1} X_i(n)$. By this construction, all the coefficients of ρ are restricted to the interval $[-1,1]$. Since the coefficient $\rho_{ij} = \langle x_i(t)x_j(t) \rangle$, in matrix notation, such matrix ρ can be also expressed as

$$\rho = \frac{1}{N}\mathbb{X}\mathbb{X}^t \tag{C-33}$$

where $\mathbb{X}$ is a $d \times N$ matrix with elements $\big(x_{i,n} \equiv x_i(n \times \Delta t); i = 1, \ldots, d; n = 1, \ldots, N\big)$. However, such historical correlation matrix ρ is not always able to be used directly in any financial application (e.g. option pricing, portfolio optimization) if the number of observation N is not very large compared to d. Let us give an example: if we construct a historical correlation matrix of first 250 assets in the S&P500 index list using 254 observations (e.g from 07–Avril–2008 to 07–Avril–2009, it means 1 year data, $N/d = 1.016$), the result is a non positive–definite matrix which can not be directly used in an option pricing application. However, if we increase the number of observations to 505 for the same 250 assets (e.g from 07–Avril–2007 to 07–Avril–2009, it means 2 year data, $N/d = 2.02$), the result is a positive–definite one. Therefore in the first case where we can not increase the number of observations by any reason, we need to re-calibrate the historical correlation matrix in order to make it applicable for the financial applications. We detail such re-calibration problem in the next section.

C.2 Re–calibration of the historical correlation matrix

This section addresses the re-calibration problem for a non positive–definite historical correlation matrix in case we have no chance to increase the number of asset price observations. Some recent studies [9, 12] have applied the Random Matrix Theory (RMT) in such financial re–calibration problem.

C.2.1 Random matrix theory

For a given basket of d assets, the correlation matrix ρ contains $\frac{d(d-1)}{2}$ entries to be computed. Such entries ρ_{ij} were determined in (C-32). The studies in [9, 12] showed that the determination of the correlation coefficients is "noisy" if the number of observation N is not very large compared to d. Hence in this case, one must be very careful when using such correlation matrix in any financial application [11]. In [9, 12] the authors discussed how to distinguish such useful values from the "noise" by using RMT. Consider a random correlation matrix

$$R = \frac{1}{N} A A^t \tag{C-34}$$

where A is a $d \times N$ matrix containing d times series of N random variables with zero mean and unit variance, that are uncorrelated. We denote $P_R(\lambda_R)$ the probability density function of eigenvalue λ_R [9], as

$$P_R(\lambda_R) = \frac{Q}{2\pi\sigma^2} \frac{\sqrt{(\lambda_{R,max} - \lambda_R)(\lambda_R - \lambda_{R,min})}}{\lambda} \tag{C-35}$$

for λ_R within the interval $\big[\lambda_{R,min}, \lambda_{R,max}\big]$, where $Q = \dfrac{N}{d}$, $\sigma^2 = 1$ and these $\lambda_{R,min}, \lambda_{R,max}$ are given by

$$\lambda_{R,min}^{R,max} = \sigma^2 \Big(1 + \frac{1}{Q} \pm 2\sqrt{\frac{1}{Q}}\Big) \tag{C-36}$$

Next we compare the probability density function $P_\rho(\lambda_\rho)$ with $P_R(\lambda_R)$. We only consider the eigenvalues λ_ρ of ρ such that they are out of the interval $\left[\lambda_{R,min}, \lambda_{R,max}\right]$. Such corresponding eigenvectors contain the important information for the financial applications. Hence, we count the number of eigenvalues that are greater than $\lambda_{R,max}$ denoting it n and respectively m for the ones that are smaller than $\lambda_{R,min}$. Based on these informations, we propose in the following section a method which can reconstruct the historical correlation matrix in order to obtain a well defined correlation matrix.

C.2.2 Re–calibration algorithm

We diagonalize the historical correlation matrix ρ such that $\rho = VDV^t$ where V is the matrix that contains the eigenvectors of ρ and D is the diagonal matrix whose diagonal contains the eigenvalues of ρ. We set a small non–negative value to the negative diagonal elements of D. Then we compute the matrix $\overline{D}$ which is the reduction matrix of D by simplifying the eigenvalues of D within the interval $\left[\lambda_{R,min}, \lambda_{R,max}\right]$, as follow

$$\overline{D} = \sum_{j=1}^{n} D^j + \sum_{j=d-m+1}^{d} D^j + diag(\ \underbrace{0,\ldots,0}_{n\ elements}\ , T_D, T_D, \ldots, T_D,\ \underbrace{0,\ldots,0}_{m\ elements}\ ) \qquad \text{(C-37)}$$

where $D^j = diag(\underbrace{0,\ldots,0,\lambda_{\rho,j},0,\ldots,0}_{d\ elements})$ and the constant T_D is the trace of the matrix D
. To be more clear, each D^j is a square matrix of order d, with the elements of vector on the main diagonal. Once having the new diagonal matrix $\overline{D}$ then we reconstruct an approximation of the historical correlation matrix ρ under the new form $\overline{\rho} = V\overline{D}V^\top$. The last step is to re-normalize the coefficients of the matrix $\overline{\rho}$ such that

$$\overline{\rho}_{ij} = \frac{\overline{\rho}_{ij}}{\sqrt{\overline{\rho}_{ii}}\sqrt{\overline{\rho}_{jj}}} \qquad \text{(C-38)}$$

Now, the matrix $\overline{\rho}$ presents a well–defined correlation matrix such that it is a positive–definite matrix with the diagonal equal to 1 and all the coefficients are in the interval of $[-1, 1]$.

References

[1] P. Alexius, B.M. Elahi, F. Hedman, P. Mucci, G. Netzer, and Z.A. Shah. A Black-Box Approach to Performance Analysis of Grid Middleware. *LNCS*, 2008.

[2] D. Baileym, J. Barton, T. Lasinski, and H. Simon. The NAS Parallel Benchmarks. *Technical Report RNR-91-002 Revision 2, NAS Systems Division, NASA Ames Research Center*, August 1991.

[3] F. Baude, D. Caromel, C. Dalmasso, M. Danelutto, V. Getov, L. Henrio, and C. Pérez. GCM: A grid extension to fractal for autonomous distributed components. *Annals of Telecommunications*, 64(1):5–24, 2009.

[4] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[5] S. Bezzine, V. Galtier, S. Vialle, F. Baude, M. Bossy, V.D. Doan, and L. Henrio. A Fault Tolerant and Multi-Paradigm Grid Architecture for Time Constrained Problems. Application to Option Pricing in Finance. *2nd IEEE International Conference on e-Science and Grid Computing*, Netherlands, December 2006.

[6] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637, 1973.

[7] P. Glasserman. *Monte Carlo Methods in Financial Engineering.* Springer, 2004.

[8] J. Hull. *Options, futures, and other derivatives.* Prentice Hall Upper Saddle River, NJ, 2000.

[9] L. Laloux, P. Cizeau, M. Potters, and J.P. Bouchaud. Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*, 3(3):391–398, 2000.

[10] D. Lamberton and B. Lapeyre. *Introduction to stochastic calculus applied to finance.* CRC Press, 1996.

[11] S. Pafka and I. Kondor. Noisy covariance matrices and portfolio optimization. *The European Physical Journal B-Condensed Matter and Complex Systems*, 27(2):277–280, 2002.

[12] V. Plerou, P. Gopikrishnan, B. Rosenow, L.A.N. Amaral, and HE Stanley. A random matrix theory approach to financial cross-correlations. *Physica A: Statistical Mechanics and its Applications*, 287(3-4):374–382, 2000.

[13] G. Tsouloupas and MD Dikaiakos. Gridbench: A tool for benchmarking grids. In *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pages 60–67, 2003.

[14] P. Wilmott et al. *Derivatives: The Theory and Practice of Financial Engineering.* J. Wiley, 1998.

Contents