A financial engineering benchmark for performance analysis of grid middlewares

Viet_Dung Doan[1], Abhijeet Gaikwad[2], Mireille Bossy[1], Françoise Baude[1], and Frédéric Abergel[2]

[1] INRIA Sophia Antipolis – Université de Nice – CNRS - I3S
First.Last@sophia.inria.fr
[2] Laboratoire de Mathématiques Appliquées aux Systèmes – Ecole Centrale de Paris First.Last@ecp.fr

Abstract : Pricing and hedging of higher order derivatives such as multidimensional (up to 100 underlying assets) European and first generation exotic options represent mathematically complex and computationally intensive problems. Grid computing promises to give the capability to handle such intense computations. With several Grid middleware solutions available for gridifying traditional applications, 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. In this paper we present SuperQuant Financial Benchmark Suite to evaluate and quantify the overhead imposed by a Grid middleware on throughput of the system and turnaround times for computation. This approach is a step towards producing a middleware independent, reproducible, comparable, self-sufficient and fair performance analysis of Grid middlewares. 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 and by financial application developers to verify implementation of their financial algorithms. In this paper we explain the motivation and the details of the proposed benchmark suite. As a proof of concept, we utilize the benchmarks in an International Grid Programming contest and demonstrate the result of initial experiments.

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 become a passé.

Grid middleware is what makes Grid computing work and easier to work with. It provided 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 etc. 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[3] and the Globus Alliance[4]. Recently, we have seen an upsurge of academic and commercial middleware providers such as gLite[5], ProActive/GCM Parallel Suite[6], Alchemi .NET Grid computing framework[7], and KAAPI/TakTuk[8] etc. 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 codes, workloads that may represent varied computation and are developed with different programming paradigms. Some examples are STREAM[9], LINPACK[10] and MPI

[3] http://www.cs.wisc.edu/condor/
[4] http://www.globus.org/
[5] http://glite.web.cern.ch/glite/
[6] http://proactive.inria.fr/
[7] http://sourceforge.net/projects/alchemi/
[8] http://kaapi.gforge.inria.fr/
[9] http://www.gridstream.org/
[10] http://www.netlib.org/benchmark/

Benchmarks[11], SPEC[12] and most popular NAS Parallel benchmark[13]. 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 components like middleware. Furthermore, in order to have fair evaluation, thus 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, 14]. In our work, however, for the sake of simplicity we assume the benchmark are run on Grid nodes in isolation. 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 paper 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 middleware solutions, scalability of financial algorithms ...

The rest of this paper 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

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 necessary problem specific software design changes. Hence, in order to formalize efforts to design and evaluate any

[11] http://hcl.ucd.ie/project/mpiblib
[12] http://www.spec.org/mpi2007/press/release.html
[13] http://www.nas.nasa.gov/Resources/Software/npb.html

Grid middleware, in this paper we present SuperQuant financial benchmark suite.

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 paper, 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, (1) an embarrassingly parallel kernel, (2) input/output data and Grid metric descriptors, and (3) 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[14] .

The kernel is based on simple computationally intensive financial problems, pricing and hedging of high dimensional European options, as described below. The definitions of financial terms in this section can be found in common textbooks [8, 15], although reader may find the following information self–explanatory.

European Option Pricing

The Black–Scholes (BS) model describes the evaluation of a basket of assets price through a system of stochastic differential equations (SDEs) [10],

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

- $S = \{S^1, \ldots, S^d\}$ is a basket of d assets.
- r is the constant interest rate for every maturity date and at any time.
- $\delta = \{\delta_1, \ldots, \delta_d\}$ are the constant dividend rates.
- $B = \{B^1, \ldots, B^d\}$ is a correlated d-dimensional Brownian motion (BM).
- $\sigma = \{\sigma_1, \ldots, \sigma_d\}$ is a constant volatility vector.

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 $\Phi(f(S_t), t \in [0, T])$ 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, 15]. According to the Arbitrage Pricing Theory [10], the fair price V for the option contract is given by the following expression: $V(S_0, 0) = \mathbb{E}\big[e^{-rT}\Phi(f(S_T))\big]$. The expectation value is calculated by computing the mean with MC simulation based methods [7]. We have such that

$$V(S_0, 0) = \mathbb{E}\big[e^{-rT}\Phi(f(S_T))\big] = \frac{1}{nbMC}\sum_{j=1}^{nbMC} e^{-rT}\Phi(f(S_T)^j) \tag{2}$$

where $nbMC$ is the number of Monte Carlo simulations. The fundamental theorem of Monte Carlo methods shows that

[14] http://www-sop.inria.fr/oasis/plugtests2008/ProActiveMonteCarloPricingContest.html

$$\lim_{N\to\infty} \mathbb{E}\big[e^{-rT}\Phi(f(S_T))\big] \to V(S_0, 0)$$

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 parallel approach for such option

Algorithm 1 Pricing a call GA option of d assets

Require: S_0^j, d, r, δ_i, σ_i, N_T, A, number of simulations $nbMC$
1: **for** $j = 0$ to $nbMC - 1$ **do**
2: **for** $i = 0$ to $d - 1$ **do**
3: $S_T^i = S_0^i \exp(((r - \delta_i) - \sigma_i^2/2)(T) + \sqrt{T}\sum_{k=1}^{d} a_{ik} Z_0^k)$
4: $f(S_T) = f(S_T) \times S_T^i$
5: **end for**
6: $f(S_T) = \sqrt[d]{f(S_T)}$
7: $C_j = e^{-rT}\big(f(S_T) - K\big)^+$
8: $C_j^2 = \Big(e^{-rT}\big(f(S_T) - K\big)^+\Big)^2$
9: **end for**
10: return $\widehat{C} = \dfrac{C_0 + \cdots + C_{nbMC-1}}{nbMC} \equiv V$

pricing using MC methods can be found in [5].

European Greeks Hedging

The Greeks represent sensitivities of option price with respect to parameters like time remained to maturity, volatility, or interest rate. Usually Greeks are higher order derivatives that are computed using finite difference methods [7]. Since Greeks, are not observed in the real time market but, are information that needs to be computed, their accurate values are important, and much more compute intensive. The detail explanation of Greeks such as Delta (Δ), Gamma (Γ), Rho (ρ) and Theta (θ) can be found in [8, 15]. Consider the GA option pricing above, let us denote $V(S_0, 0)|_{S_0^i \pm \epsilon_S}$ the option prices with the respect to the change of the asset S^i, $V(S_0, 0)|_{r\pm\epsilon_r}$ and $V(S_0, 0)|_{T\pm\epsilon_T}$ the option prices with respect to the change of the interest rate r and of the maturity T. Algorithm 2 below presents the pseudo code for the Greeks hedging by using finite difference methods.

The Composition of the Kernel

The core benchmark kernel consists of a batch of 1000 well calibrated *TestCase*s. Each *TestCase* is a high–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

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

Require: $V(S_0,0)$, $V(S_0,0)|_{S_0^i-\epsilon_S}$, $V(S_0,0)|_{S_0^i+\epsilon_S}$, $V(S_0,0)|_r$ and $V(S_0,0)|_T$
Ensure: Δ, Γ, ρ, θ
1: **for** $i = 0$ to d **do**
2: $\quad \Delta^i = \dfrac{V(S_0,0)|_{S_0^i+\epsilon_S} - V(S_0,0)|_{S_0^i-\epsilon_S}}{2S_0^i\epsilon_S}$
3: $\quad \Gamma^{i,i} = \dfrac{V(S_0,0)|_{S_0^{i,i}+\epsilon_S} - 2V(S_0,0) + V(S_0,0)|_{S_0^{i,i}-\epsilon_S}}{(S_0^{i,i}\epsilon_S)^2}$
4: **end for**
5: $\rho = \dfrac{V(S_0,0)|_{r+\epsilon_r} - V(S_0,0)|_{r-\epsilon_r}}{2r\epsilon_r}$ $\theta = \dfrac{V(S_0,0)|_{T+\epsilon_T} - V(S_0,0)|_{T-\epsilon_T}}{2T\epsilon_T}$
6: return Δ, Γ, ρ, θ

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[16]. In Figure 1 we present the computational time on a single core for each type of option within the benchmark suite. In order

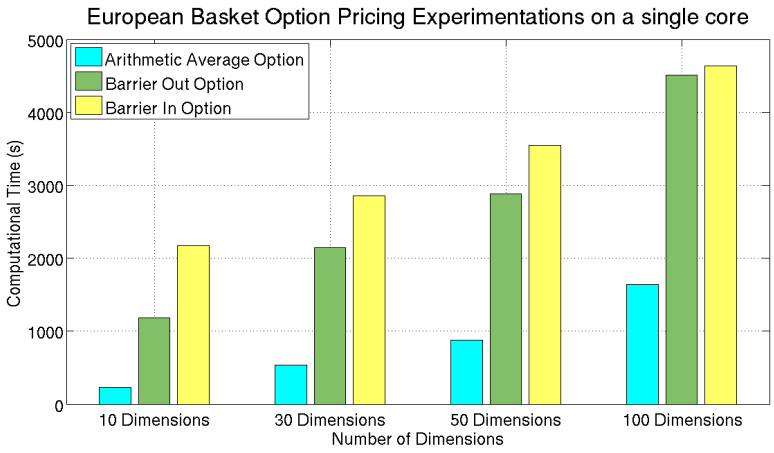

Fig. 1. Computational time of several European basket option pricings on a single core (Intel Xeon(R), E5335, 2.00GHz, 2G RAM)

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

[15] http://www2.standardandpoors.com
[16] http://www.cboe.com/

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

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 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 and etc.
- 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.

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

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

The estimator is strongly consistent, in the sense that

$$\widehat{C} \to C \text{ with probability 1, when } nbMC \to \infty$$

Hence, for a fixed number of MC simulations the Central Limit Theorem gives us the rate of convergence. We compute the estimator for the standard deviation as

$$s_C = \sqrt{\Big(\frac{1}{nbMC}\sum_{j=0}^{nbMC-1} C_j^2\Big) - \Big(\frac{1}{nbMC}\sum_{j=0}^{nbMC-1} C_j\Big)^2}.$$

The value $\widehat{C}$ is obtained with a 95% confidence within the following interval $\big[\pm 0.96\frac{s_C}{\sqrt{nbMC}}\big]$. We decide the tolerable error in computing the results is 10^{-3} which produces a number of MC simulations of $\frac{10^6}{s_C}$. 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". These reference results are computed with sufficiently large number of MC simulations (more than 10^6 simulations), in order to achieve lower confidence interval. The **Output Evaluator** employs a point based scheme to grade the results and also provides a detail 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 total time limit, the team will gain 1 point.

"Reference" Results Validation

The "reference" results provided in the benchmark suite are not analytical results and are computed using MC based methods. Pricing or hedging of high–dimensional European options is not possible with a standard analytical

BS formula [8, 7]. This intriguing question of correctness of the "reference" results also diverted us to investigate methods to validate the results computed by simulation.

We observed that in some specific cases we can analytically reduce a basket of assets into a one–dimensional "pseudo" asset, see in Appendix B. The option price on this "pseudo" asset can be computed by using the BS formula. This way we can compare simulated and analytical results. Further details of the reduction techniques are given in Appendix B.1 and B.2. 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 with prices modeled by SDEs (1). The parameters are given as $S_0^i = 100, i = 1, \ldots, 100$, $K = 100$, $r = 0.0$, $\delta_i = 0.0$, $\sigma = 0.2$ and $T = 1$ year. The basket option is simulated by using 10^6 MC simulations by using Algorithm 1. The "pseudo" asset is $\Sigma_t = \prod_{i=1}^{d} S_t^{i\,\frac{1}{d}}$ and it is the solution of the one–dimensional SDE: $d\Sigma_t/\Sigma_t = \left(\tilde{\mu}dt + \tilde{\sigma}dZ_t\right)$ where $\tilde{\mu} = r + \frac{\sigma^2}{2d} - \frac{\sigma^2}{2}, \tilde{\sigma} = \frac{\sigma}{\sqrt{d}}$ and Z_t is a Brownian motion. The parameters of Σ are given as $\Sigma_0 = 100, \tilde{\mu} = 0.0198, \tilde{\sigma} = 0.02$. We are interested in comparing the estimated option price V of d assets with the analytical "pseudo" one $\tilde{V}$ on Σ. We denote the absolute error $\Delta V = |V - \tilde{V}|$, then the relative error is computed as follow $\eta = \frac{\Delta V}{\tilde{V}}$. In Table 1 we present the numerical results. The first column represents the estimated option prices and their 95% confidence interval. The second column gives the analytical option prices. The last two columns show the absolute and relative errors. As it can be observed, the errors are very small. We can

Table 1. Call/Put price of a GA of 100 assets option and of the "pseudo" one

Call Price V (95% CI)	"Pseudo" Call Price $\tilde{V}$	$\Delta V(10^{-4})$	η (%)
0.16815 (0.00104)	0.16777	3.8	0.22
Put Price V (95% CI)	"Pseudo" Put Price $\tilde{V}$	$\Delta V(10^{-4})$	η (%)
2.12868 (0.00331)	2.12855	1.2	0.01

reduce the errors in case of call option pricing by increasing the number of MC simulations. In this example, we also consider the relation between the Delta hedging of both options in Table 2, see more in Appendix B.3 and B.4.

Table 2. Delta value of the Call/Put GA of 100 assets and of the "pseudo" one

Basket Call Delta Δ	"Pseudo" Call Delta $\tilde{\Delta}$
0.00160	0.16030
Basket Put Delta Δ	"Pseudo" Put Delta $\tilde{\Delta}$
-0.00818	-0.82010

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**[17] at INRIA Sophia Antipolis. The details of the contest and the benchmark input data can be found on the Plugtest and Challenge website. Each participant was given an exclusive one hour access 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 file of Grid'5000 resources which were provided during the contest is given in Table 3.

4.1 Challenge Results

Figure 2 presents the final results of the Challenge. The participants primarily

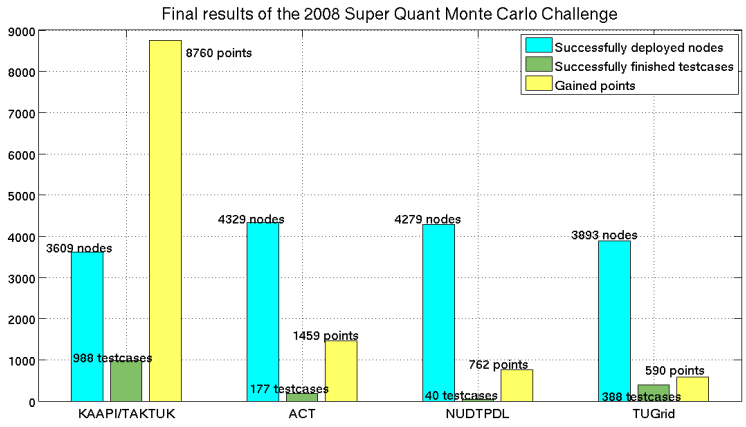

Fig. 2. Final results of the 2008 SuperQuant Monte Carlo challenge

used two middlewares, ProActive, an open source Java based Grid middleware and KAAPI/TAKTUK, which coupled 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 the maximum 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 Grid Component Models

[17] http://www.etsi.org/plugtests/GRID2008/About_GRID.htm
[18] https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home
[19] https://www.intrigger.jp/wiki/index.php/InTrigger

Table 3. Grid'5000 configuration for the "2008 SuperQuant Monte Carlo" challenge

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

(GCM), recently standardized by the ETSI technical committee GRID[20] for deploying the application over large number Grid nodes [3]. The infrastructure descriptors and application descriptors required by GCM 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.

5 Conclusion and Perspectives

In this paper 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.

[20] http://www.etsi.org/WebSite/Technologies/GRID.aspx

APPENDIX

A Simulation of correlated Brownian motions

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

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

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

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

so that $Covariance(B_t^i, B_t^j) = \rho_{ij} t$. 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 $\big(a_{ik}, i, k = 1, \ldots, d\big) = A$, 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

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

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

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

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

A.1 Construction of a correlated d-dimensional Brownian motion with a standard one

Consider a standard q–dimensional Brownian motion $W = \big(W^1, \ldots, W^q\big)$, where each W^i is independent to each other. Consider a matrix $\big(\alpha_{ik}, i = 1, \ldots, d; k = 1, \ldots, q\big)$ be a constant $d \times q$ matrix. We define processes $B^i, i = 1, \ldots, d$ by

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

with

$$\sigma_i = \left[\sum_{k=1}^{q} \alpha_{ik}^2 \right]^{\frac{1}{2}}. \tag{A-7}$$

We are going to show that $\big(B^1, \ldots, B^d\big)$ is a d–dimensional correlated Brownian motion with correlation matrix $\big(\rho_{ij} = \dfrac{1}{\sigma_i \sigma_j} \sum_{k=1}^{q} \alpha_{ik}\alpha_{jk}, i, j = 1, \ldots, d\big)$. 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

$$dB_t^i = \sum_{k=1}^{q} \frac{\alpha_{ik}}{\sigma_i} dW_t^k = \frac{1}{\sigma_i} \sum_{k=1}^{q} \alpha_{ik} dW_t^k \tag{A-8}$$

As W^i a standard Brownian motion, we have

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

Hence, by (A-7) and Itô Lemma

$$\mathbb{E}\big[B_t^i B_t^i\big] = \sum_{k=1}^{q} \frac{\alpha_{ik}^2}{\sigma_i^2} \mathbb{E}\big[W_t^k W_t^k\big] = \frac{1}{\sigma_i^2} \sum_{k=1}^{q} \alpha_{ik}^2 t = t \tag{A-9}$$

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

$$\mathbb{E}\big[B_t^i B_t^j\big] = \sum_{k=1}^{q} \frac{\alpha_{ik}\alpha_{jk}}{\sigma_i \sigma_j} \mathbb{E}\big[W_t^k W_t^k\big] = \frac{1}{\sigma_i \sigma_j} \sum_{k=1}^{q} \alpha_{ik}\alpha_{jk} t \tag{A-10}$$

From (A-9) and (A-10), we proved that $\big(B^1, \ldots, B^d\big)$ 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-8 by changing α with a, we can identify $\big(B_t^1, \ldots, B_t^d\big)$ as $B_t^i = \frac{1}{\sigma_i} \sum_{k=1}^{d} a_{ik} W_t^k$. Replace 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 $(\Omega, \mathcal{F}, (\mathcal{F}_t, t > 0), \mathbb{P})$, 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

$$dS_t^i = S_t^i r dt + S_t^i \sum_{k=1}^{d} a_{ik} dW_t^k, \ i = 1, \dots, d \tag{B-11}$$

where $a_{ik}, i, k = 1, \dots, d$ 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 "pseudo" variable $\Sigma_t = f(S_t)$, where

$$f(x) = \prod_{i=1}^{d} x_i^{\alpha_i}, \forall x = (x_1, ..., x_d) \tag{B-12}$$

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

$$df(S_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial s_i}(S_t) dS_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial s_i \partial s_j}(S_t) S_t^i S_t^j \sum_{k=1}^{d} a_{ik} a_{jk} dt \tag{B-13}$$

With $f(\cdot)$ defined in (B-12), we compute the first term of (B-13),

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

by using the definition of dS_t^i in (B-11), we have

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

We compute the second term of (B-13), we get

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

Hence, by identifying $\Sigma_t = f(S_t) = \prod_{i=1}^{d}(S_t^i)^{\alpha_i}$ then Equation (B-13) becomes:

$$\begin{aligned}\frac{d\Sigma_t}{\Sigma_t} &= \left(\sum_{i=1}^{d}\alpha_i r+\frac{1}{2}\sum_{i,j=1}^{d}\Big(\big(\alpha_i\alpha_j\big)_{i\neq j}+\big(\alpha_i(\alpha_i-1)\big)_{i=j}\Big)\sum_{k=1}^{d}a_{ik}a_{jk}\right)dt\\ &+\sum_{i,k=1}^{d}\alpha_i a_{ik}dW_t^k\end{aligned} \tag{B-14}$$

Considering the process X_t defined by $X_t = \sum_{i,k=1}^{d}\alpha_i a_{ik}W_t^k$. Then Equation (B-14) reduces to

$$\frac{d\Sigma_t}{\Sigma_t} = \underbrace{(r-\widehat{\delta})}_{\widehat{\mu}}dt + dXt \tag{B-15}$$

where

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

could be viewd as the dividend yield by the "pseudo" asset Σ.

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

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

$$f(x) = \prod_{i=1}^{d}x_i^{\frac{1}{d}} \tag{B-16}$$

Start from (B-15)

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

with now

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

and

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

Because W_t^i is a standard Brownian motion, $W_t^i \sim \mathcal{N}(0,t)$, therefore $\sum_{i=1}^d W_t^i \sim \mathcal{N}(0,dt)$. Consider a standard one–dimensional Brownian motion $Z_t \sim \mathcal{N}(0,t)$, we can identify $\sum_{i=1}^d W_t^i = \sqrt{d}Z_t$ and $\sum_{i=1}^d dW_t^i = \sqrt{d}\, dZ_t$ Finally, the equation (B.2) becomes,

$$\frac{d\Sigma_t}{\Sigma_t} = \left(r + \frac{\sigma^2}{2d} - \frac{\sigma^2}{2}\right)dt + \frac{\sigma}{\sqrt{d}}dZ_t \tag{B-17}$$

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 assumed that $\Sigma_t > 0, \forall t$, we have

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

$$\log\Sigma_t = \log\Sigma_0 + \big(\tilde{\mu} - \frac{\tilde{\sigma}^2}{2}\big)t + \tilde{\sigma}B_t$$

this leads to the explicit solution

$$\Sigma_T^{\Sigma_t,t} = \Sigma_t \exp\Big(\big(\tilde{\mu} - \frac{\tilde{\sigma}^2}{2}\big)(T-t) + \tilde{\sigma}B_T\Big), \forall t \in [0,T]. \tag{B-18}$$

B.3 Option price $\tilde{V}$ of the option bases the single asset Σ

Consider a call European option on the asset $\Sigma_t$modeled by Equation (B-18). We can compute such option value by using the Black Schole formula. The call option value at time t is,

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

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

$$\tilde{V}(\Sigma_t, t) = \Sigma_t N(d_1) - K\exp(-r(T-t))N(d_2) \tag{B-19}$$

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

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

and we have,

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

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

Delta $\tilde{\Delta}_t$ hedging for the option price $\tilde{V}$

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

$$\tilde{\Delta}_t = \frac{\partial \tilde{V}(\Sigma_t, t)}{\partial \Sigma_t}$$

Remember that d_1 is a function of Σ_t and we have

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

Hence, a classical computation gives,

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

Gamma $\tilde{\Gamma}_t$ hedging for the option price $\tilde{V}$

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

$$\tilde{\Gamma} = \frac{d^2 \tilde{V}(\Sigma_t, t)}{\partial \Sigma_t^2} = \frac{\partial \tilde{\Delta}}{d\Sigma_t}$$

Hence, we get

$$\tilde{\Gamma}_t = \frac{\partial N(d_1)}{\partial d_1} \frac{\partial d_1}{\partial \Sigma_t} = \frac{e^{-\frac{{d_1}^2}{2}}}{\sqrt{2\pi}} \frac{1}{\Sigma_t \tilde{\sigma}\sqrt{T-t}} \tag{B-21}$$

Theta $\tilde{\Theta}_t$ hedging for the option price $\tilde{V}$

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

$$\tilde{\Theta} = -\frac{\partial \tilde{V}(\Sigma_t, t)}{\partial \tau}$$

Hence,

$$\tilde{\Theta}_t = -\Sigma_t \frac{\sigma}{2\sqrt{\pi}} N(d_1) - rKe^{-r\tau} N(d_2) \tag{B-22}$$

Rho $\tilde{\rho}_t$ hedging for the option price $\tilde{V}$

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

$$\tilde{\rho}_t = \frac{\partial \tilde{V}(\Sigma_t, t)}{\partial r}$$

Hence,

$$\tilde{\rho}_t = (T-t)Ke^{-r(T-t)} N(d_2) \tag{B-23}$$

B.4 Option price V of the option bases on the basket of d assets

The price V of the call European option on the basket of d assets with payoff function $\Phi(x) = (x-K)^+$ is given by $\tilde{V}$ in (B-19) where $\Sigma_t = f(S_t) = \prod_{i=1}^d S_t^{i\,\frac{1}{d}}$.

Delta Δ hedging for the option price V

The vector of Deltas $\left(\Delta_t^i, i=1,\ldots,d\right)$ respects with the basket $\left(S_t^1,\ldots,S_t^d\right)$ of the basket option price $\tilde{V}(f(S_t),t)$ is computed as follow,

$$\Delta_t^i = \frac{\partial}{\partial S_t^i}\big(\tilde{V}(f(S_t),t)\big) = \frac{\partial}{\partial S_t^i} f(S_t) \times \frac{\partial}{\partial \Sigma_t} V(\Sigma_t,t) = \frac{\partial}{\partial S_t^i} f(S_t)\tilde{\Delta} \tag{B-24}$$

Hence, with $f(S_t)$ in (B-16) we get

$$\prod_{i=1}^d \Delta^i = \left(\frac{1}{d}\tilde{\Delta}\right)^d \frac{1}{\prod\limits_{i=1}^d S_t^i}\left(\sqrt[d]{\prod_{i=1}^d S_t^i}\right)^d = \left(\frac{1}{d}\tilde{\Delta}\right)^d \tag{B-25}$$

Gamma Γ hedging for the option price V

Because the Gamma is the second derivative which respects with the basket $\left(S_t^1,\ldots,S_t^d\right)$ of the basket option price $\tilde{V}(f(S_t),t)$. Therefore the matrix of Gamma $\left(\Gamma_t^{ij}, i,j=1,\ldots,d\right)$ is computed as follow,

$$\begin{aligned}
\Gamma_t^{ij} &= \frac{\partial^2}{\partial S_t^i S_t^j}\big(\tilde{V}(f(S_t),t)\big),\ i,j=1,\ldots,d \\
&= \frac{\partial}{\partial S_t^j}\left(\frac{\partial}{\partial S_t^i} f(S_t) \times \frac{\partial}{\partial \Sigma_t} V(\Sigma_t,t)\right) \\
&= \frac{\partial^2}{\partial S_t^i S_t^j} f(S_t)\tilde{\Delta} + \frac{\partial}{\partial S_t^i} f(S_t)\frac{\partial}{\partial S_t^j} f(S_t)\frac{\partial^2}{\partial {\Sigma_t}^2} V(\Sigma_t,t) \\
&= \frac{\partial^2}{\partial S_t^i S_t^j} f(S_t)\tilde{\Delta}_t + \frac{\partial}{\partial S_t^i} f(S_t)\frac{\partial}{\partial S_t^j} f(S_t)\tilde{\Gamma}_t
\end{aligned} \tag{B-26}$$

Theta Θ hedging for the option price V

$$\Theta_t = \frac{\partial}{\partial \tau}\tilde{V}(\Sigma_t,t) = \tilde{\Theta}_t, \text{ given in (B-22)} \tag{B-27}$$

Rho ρ hedging for the option price V

$$\rho_t = \frac{\partial}{\partial r}\tilde{V}(\Sigma_t,t) = \tilde{\rho}_t, \text{ given in (B-23)} \tag{B-28}$$

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. 1 business day),

$$X_i(t) = \log\big(S^i_{t+\Delta t} - S^i_t\big).$$

Here the increment $\big(S^i_{t+\Delta t} - S^i_t\big)$ 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

$$\begin{aligned}
\rho_{ij} &= \langle x_i(t)x_j(t)\rangle \\
&= \left\langle \left(\frac{X_i(t) - \langle X_i\rangle}{(\langle X_i^2\rangle - \langle X_i\rangle^2)^{\frac{1}{2}}}\right)\left(\frac{X_j(t) - \langle X_j\rangle}{(\langle X_j^2\rangle - \langle X_j\rangle^2)^{\frac{1}{2}}}\right)\right\rangle \\
&= \left\langle \left(\frac{X_i(t)X_j(t) - X_j(t)\langle X_i\rangle - X_i(t)\langle X_j\rangle + \langle X_i\rangle\langle X_j\rangle}{(\langle X_i^2\rangle - \langle X_i\rangle^2)^{\frac{1}{2}}(\langle X_j^2\rangle - \langle X_j\rangle^2)^{\frac{1}{2}}}\right)\right\rangle \\
&= \frac{\langle X_iX_j\rangle - \langle X_j\rangle\langle X_i\rangle - \langle X_i\rangle\langle X_j\rangle + \langle X_i\rangle\langle X_j\rangle}{(\langle X_i^2\rangle - \langle X_i\rangle^2)^{\frac{1}{2}}(\langle X_j^2\rangle - \langle X_j\rangle^2)^{\frac{1}{2}}} \\
&= \frac{\mathbb{E}(X_iX_j) - \mathbb{E}(X_i)\mathbb{E}(X_j)}{\big(\mathbb{E}(X_i^2) - \mathbb{E}(X_i)^2\big)^{\frac{1}{2}}\big(\mathbb{E}(X_j^2) - \mathbb{E}(X_j)^2)\big)^{\frac{1}{2}}}
\end{aligned} \tag{C-29}$$

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

$$\rho = \frac{1}{N}\mathbb{X}\mathbb{X}^t \tag{C-30}$$

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 applications (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 results

is a non positive–definite matrix which can not be directly used in 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 and that we can not increase the number of the observations by any reason, we need to re-calibration the historical correlation matrix in order to make it applicable in 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. One popular method among many for such re–calibration problem is using the Random Matrix Theory (RMT). This method was used because it has a long history [6] and many results are known [11]. Furthermore, these results are also interest in a financial context [9, 13].

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-29). Based on the studies in [9, 13], it is 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 applications [12]. In [9, 13] the authors discussed how to distinguish such useful values from the "noise" by using RMT. Consider a random correlation matrix

$$R = \frac{1}{N} A A^t \tag{C-31}$$

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

$$P_R(\lambda_R) = \frac{Q}{2\pi\sigma^2} \frac{\sqrt{(\lambda_{R,max} - \lambda_R)(\lambda_R - \lambda_{R,min})}}{\lambda} \tag{C-32}$$

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

$$\lambda_{R,min}^{R,max} = \sigma^2\Big(1 + \frac{1}{Q} \pm 2\sqrt{\frac{1}{Q}}\Big) \tag{C-33}$$

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 $\lambda_{R,max}$ is n and respectively is m for the ones that are smaller than $\lambda_{R,min}$. Based on these information, we propose a method which can reconstruct the historical correlation matrix in order to obtain a well defined correlation matrix in the following section.

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 that its 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

$$\overline{D} = \sum_{j=1}^{n} D^j + \sum_{j=d-m+1}^{d} D^j + diag(\underbrace{0,\ldots,0}_{n\ elements}, T_D, T_D, \ldots, T_D, \underbrace{0,\ldots,0}_{m\ elements}) \tag{C-34}$$

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

$$\overline{\rho}_{ij} = \frac{\overline{\rho}_{ij}}{\sqrt{\overline{\rho}_{ii}}\sqrt{\overline{\rho}_{jj}}} \tag{C-35}$$

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 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. O. Bohigas and M.J. Giannoni. Mathematical and computational methods in nuclear physics. *Lecture Notes in Physics*, 209, 1983.
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. M.L. Mehta. *Random matrices*. Academic Press, 2004.
12. 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.
13. 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.
14. G. Tsouloupas and MD Dikaiakos. Gridbench: A tool for benchmarking grids. In *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pages 60–67, 2003.
15. P. Wilmott et al. *Derivatives: The Theory and Practice of Financial Engineering*. J. Wiley, 1998.