Spring 5-31-2013

High performance digital signal processing: Theory, design, and applications in finance

Mustafa Ugur Torun
New Jersey Institute of Technology

Recommended Citation

Torun, Mustafa Ugur, "High performance digital signal processing: Theory, design, and applications in finance" (2013). *Dissertations*. 383.
https://digitalcommons.njit.edu/dissertations/383

ABSTRACT

HIGH PERFORMANCE DIGITAL SIGNAL PROCESSING: THEORY, DESIGN, AND APPLICATIONS IN FINANCE

by
Mustafa Ugur Torun

The way scientific research and business is conducted has drastically changed over the last decade. Big data and data-intensive scientific discovery are two terms that have been coined recently. They describe the tremendous amounts of noisy data, created extremely rapidly by various sensing devices and methods that need to be explored for information inference. Researchers and practitioners who can obtain meaningful information out of big data in the shortest time gain a competitive advantage. Hence, there is more need than ever for a variety of high performance computational tools for scientific and business analytics. Interest in developing efficient data processing methods like compression and noise filtering tools enabling real-time analytics of big data is increasing.

A common concern in digital signal processing applications has been the lack of fast handling of observed data. This problem has been an active research topic being addressed by the progress in analytical tools allowing fast processing of big data. One particular tool is the Karhunen-Loève transform (KLT) (also known as the principal component analysis) where covariance matrix of a stochastic process is decomposed into its eigenvectors and eigenvalues as the optimal orthonormal transform. Specifically, eigenanalysis is utilized to determine the KLT basis functions. KLT is a widely employed signal analysis method used in applications including noise filtering of measured data and compression. However, defining KLT basis for a given signal covariance matrix demands prohibitive computational resources in many real-world scenarios.

In this dissertation, engineering implementation of KLT as well as the theory of eigenanalysis for auto-regressive order one, AR(1), discrete stochastic processes are investigated and novel improvements are proposed. The new findings are applied to well-known problems in quantitative finance (QF). First, an efficient method to derive the explicit KLT kernel for AR(1) processes that utilizes a simple root finding method for the transcendental equations is introduced. Performance improvement over a popular numerical eigenanalysis algorithm, called divide and conquer, is shown. Second, implementation of parallel Jacobi algorithm for eigenanalysis on graphics processing units is improved such that the access to the dynamic random access memory is entirely coalesced. The speed is improved by a factor of 68.5 by the proposed method compared to a CPU implementation for a square matrix of size 1,024. Third, several tools developed and implemented in the dissertation are applied to QF problems such as risk analysis and portfolio risk management. In addition, several topics in QF, such as price models, Epps effect, and jump processes are investigated and new insights are suggested from a multi-resolution (multi-rate) signal processing perspective. It is expected to see this dissertation to make contributions in better understanding and bridging the analytical methods in digital signal processing and applied mathematics, and their wider utilization in the finance sector. The emerging joint research and technology development efforts in QF and financial engineering will benefit the investors, bankers, and regulators to build and maintain more robust and fair financial markets in the future.

HIGH PERFORMANCE DIGITAL SIGNAL PROCESSING: THEORY, DESIGN, AND APPLICATIONS IN FINANCE

by
Mustafa Ugur Torun

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

May 2013

APPROVAL PAGE

HIGH PERFORMANCE DIGITAL SIGNAL PROCESSING: THEORY, DESIGN, AND APPLICATIONS IN FINANCE

Mustafa Ugur Torun

Dr. Ali N. Akansu, Dissertation Advisor Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Michael A. Ehrlich, Committee Member Date
Assistant Professor of Finance, NJIT

Dr. Richard A. Haddad, Committee Member Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Sanjeev R. Kulkarni, Committee Member Date
Professor of Electrical Engineering, Princeton University

Dr. Osvaldo Simeone, Committee Member Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Mustafa Ugur Torun
Degree:	Doctor of Philosophy
Date:	May 2013

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2013

- Master of Science in Electrical and Electronics Engineering,
  Dokuz Eylul University, Izmir, Turkey, 2007

- Bachelor of Science in Electrical and Electronics Engineering,
  Dokuz Eylul University, Izmir, Turkey, 2005

Major:	Electrical Engineering

Presentations and Publications:

M. U. Torun and A. N. Akansu, "An Efficient Method to Derive Explicit KLT Kernel for Discrete First Order Auto-Regressive Process," *submitted to IEEE Transactions on Signal Processing*.

A. N. Akansu and M. U. Torun, "Toeplitz approximation to empirical correlation matrix of asset returns: A signal processing perspective," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 4, pp. 319-326, Aug. 2012.

M. U. Torun, A. N. Akansu, and M. Avellaneda, "Portfolio risk in multiple frequencies," *IEEE Signal Processing Magazine, Special Issue on Signal Processing for Financial Applications,* vol. 28, no. 5, pp. 61-71, Sept. 2011.

A. N. Akansu, H. Agirman-Tosun, and M. U. Torun, "Optimal design of phase function in generalized DFT," *Physical Communication, Elsevier,* vol. 3, no. 4, pp. 255-264, Dec. 2010.

M. U. Torun and D. Kuntalp, "Complexity reduction of RBF multiuser detector for DS-CDMA using genetic algorithm," *Turkish Journal of Electrical Engineering and Computer Sciences, to appear,* 2013.

O. Yilmaz, M. U. Torun, and A. N. Akansu, “A Fast Derivation of Karhunen-Loève Transform Kernel for First-Order Autoregressive Discrete Process,” *Big Data Analytics Workshop, ACM Sigmetrics 2013*, Pittsburgh, PA, June 2013, *accepted.*

M. U. Torun and A. N. Akansu “A Novel Method to Derive Explicit KLT Kernel for AR(1) Process,” *IEEE ICASSP 2013*, Vancouver, Canada, May 2013, *accepted.*

M. U. Torun, O. Yilmaz, and A. N. Akansu, “FPGA Based Eigenfiltering for real-time portfolio risk analysis,” *IEEE ICASSP 2013*, Vancouver, Canada, May 2013, *accepted.*

M. U. Torun and A. N. Akansu, “A novel GPU implementation of eigenanalysis for risk management,” *Proc. IEEE SPAWC 2012*, Cesme, Turkey, June 2012.

M. U. Torun, O. Yilmaz, and A. N. Akansu, “Novel GPU implementation of Jacobi algorithm for Karhunen-Loève transform of dense matrices,” *Proc. IEEE CISS 2012*, Princeton, New Jersey, March 2012.

A. N. Akansu and M. U. Torun, “On Toeplitz approximation to empirical correlation matrix of financial asset returns,” *Proc. IEEE CISS 2012*, Princeton, New Jersey, March 2012.

W. P. Weydig, M. U. Torun, and A. N. Akansu, “Implementation of generalized DFT on field programmable gate array,” *Proc. IEEE ICASSP 2012*, Kyoto, Japan, March 2012.

M. U. Torun and A. N. Akansu, “On Epps effect and rebalancing of hedged portfolio in multiple frequencies,” *Proc. IEEE CAMSAP 2011*, San Juan, Puerto Rico, Dec. 2011.

M. U. Torun and A. N. Akansu, “On basic price model and volatility in multiple frequencies,” *Proc. IEEE SSPW 2011*, Nice, France, June 2011.

M. U. Torun, A. N. Akansu, and M. Avellaneda, “Risk management for trading in multiple frequencies,” *Proc. IEEE ICASSP 2011*, Prague, Czech Republic, May 2011.

M. U. Torun and D. Kuntalp, “Genetic algorithm assisted radial basis function multiuser detector for DS-CDMA,” *Proc. IEEE SIU 2007*, Eskisehir, Turkey, June 2007.

M. U. Torun, Y. Isler, D. Kuntalp, and M. Kuntalp, “Classification of branch block beats using higher order spectral analysis and neural networks,” *Proc. IEEE SIU 2006*, Antalya, Turkey, April 2006.

M. U. Torun and A. Ozkurt, “Distance and speed measurement using stereo analysis,” *Proc. IEEE SIU 2006*, Antalya, Turkey, April 2006.

Sevgili annem, babam, ağabeyim,
ve
Biricik Tuba'ma

To my dear mother, father, brother,
and
My one and only Tuba

ACKNOWLEDGMENT

I express my sincere gratitude to my advisor, Prof. Ali N. Akansu, for his endless support, guidance, and encouragement throughout my doctoral study. Words fall short to describe the amount of effort he has put into this work which I deeply appreciate. He taught me how to be innovative, entrepreneurial, thorough, detail- and, result-oriented. I am going to miss our countless hours of discussions over a coffee on exciting research as well as anything related to life in general.

I am also grateful to Prof. Richard A. Haddad and Prof. Osvaldo Simeone of the ECE Department of New Jersey Institute of Technology (NJIT); Prof. Michael A. Ehrlich of the School of Management of NJIT; and Prof. Sanjeev R. Kulkarni of the EE Department of Princeton University for serving on my dissertation committee and for their continued support. I am thankful to Prof. Marco Avellaneda of Courant Institute of Mathematical Sciences of New York University, and his former PhD student, Dr. Stanley Zhang, for their guidance and support at the early days of this interdisciplinary work.

I would like to thank Dokuz Eylül University (DEU) and NJIT for the financial support that made this study possible. I also appreciate the encouragement of Prof. Cüneyt Güzeliş, former dean of the Engineering Faculty of DEU, Prof. Damla Gürkan-Kuntalp, and Prof. Yeşim Zoral of the EE Department of DEU to study at NJIT.

I have always appreciated the companionship of many friends including Dr. Handan Ağırman, Onur Yılmaz, Burçak Özlüdil-Altın, and Ersin Altın that made my years at NJIT far more enjoyable. I am lucky that I have met them and thankful for their support.

The last and the most, I would like to thank my parents, Servet Torun and Ahmet Fikret Torun; my brother, Enver Mehmet Torun; my dear wife, Tuba Kırcı-Torun; and my parents-in-law, Dilşad Kırcı and Mehmet Akif Kırcı; for their endless support, love, and patience. I cannot stress enough how much grateful I am to my wife for always being there, right next to me. Without them, this work would not have been possible.

TABLE OF CONTENTS

Chapter Page

TABLE OF CONTENTS
(Continued)

Chapter Page

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure Page

LIST OF FIGURES
(Continued)

Figure Page

CHAPTER 1

INTRODUCTION

Scientific community is on the verge of a new era in which research is powered by extensive and high performance computational tools required to manipulate and analyze vast amounts of complex data generated or collected through highly complex simulators or ever increasing types and numbers of sensors. This new type of science is a combination of the experimental, theoretical, and computational sciences that have been around for about two millennia, a couple of centuries, and approximately three decades, respectively. It is called as the *data-intensive science* or the "*fourth paradigm* for scientific exploration" [1, 2]. In a similar fashion, business world is transforming such that companies that are able to analyze very large, usually unstructured, data and determine a business trend the fastest have the competitive edge. According to a recent joint research by IBM and MIT [3], number of organizations that use analytics to achieve a competitive advantage has risen sixty percent over a year and they are over two times more likely to outperform their peers within the same industry. There is more need than ever for a variety of high performance computational tools for scientific and business analytics and it is expected that this demand will be higher in the near future.

Big data is a term coined recently and used to describe digital data that has a combination of features such as extremely large in volume, very fast in arrival-rate, unstructured in context thus immensely complex to analyze, and highly noisy such that its credibility is arguable without proper filtering. Those features are also referred to as "The Four Vs", corresponding to volume, velocity, variety, and veracity [4]. IBM estimates that 2.5 quintillion (2.5×10^{18}) bytes of data are created daily and the size of the collective data on Earth increases by a rate of 80% every year. It had grown from 0.8 zettabytes (ZB), i.e., 0.8×10^{21} bytes, to 1.8 ZB from 2009 to 2011 and is expected to be 35 ZB in four years [4]. Hence, demand for better data storage systems and compression algorithms will be much higher in order to handle the high *volume*. Moreover, data hits the wire at an ever increasing *velocity*, making it harder to realize systems that perform real-time analytics. For instance, with the advent of the high frequency trading, today's companies in the financial services industry have to analyze and derive meaningful information out of millions of messages per second with a latency measured in sub-microseconds [4]. Furthermore, data is distributed and stored not only in a large *variety* of formats that are usually unstructured in context, e.g., comments for a post in social-media, that has to be curated [2]; but also along with very high noise, e.g., comments that are actually spam generated by robots, that reduces the *veracity* and credibility of the information it contains. The latter makes it hard for scientists and companies to obtain quality insight from the data. Big data is here and it is expected to be bigger in the future. There is a high demand and interest in developing efficient compression and noise filtering tools that would enable real-time analytics on big data.

In *digital signal processing* (DSP), manipulation and analysis of discrete functions that convey information, i.e., signals, either deterministic or random, are of interest [5, 6]. Signals are manipulated for purposes that include enhancement, denoising, or compression and analyzed such that meaningful information is extracted. Big data is a DSP engineer's both dream and nightmare. The common problem in DSP applications used to be the lack of observed data. Recently, problem has been transformed into creation of fast tools in order to be able to process tremendous amount of data. One particular tool is the *eigenanalysis* in which a square matrix is decomposed into its *eigenvalues* and *eigenvectors*. Eigenanalysis has found its use in DSP as well as in in many other disciplines including but not limited to quantum mechanics, astronomy, geology, marketing, medicine, mechanics, and forensics. Specifically, eigenanalysis is used to determine the basis functions of the *Karhunen-Loève transform* (KLT) [5], also known as the *principal component analysis* (PCA) [7]. Being a signal dependent transform, KLT has been extensively used to filter out the noise as well as to compress signals. Hence, it is a good candidate to solve the problems in the *veracity* category of the big data challenge. However, defining KLT basis for a given signal demands prohibitive computational resources in many cases where the *volume* is large and *velocity* is high [5].

Main focus of this dissertation is on furthering the implementations of KLT as well as eigenanalysis such that they operate much faster, by delving vertically into the theory and design, and apply the findings in to common problems in finance. Specifically, contributions of the dissertation include the following.

1. An efficient method to derive the explicit KLT kernel for *auto-regressive order one*, i.e., AR(1), discrete stochastic processes is proposed by utilizing a relatively new and simple root finding method for the transcendental equations [8].

2. Implementation of *parallel Jacobi algorithm* for eigenanalysis is improved such that the access to the *dynamic random access memory* (DRAM) is entirely *coalesced* providing 68.5 times better performance over traditional methods for a square matrix of size 1,024. The parallel computational device of choice is *general purpose graphics processing unit* (GPGPU or GPU) as it is the mainstream device for parallel computing at the time of writing. However, the concepts proposed can be applied to any computational device that uses DRAM or a similar type of medium for temporary storage.

3. Tools developed in the dissertation are applied to common problems that arise in finance which is by all means the locomotive industry in high performance computing and big data analytics. Namely, applications of the proposed methods into some common problems of *quantitative finance*, i.e., a field that deals with statistics and stochastic calculus to explain and model the pricing of financial assets, and their ties to DSP are discussed.

Further details on above three points are given next. Outline of the dissertation is discussed at the end of the chapter.

1.1 Explicit Karhunen-Loève Transform Kernel for Discrete AR(1) Process

KLT is the optimal block transform in a sense that correlated observations of stationary stochastic signals are transformed into nonstationary and perfectly uncorrelated components (transform coefficients). The coefficient with the highest variance corresponds to the most covariability within the observations, hence the most meaningful information [5]. Therefore, in denoising and compression applications, the coefficients with large variances are kept and the ones with low variances corresponding to noise are discarded [7]. KLT basis functions are the eigenvectors of the covariance matrix of the signal. Hence, it is a signal dependent transform as opposed to other popular transforms like discrete Fourier transform (DFT) and discrete cosine transform (DCT). The fact that DFT and DCT have their explicit kernel that define an orthogonal set regardless of signal statistics and efficient implementations has made them the only feasible option for various engineering applications [5]. Fast implementation of KLT is of great interest to several disciplines, and there were prior attempts reported in the literature to derive closed-form kernel expressions for certain classes of stochastic processes. In particular, such kernels in their implicit forms for continuous-time stochastic processes with exponential auto-correlation function [9, 10, 11, 12] and AR(1) discrete-time stochastic processes [13] are reported in the literature. The one for the discrete AR(1) process is of interest in this dissertation. It is a function of $\{\omega_k\}$ and ρ where $\{\omega_k\}$ $k \geq 0$ are the positive roots of the transcendental equation as given

$$\tan(N\omega) = -\frac{(1-\rho^2)\sin(\omega)}{\cos(\omega) - 2\rho + \rho^2\cos(\omega)}, \tag{1.1}$$

N is the transform size, and ρ is the first-order correlation coefficient of the AR(1) source. Traditionally, roots of (1.1), $\{\omega_k\}$, are found using numerical techniques which in general come with convergence problems. In this dissertation, a simple yet powerful root finding method for the transcendental equations [8] is revisited and an efficient method to derive explicit KLT kernel for AR(1) process is proposed. Moreover, derivation of explicit KLT kernel for an AR(1) process, introduced first in its implicit form in [13] as well as the steps leading to (1.1) are also presented in detail. Furthermore, implementation procedure for the proposed technique is provided in order to highlight its merit.

1.2 Parallel Implementation of Eigenanalysis

Parallel *Jacobi algorithm* (JA) is one of the numerical methods used to approximate the eigenvectors and eigenvalues of a matrix. Although it was introduced in 1846 [14], it did not generate much interest until the last few decades due to its computational load and implementation cost. However, it has an inherent parallelism. Moreover, it was shown that it is a more stable numerical algorithm than the popular QR [15]. Recently, Jacobi algorithm has become more feasible to implement with the dramatic advances in computational hardware. Those advances have not been in the clocking speed but in building massively parallel computational devices with economies of scale. One particular example is the general purpose graphics processing units (GPGPU or GPU) which is considered as a disruptive technology that changed the way researchers and practitioners think about software. At the time of writing, a GPU with computational power in the teraflop (10^{12} of floating point operations) per second range is affordable by many end-users [16]. Parallel devices including GPUs are expected to to get more affordable and powerful in near future.

The limiting factor in parallel devices is the input/output (I/O), specifically the I/O between the on-board memory and the processor. Current commonly used type of memory is DRAM. It is expected to be the mainstream type of memory in the near future due to its cost-effectiveness, i.e., only one capacitor and transistor is enough for one bit. However, DRAM provides its peak performance when neighboring blocks of data is written to or read from it. Therefore, unstructured, i.e., noncoalesced, memory access patterns reduce the performance of parallel applications drastically. In each iteration of JA, a square matrix is updated as

$$\mathbf{A}^{(k+1)} = \mathbf{J}^{\mathrm{T}}(p,q)\mathbf{A}^{(k)}\mathbf{J}(p,q), \tag{1.2}$$

where T is the matrix transpose operator, $\mathbf{A}^{(k)}$ is the approximated eigenvalue matrix at the k*th* iteration with $\mathbf{A}^{(0)}$ being the input matrix, $\mathbf{J}(p, q, \theta)$ is the Jacobi rotation matrix which is different from an identity matrix of same size by only four elements located on the p*th* and q*th* rows and columns with $1 \leq p < q \leq N$, and N is the matrix size. When a traditional data structure, e.g., *row-major array*, is used for the matrices in the update operation given in (1.2), degradation in DRAM performance significantly affects the entire design. In this dissertation, innovative data-structures specifically designed to improve the performance of the implementation of (1.2) on a symmetric matrix are provided. Improvement over traditional data structures is drastic and evaluated to be 68.5 times faster for a square matrix of size 1,024.

1.3 Application of KLT in Portfolio Risk Analysis and Management

Tools developed in the first two parts of the dissertation are applied to common problems that arise in finance, specifically analysis and management of the portfolio risk. A portfolio is comprised of multiple financial assets. The standard deviation of portfolio return is a widely used risk metric in finance as given

$$\sigma_p = \left(\mathbf{q}^{\mathrm{T}}\mathbf{C}\mathbf{q}\right)^{1/2} = \left(\mathbf{q}^{\mathrm{T}}\mathbf{\Sigma}^{\mathrm{T}}\mathbf{P}\mathbf{\Sigma}\mathbf{q}\right)^{1/2}, \tag{1.3}$$

where $\mathbf{q}$ is $N \times 1$ capital allocation vector, $\mathbf{C}$ is the $N \times N$ covariance matrix of the returns of the assets in the portfolio, $\mathbf{\Sigma}$ is the $N \times N$ diagonal matrix comprised of the *volatilities* (standard deviation of returns) of each asset, and $\mathbf{P}$ is the $N \times N$ correlation matrix. $\mathbf{P}$ is also referred to as the *financial correlation matrix* as it contains the cross-correlation of returns of the financial assets. A desirable portfolio delivers maximum return on investment with minimum risk [17]. Therefore, the return of each asset is individually assessed, and also compared against competing assets in the portfolio. Pair-wise correlations of asset returns populate the empirical financial correlation matrix $\hat{\mathbf{P}}$, that reveals significant information on portfolio risk and its variations in time. A portfolio manager *analyzes* these variations and rebalances the portfolio in order to *manage* the risk and keep it within allowed range for the desired return. However, severe nonstationarity with high level of intrinsic noise is common in asset returns. Hence, empirical financial correlation matrix needs to be tamed accordingly. It is a common practice to employ KLT to filter out this undesirable noise component from the measured correlations [18, 19, 20]. Since eigenanalysis is an inherent part of the KLT, in some references this practice is also called as *eigenfiltering*.

In the last part of the dissertation, intrinsic noise in the empirical financial correlation matrix and its eigenfiltering are studied in detail. Moreover, approximation to the empirical financial correlation matrix by a Toeplitz matrix structure and use of DCT as an approximation to KLT are proposed and also their effects on speed and error of the risk analysis are discussed. The motivations include the availability of the explicit kernel for the former and closeness of the DCT and KLT kernels for highly correlated processes for the latter. Finally, a simple and common method to manage the risk is presented and two novel improvements on it are discussed.

It is worth to note that, in the pursuit, several fundamental topics in quantitative finance, such as price models, Epps effect, jump processes are reviewed, explained, and quantified from a DSP engineer's perspective. Moreover, a novel risk analysis metric for an investment strategy in which capital allocation in each asset in the porftolio is rebalanced at different speeds (frequencies) is introduced. Furthermore, risk analysis via eigenfiltering of noise is extended to the case of a *hedged portfolio*. It is expected that this dissertation will contribute to the efforts in increasing the collaboration of researchers in DSP and quantitative finance fields [21, 22]. It is noted that both have strong ties that are mostly unexplored and different names are often used for the same concepts which has rendered interdisciplinary communication inefficient.

1.4 Dissertation Outline

This dissertation is organized as follows. In Chapter 2, mathematical preliminaries required for the discussions in the later chapters are given. Topics discussed in this chapter include stochastic processes and measurements; commonly used models for stochastic processes, i.e., AR, MA, and ARMA models; eigenanalysis to decompose a square matrix in its eigenvalues and corresponding eigenvectors; definition of orthonormal block transforms and their performance metrics; Karhunen-Loève Transform (KLT) as well as its closed-form kernel for the AR(1) discrete-time stochastic processes, and discrete cosine transform (DCT) along with its relation to KLT for highly correlated signals.

In Chapter 3, an efficient method to derive the explicit KLT kernel for AR(1) processes is proposed. The mathematical treatment to arrive at the transcendental tangent equation of concern by analyzing the characteristic values and functions of a continuous-time stochastic process with exponential auto-correlation is presented in Section 3.2. Then, steps that lead to the closed-form expression for the KLT kernel as well as to (1.1) for AR(1) process (initially introduced in [13] without detailed discussion on the derivation) are provided in Section 3.3. An explicit root finding method for transcendental equations introduced by Luck and Stevens [8] in order to address the problem at hand is revisited in in Section 3.4. Next, the proposed method to derive the explicit KLT kernel for an AR(1) process in an efficient way is discussed in detail in Section 3.5. Finally, in Section 3.6, implementation advantages of the proposed method are highlighted. Roots of (1.1) for various matrix sizes, N, are provided in Appendix A. MATLAB™ and C codes for the method proposed in the chapter are given in Appendix B.

In Chapter 4, better data structures proposed to improve the access patterns to the DRAM hence the performance of the Jacobi algorithm on parallel devices, namely GPUs, are discussed. Classical Jacobi algorithm (JA), the notation employed in the chapter, and parallel version of the JA are discussed in detail in Sections 4.1, 4.2, and 4.3, respectively. Single- and multi-threaded central processing unit (CPU) implementations of the

JA are briefly discussed in Section 4.4. Next, GPU implementation of JA is discussed in Section 4.5. Compute unified device architecture (CUDA™) language developed and maintained by NVIDIA®, importance of coalesced memory access in GPU applications, implementation of JA using traditional data structures, and proposed data-structures to improve the performance are discussed in the same section. Chapter is concluded with the comparison in speed between different GPU implementations as well as between CPU and GPU implementations that are reported in Section 4.6.

In Chapter 5, fundamentals of quantitative finance (QF) are given from the perspective of a DSP engineer in order to pave the way for the discussions of Chapter 6. Chapter 5 briefly revisits the basic continuous- and discrete-time price models for stocks, the return process, jumps observed in the return process, cross-correlations of return processes and their use in applications such as portfolio optimization, pairs trading, and hedging. In the last section of the chapter, Section 5.4, Epps effect [23] which states that the cross-correlations of return processes decrease as the sampling frequency is increased, is discussed in detail. Solution to the stochastic differential equation to obtain the price in the continuous-time geometric Brownian price model is discussed in Appendix C.

In Chapter 6, portfolio risk analysis and management is discussed. The main objective in the chapter is to apply the tools developed in the previous chapters into portfolio risk problems. The intrinsic noise in the empirical financial correlation matrix and its filtering via KLT for both traditional and hedged portfolios are discussed in Section 6.1. Then, in Section 6.2, a modification to the risk metric given in (1.3) is proposed in order to be able to assess the risk of a portfolio in where rebalancing in the assets are performed at different frequencies. Next, in Section 6.3, approximation to the empirical financial correlation matrix via Toeplitz matrices and using DCT as an approximation to KLT are proposed in order to speed up the eigenfiltering of the noise in the empirical financial correlation matrix. Effects of those approximations in terms of error are also studied in the same section. Chapter is concluded by discussing a straightforward risk management method and two proposed novel modifications to it in Section 6.4. Performance evaluations via *back-testing* of the risk management methods discussed are given at the end of the same section.

It is noted that there is no specific chapter for computer simulations or experiments as those are embedded in the discussion whenever it is necessary. The summary of the contributions of the dissertation and future work are discussed in the last chapter.

CHAPTER 2

MATHEMATICAL PRELIMINARIES

In this chapter, necessary preliminary mathematical background for the discussions of the later chapters is provided. Fundamentals of discrete auto-regressive one, i.e., AR(1), stochastic process, block transforms, performance metrics of block transforms, kernels of Karhunen-Loève Transform (KLT) and discrete cosine transform (DCT), and their similarity for discrete AR(1) processes for highly correlated signals are discussed. More detail on the discussion can be found in the literature including [5, 6, 24].

2.1 Discrete AR(1) Stochastic Signal Model

Random processes and information sources are mathematically described by a variety of signal models including auto-regressive (AR), moving average (MA), and auto-regressive moving average (ARMA). AR source models, also called all-pole models, have been successfully used in applications including speech processing for decades. First-order AR model, AR(1), is a first approximation to many natural signals like images, and it has been widely employed in various disciplines. AR(1) signal is generated through the first-order regression formula written as [5, 25]

$$x(n) = \rho x(n-1) + \xi(n), \tag{2.1}$$

where $\xi(n)$ is a zero-mean white noise sequence, i.e.,

$$\begin{aligned} E\{\xi(n)\} &= 0 \\ E\left\{\xi(n)\xi(n+k)\right\} &= \sigma_\xi^2 \delta_k, \end{aligned} \tag{2.2}$$

$E\left\{\cdot\right\}$ is the expectation operator, and δ_k is the Kronecker delta function. First-order correlation coefficient, ρ, is real in the range of $-1 < \rho < 1$, and the variance of $x(n)$

is given as follows

$$\sigma_x^2 = \frac{1}{(1-\rho^2)}\sigma_\xi^2. \tag{2.3}$$

Auto-correlation sequence of $x(n)$ is expressed as

$$R_{xx}(k) = E\left\{x(n)x(n+k)\right\} = \sigma_x^2\rho^{|k|}; k = 0, \pm1, \pm2, \ldots. \tag{2.4}$$

The resulting Toeplitz auto-correlation matrix of size $N \times N$ of an AR(1) process is shown as

$$\mathbf{R}_x = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix}. \tag{2.5}$$

It is possible to model any ARMA and MA process by an AR process of infinite order [25]. Hence, approximation can be done with an AR process of sufficiently high order. An example for an ARMA(1,1) process is given on page 112 of [25] which is repeated here for convenience. Transfer function of an ARMA(1,1) process is given as

$$H\left(z\right) = \frac{1+bz^{-1}}{1+az^{-1}}. \tag{2.6}$$

Transfer function given in (2.6) can be modeled as follows

$$H\left(z\right) = \left(1 + \sum_{k=1}^{\infty} c_k z^{-k}\right)^{-1}, \tag{2.7}$$

where c_k is the k*th* coefficient of the AR(∞). It follows from (2.6) and (2.7) that

$$\frac{1+az^{-1}}{1+bz^{-1}} = 1 + \sum_{k=1}^{\infty} c_k z^{-k}. \tag{2.8}$$

Inverse z-transform of (2.8) is given as [25]

$$c_k = \begin{cases} 1 & k = 0 \\ (a-b)\,(-b)^{k-1} & k \geq 1. \end{cases} \tag{2.9}$$

Approximation can be done by choosing the number of coefficients finite, say L, where coefficient c_L is negligible, i.e., $c_L \cong 0$.

2.2 Eigenanalysis

An eigenvalue λ and an eigenvector $\boldsymbol{\phi}$ of size $N \times 1$ of a matrix $\mathbf{A}$ of size $N \times N$ must satisfy the eigenvalue equation as given [5, 12, 26]

$$\mathbf{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}. \tag{2.10}$$

Equality given in (2.10) can be rewritten as

$$\mathbf{A}\boldsymbol{\phi} - \lambda\mathbf{I}\boldsymbol{\phi} = (\mathbf{A} - \lambda\mathbf{I})\,\boldsymbol{\phi} = \mathbf{0}, \tag{2.11}$$

such that $(\mathbf{A} - \lambda\mathbf{I})$ is singular where $\mathbf{0}$ is an $N \times 1$ vector with its elements all equal to zero as given

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}},$$

and $\mathbf{I}$ is the $N \times N$ identity matrix. Namely,

$$|\mathbf{A} - \lambda\mathbf{I}| = 0, \tag{2.12}$$

where $|\cdot|$ is the matrix determinant operator. It is noted that if $\mathbf{A}$ is a real and symmetric matrix, its eigenvectors with different eigenvalues are linearly independent. Hence, determinant given in (2.12) is a polynomial in λ of degree N with N roots and (2.11) has N solutions for $\boldsymbol{\phi}$ that result in the eigenpair set $\{\lambda_k, \boldsymbol{\phi}_k\}$ where $0 \leq k \leq N-1$ and $\{\cdot\}$ denotes a set.

2.3 Block Transforms

A family of linearly independent N orthonormal real discrete-time sequences, $\{\phi_k(n)\}$, on the interval $0 \leq n \leq N-1$ satisfies the inner product relationship [5]

$$\sum_{n=0}^{N-1} \phi_k(n)\phi_l(n) = \delta_{k-l} = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases}. \tag{2.13}$$

Equivalently, the orthonormality can also be expressed on the unit circle of the complex plane, $z = e^{j\omega}; -\pi \leq \omega \leq \pi$, as follows

$$\sum_{n=0}^{N-1} \phi_k(n)\phi_l(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_k(e^{j\omega})\Phi_l(e^{j\omega})d\omega = \delta_{k-l}, \tag{2.14}$$

where $\Phi_k(e^{j\omega})$ is the discrete time Fourier transform (DTFT) of $\phi_k(n)$. In matrix form, $\{\phi_k(n)\}$ are the rows of the transform matrix, and also called basis functions

$$\Phi = [\phi_k(n)] : \ k, n = 0, 1, ..., N-1, \tag{2.15}$$

with the orthonormality property stated as

$$\Phi\Phi^{-1} = \Phi\Phi^{\mathrm{T}} = \mathbf{I}, \tag{2.16}$$

where T indicates transposed version of a matrix or a vector. A stochastic signal vector

$$\mathbf{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^{\mathrm{T}}, \tag{2.17}$$

is mapped into the orthonormal space through forward transform operator

$$\boldsymbol{\theta} = \Phi\mathbf{x}, \tag{2.18}$$

where $\boldsymbol{\theta}$ is transform coefficients vector as given

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_{N-1} \end{bmatrix}^{\mathrm{T}}. \tag{2.19}$$

Similarly, the inverse transform yields the signal vector

$$\mathbf{x} = \Phi^{-1}\boldsymbol{\theta} = \Phi^{\mathrm{T}}\boldsymbol{\theta}. \tag{2.20}$$

Hence, one can derive the correlation matrix of transform coefficients as follows

$$\mathbf{R}_\theta = E\left\{\boldsymbol{\theta}\boldsymbol{\theta}^{\mathrm{T}}\right\} = E\left\{\Phi\mathbf{x}\mathbf{x}^{\mathrm{T}}\Phi^{\mathrm{T}}\right\} = \Phi E\left\{\mathbf{x}\mathbf{x}^{\mathrm{T}}\right\}\Phi^{\mathrm{T}} = \Phi\mathbf{R}_x\Phi^{\mathrm{T}}. \tag{2.21}$$

Furthermore, total energy of the transform coefficients is written as

$$E\left\{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}\right\} = \sum_{k=0}^{N-1} E\left\{\theta_k^2\right\} = \sum_{k=0}^{N-1}\sigma_k^2. \tag{2.22}$$

It follows from (2.16) and (2.18) that

$$E\left\{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}\right\} = E\left\{\mathbf{x}^{\mathrm{T}}\Phi^{\mathrm{T}}\Phi\mathbf{x}\right\} = E\left\{\mathbf{x}^{\mathrm{T}}\mathbf{x}\right\} = \sum_{n=0}^{N-1}\sigma_x^2(n) = N\sigma^2, \tag{2.23}$$

where $\sigma_x^2(n)$ is the variance of the n*th* element of the signal vector given in 2.17 that is equal to σ^2. It follows (2.22) and (2.23) that

$$\sigma^2 = \frac{1}{N}\sum_{k=0}^{N-1}\sigma_k^2. \tag{2.24}$$

Energy preserving property of an orthonormal transform, i.e., the equality between signal variance and the average of transform coefficient variances, is evident from (2.24). It is also noted that the linear transformation of the stationary random vector process $\mathbf{x}$ via (2.18) results in a non-stationary random vector process $\boldsymbol{\theta}$, i.e., $\sigma_k^2 \neq \sigma_l^2$ for $k \neq l$ [5].

2.3.1 Performance Metrics

In practice, it is desired that variances of the transform coefficients decrease as the coefficient index k increases so that the energy is consolidated into as less number of transform coefficients as possible [5]. In other words, it is desired to minimize the energy of the approximation error as defined

$$\mathbf{e}_L = \mathbf{x} - \hat{\mathbf{x}}_L = \sum_{k=0}^{N-1} \theta_k \boldsymbol{\phi}_k - \sum_{k=0}^{L-1} \theta_k \boldsymbol{\phi}_k = \sum_{k=L}^{N-1} \theta_k \boldsymbol{\phi}_k, \tag{2.25}$$

where $0 < L \leq N-1$. There are three commonly used metrics to measure the performance of a given orthonormal transform [5]. The compaction efficiency of a transform, i.e., the ratio of the energy in the first L transform coefficients to the total energy, is defined as

$$\eta_E(L) = 1 - \frac{E\left\{\mathbf{e}_L^{\mathrm{T}} \mathbf{e}_L\right\}}{E\left\{\mathbf{e}_0^{\mathrm{T}} \mathbf{e}_0\right\}} = \frac{\sum_{k=0}^{L-1} \sigma_k^2}{N \sigma_x^2}. \tag{2.26}$$

This is an important metric to assess the efficiency of a transform for the given signal type. The gain of transform coding (TC) over pulse code modulation (PCM) performance of an $N \times N$ unitary transform for a given input correlation is particularly significant and widely utilized in transform coding applications as defined

$$G_{TC}^N = \frac{\frac{1}{N} \sum_{k=0}^{N-1} \sigma_k^2}{\left(\prod_{k=0}^{N-1} \sigma_k^2\right)^{1/N}}. \tag{2.27}$$

Similarly, decorrelation efficiency of a transform is defined as

$$\eta_c = 1 - \frac{\sum_{k=0}^{N-1} \sum_{l=1;l\neq k}^{N-1} \left|\mathbf{R}_\theta(k,l)\right|}{\sum_{k=0}^{N-1} \sum_{l=1;l\neq k}^{N-1} \left|\mathbf{R}_x(k,l)\right|}. \tag{2.28}$$

It is desired to have high compaction efficiency, $\eta_E(L)$, high gain of TC over PCM, G_{TC}^N, and high decorrelation efficiency, η_c, for a given $N \times N$ orthonormal transform. Detailed discussion on the performance metrics for the orthonormal transforms can be found in [5].

2.3.2 Karhunen-Loève Transform

KLT provides optimal geometric mean of coefficient variances with a diagonal correlation matrix, $\mathbf{R}_\theta$ given in (2.21), with best possible repacking of signal vector energy into as few transform coefficients as possible. It is noted that $\eta_c = 1$ for KLT where transform coefficients are perfectly decorrelated (pairwise), and signal energy is optimally packed as measured in (2.26) and (2.27) for the given $\mathbf{R}_x$ of (2.21) and transform size N. KLT minimizes the energy of the approximation error given in (2.25) subject to the orthonormality constraint given in (2.16). Hence, the cost function is defined as [5]

$$J = \sum_{k=L}^{N-1} J_k = E\left\{\mathbf{e}_L^{\mathrm{T}}\mathbf{e}_L\right\} - \sum_{k=L}^{N-1} \lambda_k \left(\boldsymbol{\phi}_k^{\mathrm{T}}\boldsymbol{\phi}_k - 1\right), \tag{2.29}$$

where λ_k is the k*th* Lagrangian multiplier. It follows from (2.18) and (2.25) that (2.29) can be rewritten as

$$J = \sum_{k=L}^{N-1} J_k = \sum_{k=L}^{N-1} \boldsymbol{\phi}_k^{\mathrm{T}}\mathbf{R}_x\boldsymbol{\phi}_k - \sum_{k=L}^{N-1} \lambda_k \left(\boldsymbol{\phi}_k^{\mathrm{T}}\boldsymbol{\phi}_k - 1\right), \tag{2.30}$$

Taking gradient of one of the components of the error J, i.e., J_k, with respect to $\boldsymbol{\phi}_k$ and setting it to zero as follows [5]

$$\nabla J_k = \frac{\partial J_k}{\partial \boldsymbol{\phi}_k} = 2\mathbf{R}_x\boldsymbol{\phi}_k - 2\lambda_k\boldsymbol{\phi}_k = 0, \tag{2.31}$$

yields

$$\mathbf{R}_x\boldsymbol{\phi}_k = \lambda_k\boldsymbol{\phi}_k, \tag{2.32}$$

which implies that $\boldsymbol{\phi}_k$ is one of the eigenvectors of $\mathbf{R}_x$ and λ_k is the corresponding eigenvalue. It is evident from (2.32) that basis set for KLT comprises of the eigenvectors of the auto-correlation matrix of the input, i.e., $\mathbf{R}_x$, and it needs to be recalculated whenever signal statistics change. It follows from (2.32) that

$$\begin{aligned} &\mathbf{R}_x\mathbf{A}_{KLT}^{\mathrm{T}} = \mathbf{A}_{KLT}^{\mathrm{T}}\boldsymbol{\Lambda}, \\ &\mathbf{R}_x = \mathbf{A}_{KLT}^{\mathrm{T}}\Lambda\mathbf{A}_{KLT} = \sum_{k=0}^{N-1} \lambda_k\boldsymbol{\phi}_k\boldsymbol{\phi}_k^{\mathrm{T}}, \end{aligned} \tag{2.33}$$

where $\boldsymbol{\Lambda} = diag\left(\lambda_k\right); k = 0, 1, \ldots, N-1$, and k*th* column of $\mathbf{A}_{KLT}^{\mathrm{T}}$ matrix is the k*th* eigenvector $\boldsymbol{\phi}_k$ of $\mathbf{R}_x$ with the corresponding eigenvalue λ_k. It is noted that $\{\lambda_k = \sigma_k^2\}\ \forall k$, for the given $\mathbf{R}_x$ where σ_k^2 is the variance of the k*th* transform coefficient, θ_k [5].

2.3.3 Karhunen-Loève Transform of Discrete AR(1) Process

The eigenvalues of the auto-correlation matrix for an AR(1) process given in (2.5), $\mathbf{R}_x$, are derived to be in the closed-form [13]

$$\sigma_k^2 = \lambda_k = \frac{1-\rho^2}{1-2\rho\cos(\omega_k)+\rho^2}; 0 \leq k \leq N-1, \tag{2.34}$$

where $\{\omega_k\}$ are the positive roots of the following equation

$$\tan(N\omega) = -\frac{(1-\rho^2)\sin(\omega)}{\cos(\omega)-2\rho+\rho^2\cos(\omega)}, \tag{2.35}$$

and the resulting KLT matrix of size $N \times N$ is expressed in the closed-form kernel as [13]

$$\mathbf{A}_{KLT} = [A(k,n)] = \left(\frac{2}{N+\lambda_k}\right)^{1/2} \sin\left[\omega_k\left(n-\frac{N-1}{2}\right)+\frac{(k+1)\pi}{2}\right] \tag{2.36}$$

where $0 \leq k, n \leq N-1$.

2.3.4 Discrete Cosine Transform

Computing the KLT transform matrix is difficult in practice. Therefore, fixed transforms are preferred in many applications that are concerned with the implementation cost of KLT. In contrast to input dependent KLT, discrete cosine transform (DCT) is a fixed transform that offers efficient implementation algorithms. DCT with efficient implementation is an attractive alternative to KLT particularly for highly correlated processes. DCT matrix of size N is defined as [27]

$$\mathbf{A}_{DCT} = [A(k,n)] = \frac{1}{c_k}\cos\left[\frac{k\pi(2n+1)}{2N}\right], \tag{2.37}$$

where $0 \leq k, n \leq N-1$ and

$$c_k = \begin{cases} \sqrt{N} & k = 0 \\ \sqrt{N/2} & k \neq 0 \end{cases}. \tag{2.38}$$

2.3.5 KLT and DCT of Discrete AR(1) Process in the Limit

It is known that performances of DCT and KLT for highly correlated signals are very close to each other [5]. As $\rho \to 1$, right hand side of (2.35) approaches to zero as its denominator is always non-zero. Therefore,

$$\tan(N\omega) \to 0;\ \rho \to 1, \tag{2.39}$$

or

$$\omega_k = k\pi/N;\ \rho \to 1, \tag{2.40}$$

with the exception of ω_0 which is shown to be approaching to zero as $\rho \to 0$ via small-angle substitution in [28]. Hence (2.40) holds for all k. Substituting (2.40) into (2.36) yields

$$\begin{aligned}[A(k,n)] &= \left(\frac{2}{N+\lambda_k}\right)^{1/2} \sin\left[\frac{k\pi}{N}\left(n - \frac{N-1}{2}\right) + \frac{(k+1)\pi}{2}\right] \\ &= \left(\frac{2}{N+\lambda_k}\right)^{1/2} \cos\left[\frac{k\pi}{2N}(2n+1)\right];\ \rho \to 1.\end{aligned} \tag{2.41}$$

Moreover, as $\rho \to 1$, according to (2.34), all eigenvalues except λ_0 approaches to zero. It is shown that as $\rho \to 1$, $\lambda_0 \to N$ again via small-angle substitution in [28]. Substituting those into (2.41) provides the KLT kernel for an AR(1) source when $\rho \to 1$ as given

$$[A(k,n)] = \begin{cases} \left(\frac{1}{N}\right)^{1/2} & k = 0 \\ \left(\frac{2}{N}\right)^{1/2} \cos\left[\frac{k\pi}{2N}(2n+1)\right] & k \neq 0 \end{cases};\ \rho \to 1. \tag{2.42}$$

It is noted that (2.42) is identical to (2.37) in the limit $\rho \to 1$. This very nature of DCT has made it a popular transform that is successfully employed for decomposition of highly

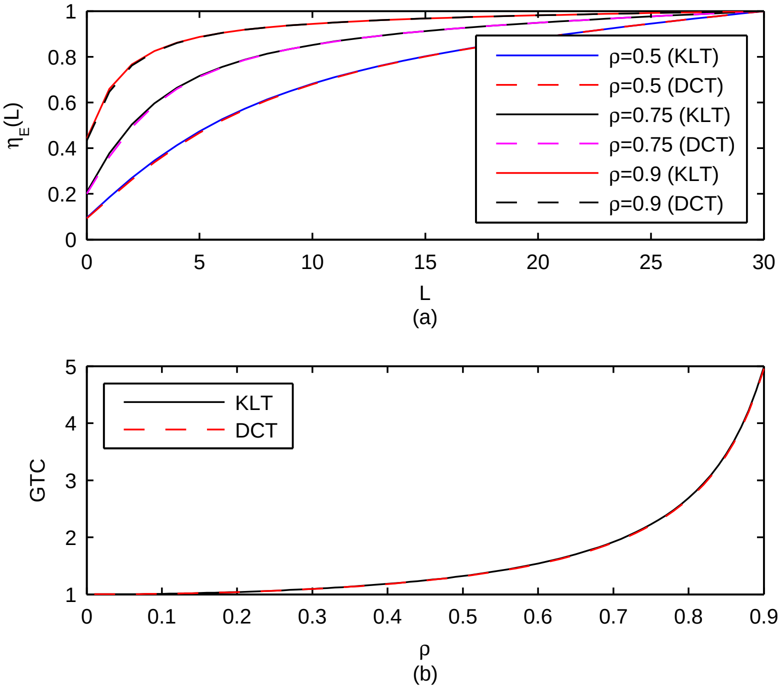


Figure 2.1 (a) $\eta_E(L)$ Performance of KLT and DCT for various values of ρ and $N = 31$, (b) G_{TC}^N performance of KLT and DCT as a function of ρ for $N = 31$.

correlated signal sources. In particular, image and video compression standards like JPEG and MPEG use DCT based 2-D transform coding [5]. In Figure 2.1.a, $\eta_E(L)$ of KLT and DCT as defined in (2.26) are displayed for various values of correlation coefficient ρ and transform size $N = 31$. Similarly, Figure 2.1.b depicts relative G_{TC}^N performance of (2.27) for KLT and DCT as a function of ρ for $N = 31$. This figure verifies the use of DCT as a replacement to KLT in applications where signals are highly correlated. Moreover, it is noted that the energy packing performance of both transforms degrade for lower values of correlation coefficient.

2.4 Chapter Summary

Discrete auto-regressive one stochastic process, i.e., AR(1), is widely used in many engineering applications. Innovations of the process are the samples of a white noise, and process regresses onto itself with a correlation coefficient ρ. It is possible to consolidate the energy of a random process through block transforms into less number of random variables. Orthonormal block transforms preserve the energy, therefore it is possible to generate the original signal back without any loss. Karhunen-Loève Transform (KLT) is the optimal block transform as it perfectly decorrelates the input signal and repacks the energy into as few transform coefficients as possible. Discrete cosine transform (DCT) is a good alternative for KLT for highly correlated signals. DCT provides fast implementations due to its fixed transform matrix whereas in KLT transform matrix has to be calculated through eigenanalysis of the input auto-correlation matrix. Closed-form expression of the KLT kernel for an AR(1) process is known [13]. Moreover, it is shown that as $\rho \to 1$, KLT kernel approaches to DCT kernel.

CHAPTER 3

EXPLICIT KLT KERNEL FOR DISCRETE AR(1) PROCESS

Fast execution of Karhunen-Loève transform (KLT) is desirable as it is the optimal block transform where its basis functions are generated based on a given signal covariance matrix as discussed in Section 2.3. Kernels in their implicit forms for a continuous-time stochastic process with exponential auto-correlation [9, 10, 11, 12] and a discrete-time auto-regressive order one, i.e., AR(1) process [13] are reported in the literature. However, both forms require one to solve transcendental tangent equations. Finding solutions of a transcendental equation has always been of a great interest in various fields [29, 30, 31]. There are a number of numerical methods reported in the literature offering approximate solutions to such equations. Although these techniques are sufficient for many cases, it is quite desirable to have exact explicit solutions for this class of equations leading to analytical treatment of problems at hand as reported in [32, 33]. Most of them are based on the approach of formulating a Riemann problem and finding the solution for the resulting transcendental equation by utilizing a canonical solution of the problem. It was shown that the implementation of this method may easily become quite difficult [33, 34, 35, 36]. Therefore, the topic has been active and several researchers proposed new techniques to improve the efficiency in finding solutions for a transcendental equation.

In this chapter, a simple yet powerful root finding method for transcendental equations [8] is revisited and an efficient method to derive explicit KLT kernel for AR(1) process is proposed. The mathematical steps required for the derivation of explicit KLT kernel for an AR(1) process and the corresponding transcendental equation (introduced first in its implicit form in [13] without detailed derivation) is also presented in the chapter. Furthermore, implementation procedure is detailed and performance evaluations are provided in order to highlight the merit of the method.

3.1 Problem Definition

As it is discussed in detail in Section 2.1, for an AR(1) process with auto-correlation function defined as

$$R_{xx}(k) = E\left\{x(n)x(n+k)\right\} = \rho^{|k|}, \tag{3.1}$$

where $k \in \mathbb{Z}$ and ρ is the first-order auto-correlation coefficient, corresponding KLT matrix $\mathbf{A}_{KLT}$ of size $N \times N$ is expressed with the closed-form kernel as [13]

$$\mathbf{A}_{KLT} = [A(k,n)] = \left(\frac{2}{N+\lambda_k}\right)^{1/2} \sin\left[\omega_k\left(n - \frac{N-1}{2}\right) + \frac{(k+1)\pi}{2}\right], \tag{3.2}$$

where $0 \le k, n \le N-1$. Corresponding transform coefficient variances, i.e., the eigenvalues of the auto-correlation matrix given in (2.5), λ_k, are derived to be in the closed-form [13]

$$\sigma_k^2 = \lambda_k = \frac{1-\rho^2}{1-2\rho\cos(\omega_k)+\rho^2}, \tag{3.3}$$

where $\{\omega_k\}$ are the positive roots of the following transcendental equation [13]

$$\tan(N\omega) = -\frac{(1-\rho^2)\sin(\omega)}{\cos(\omega)-2\rho+\rho^2\cos(\omega)}. \tag{3.4}$$

Steps leading to the equations given in (3.2), (3.3), and (3.4) are not immediately obvious and they need to be clarified. Moreover, there is a need to derive an explicit expression for the roots of the transcendental equation given in (3.4) so that, the kernel and the corresponding variances can also be derived explicitly. Sections in the rest of the chapter deal with these two problems in the same order.

3.2 Eigenanalysis of Continuous-Time Random Process with Exponential Auto-Correlation

In this section, the classic problem of deriving explicit solutions for characteristic values and functions of a continuous random process with exponential auto-correlation function is revisited since it provides the necessary background for the problem at hand, i.e., deriving the explicit KLT kernel for a discrete-time AR(1) process discussed in the next section. This problem is discussed in detail on pages 99-101 of [9], pages 187-190 of [37], and references therein. Similar discussions can also be found in [10, 11].

Characteristic values λ and corresponding characteristic functions $\phi(t)$ of a wide-sense stationary [24] continuous-time random process $x(t)$ with zero mean, i.e., $E\left\{x(t)\right\} = 0$, and exponential auto-correlation function as given

$$R_{xx}(t,s) = E\left\{x(t)x(s)\right\} = e^{-\alpha|t-s|}, \tag{3.5}$$

where $\alpha \in \mathbb{R}$ and $-\infty < t, s < \infty$ satisfy the following integral equation

$$\int_{-T/2}^{T/2} e^{-\alpha|t-s|}\phi(s)ds = \lambda\phi(t). \tag{3.6}$$

Integral equation in (3.6) can be solved by finding a linear differential equation that $\phi(t)$ must satisfy, and then substituting the general solution of the differential equation back in (3.6) to determine the value of λ [9]. In order to drop the magnitude operator, (3.6) can be rewritten as [9]

$$\lambda\phi(t) = \int_{-T/2}^{t} e^{-\alpha(t-s)}\phi(s)ds + \int_{t}^{T/2} e^{-\alpha(s-t)}\phi(s)ds. \tag{3.7}$$

Then, both sides of the equality are differentiated to obtain

$$\lambda\phi'(t) = -\alpha\int_{-T/2}^{t} e^{-\alpha(t-s)}\phi(s)ds + \alpha\int_{t}^{T/2} e^{-\alpha(s-t)}\phi(s)ds, \tag{3.8}$$

where $f'(t)$ is the first-order derivative of $f(t)$. Differentiating one more time and using the Leibniz integral rule given as

$$\frac{\partial}{\partial t}\int_{a(t)}^{b(t)} f\left(s,t\right)ds = \int_{a(t)}^{b(t)} \frac{\partial f\left(s,t\right)}{\partial t}ds + f\left[b\left(t\right),t\right]\frac{\partial b\left(t\right)}{\partial t} - f\left[a\left(t\right),t\right]\frac{\partial a\left(t\right)}{\partial t}, \tag{3.9}$$

results in the following

$$\lambda\phi''(t) = \alpha^2\int_{-T/2}^{T/2} e^{-\alpha|t-s|}\phi(s)ds - 2\alpha\phi(t). \tag{3.10}$$

It follows from (3.6) and (3.10) that

$$\phi''(t) + \frac{\alpha\left(2-\alpha\lambda\right)}{\lambda}\phi(t) = 0. \tag{3.11}$$

The characteristic function, $\phi(t)$ must satisfy the linear homogeneous differential equation of (3.11) in order to satisfy the integral equation given in (3.6). It is shown on pages 99-101 of [9] that (3.11) has solution only in the range of $0 < \lambda < \frac{2}{\alpha}$. Expression given in (3.11) is rewritten as

$$\phi''(t) + b^2\phi(t) = 0, \tag{3.12}$$

where

$$b^2 = \frac{\alpha\left(2-\alpha\lambda\right)}{\lambda},\ 0 < b^2 < \infty. \tag{3.13}$$

General solution to (3.12) is given as

$$\phi(t) = c_1 e^{jbt} + c_2 e^{-jbt}, \tag{3.14}$$

where c_1 and c_2 are arbitrary constants. Substituting (3.14) into (3.7) and solving the integral yields [9, 37]

$$e^{\alpha t}\left(c_1\frac{e^{(-\alpha+jb)T/2}}{\alpha-jb} + c_2\frac{e^{(-\alpha-jb)T/2}}{\alpha+jb}\right) + e^{-\alpha t}\left(c_1\frac{e^{(-\alpha-jb)T/2}}{\alpha+jb} + c_2\frac{e^{(-\alpha+jb)T/2}}{\alpha-jb}\right) = 0. \tag{3.15}$$

For the equality given in (3.15) to hold, following two equalities must hold

$$\begin{aligned} c_1 \frac{e^{(-\alpha+jb)T/2}}{\alpha - jb} &= -c_2 \frac{e^{(-\alpha-jb)T/2}}{\alpha + jb} \\ -c_2 \frac{e^{(-\alpha+jb)T/2}}{\alpha - jb} &= c_1 \frac{e^{(-\alpha-jb)T/2}}{\alpha + jb}. \end{aligned} \tag{3.16}$$

Therefore, solution is possible only when $c_1 = c_2$ or $c_1 = -c_2$. For $c_1 = c_2$, it follows from (3.16) that

$$\frac{-\alpha + jb}{\alpha + jb} = e^{(-\alpha+jb)T/2} e^{-(-\alpha-jb)T/2} = e^{jbT}. \tag{3.17}$$

It follows from (3.17) and the trigonometric identity given as (See (4.4.28) on page 80 of [38])

$$\arctan x = \frac{j}{2} \ln \left(\frac{j + x}{j - x} \right), \tag{3.18}$$

one of the unknowns in the general solution given in (3.14), b, satisfies the following equation

$$b \tan b \frac{T}{2} = \alpha. \tag{3.19}$$

It follows from (3.14) that for every positive b_k that satisfies the transcendental equation in (3.19), there is a characteristic function that satisfies (3.6) as given [9]

$$\phi_k(t) = c_k \cos b_k t, \tag{3.20}$$

where integer $k \geq 0$. Similarly, for $c_1 = -c_2$, b satisfies the following equation

$$b \cot b \frac{T}{2} = -\alpha. \tag{3.21}$$

Again, for every positive b_k that satisfies the transcendental equation in (3.21), there is a characteristic function that satisfies (3.6) as given [9]

$$\phi_k(t) = c_k \sin b_k t. \tag{3.22}$$

It follows from (3.13) that for both cases, corresponding eigenvalues are expressed as [9]

$$\lambda_k = \frac{2\alpha}{\alpha^2 + b_k^2}. \tag{3.23}$$

The roots of transcendental equations given in (3.19) and (3.21) provide the even and odd indexed characteristic values and functions, respectively. These two equations can be combined as a product

$$\left(b \tan b\frac{T}{2} - \alpha\right)\left(b \cot b\frac{T}{2} + \alpha\right) = 0. \tag{3.24}$$

Similarly, for every positive b_k that satisfies the transcendental equation in (3.24), there is a characteristic function that satisfies (3.6) as given

$$\phi_k(t) = c_k \sin\left(b_k t + \frac{(k+1)\,\pi}{2}\right). \tag{3.25}$$

It is noted that (3.23), (3.24), and (3.25) are the continuous analogs of (3.3), (3.4), and (3.2), respectively. Moreover, it is worth noting that the constant, c_k, in (3.25) can be found by normalizing the characteristic functions such that

$$\|\phi_k(t)\|^2 = \int_{-T/2}^{T/2} \phi_k(t)\phi_k^*(t) = 1, \tag{3.26}$$

leading to

$$c_k = \left(\frac{2}{T + \lambda_k}\right)^{1/2}. \tag{3.27}$$

Substituting (3.27) into (3.25) yields

$$\phi_k(t) = \left(\frac{2}{T + \lambda_k}\right)^{1/2} \sin\left(b_k t + \frac{(k+1)\,\pi}{2}\right). \tag{3.28}$$

It is noted that expressions given in (3.2) and (3.28) are analogs of each other.

3.3 Eigenanalysis of Discrete-Time AR(1) Process

In this section, proofs of (3.2), (3.3), and (3.4), that were first reported in [13] without detailed derivations, are given using a similar approach detailed in the previous section for the continuous-time case. For a discrete-time random signal, $x(n)$, discrete Karhunen-Loève (K-L) series expansion is given as

$$\sum_{m=0}^{N-1} R_{xx}(n,m)\phi_k(m) = \lambda_k \phi_k(n), \tag{3.29}$$

where m and n are the independent discrete variables,

$$R_{xx}(n,m) = E\left\{x(n)x(m)\right\},\; m,n = 0,1,\ldots,N-1, \tag{3.30}$$

is the auto-correlation function of the random signal, λ_k is the k*th* eigenvalue, and $\phi_k(n)$ is the corresponding k*th* eigenfunction. Auto-correlation function of the discrete AR(1) process is given as [24]

$$R_x(n,m) = R_x(n-m) = \rho^{|n-m|}. \tag{3.31}$$

Hence, the discrete K-L series expansion for an AR(1) process from (3.29) and (3.31) is stated as follows

$$\sum_{m=0}^{N-1} \rho^{|n-m|}\phi_k(m) = \lambda_k \phi_k(n). \tag{3.32}$$

In order to eliminate the magnitude operator, (3.32) can be rewritten in the form

$$\sum_{m=0}^{n} \rho^{n-m}\phi_k(m) + \sum_{m=n+1}^{N-1} \rho^{m-n}\phi_k(m) = \lambda_k \phi_k(n). \tag{3.33}$$

It follows from the continuous-time case, general solution for the k*th* eigenvector is given as [9, 37]

$$\phi_k(t) = c_1 e^{j\omega_k t} + c_2 e^{-j\omega_k t}, \tag{3.34}$$

where c_1 and c_2 are arbitrary constants, t is the independent continuous-time variable, $-T/2 \leq t \leq T/2$, and $\omega_k = b_k$. Eigenfunction given in (3.34) is shifted by $T/2$ and sampled at $t_n = nT_s$, $0 \leq n \leq N-1$ where $T_s = T/(N-1)$. Accordingly, sampled eigenfunction is written as

$$\phi_k(n) = c_1 e^{j\omega_k(n-\frac{N-1}{2})} + c_2 e^{-j\omega_k(n-\frac{N-1}{2})}. \tag{3.35}$$

As it is noted in the previous section, solution to (3.32) exists only when $c_1 = \pm c_2$. In the following discussions, $c_1 = c_2$ case is considered noting that case for $c_1 = -c_2$ is similar. For $c_1 = c_2$, it follows from (3.35) that

$$\phi_k(n) = c_1 \cos\left[\omega_k\left(n - \frac{N-1}{2}\right)\right]. \tag{3.36}$$

By substituting (3.36) in (3.33) and defining a new independent discrete variable $p = m - (N-1)/2$, (3.32) can be rewritten as follows

$$\sum_{p=-\frac{N-1}{2}}^{n-\frac{N-1}{2}} \rho^{n-p-\frac{N-1}{2}} \cos(\omega_k p) + \sum_{p=n+1-\frac{N-1}{2}}^{\frac{N-1}{2}} \rho^{p+\frac{N-1}{2}-n} \cos(\omega_k p) = \lambda_k \cos\left[\omega_k\left(n - \frac{N-1}{2}\right)\right]. \tag{3.37}$$

The first summation on the left in (3.37) is rewritten as

$$\frac{1}{2}\rho^{n-\frac{N-1}{2}}\left[\sum_{p=-\frac{N-1}{2}}^{n-\frac{N-1}{2}} \left(\rho^{-1}e^{j\omega_k}\right)^p + \sum_{p=-\frac{N-1}{2}}^{n-\frac{N-1}{2}} \left(\rho^{-1}e^{-j\omega_k}\right)^p\right]. \tag{3.38}$$

Using the fact that

$$\sum_{n=N_1}^{N_2} \beta^n = \frac{\beta^{N_1} - \beta^{N_2+1}}{1-\beta}, \tag{3.39}$$

and following simple steps, it can be shown that (3.38), hence the first summation on the left in (3.37), is equal to

$$\frac{\rho^{n+2}\cos\omega_1 - \rho\cos\omega_2 - \rho^{n+1}\cos\omega_3 + \cos\omega_4}{1 - 2\rho\cos\omega_k + \rho^2}. \tag{3.40}$$

Similarly, the second summation on the left in (3.37) is equal to

$$\frac{\rho^{N-n+1}\cos\omega_1 + \rho\cos\omega_2 - \rho^{N-n}\cos\omega_3 - \rho^2\cos\omega_4}{1-2\rho\cos\omega_k+\rho^2}, \tag{3.41}$$

where

$$\begin{aligned} \omega_1 &= \omega_k\left[(N-1)/2\right] \\ \omega_2 &= \omega_k\left[n-(N-1)/2+1\right] \\ \omega_3 &= \omega_k\left[(N-1)/2+1\right] \\ \omega_4 &= \omega_k\left[n-(N-1)/2\right] \end{aligned} \tag{3.42}$$

for both (3.40) and (3.41). It is possible to express λ_k on the right hand side of (3.37) in terms of ρ and ω_k by taking the discrete K-L expansion given in (3.32) into the frequency domain via Fourier transform as follows

$$S_x(e^{j\omega})\Phi_k(e^{j\omega}) = \lambda_k\Phi_k(e^{j\omega}), \tag{3.43}$$

where $S_x(e^{j\omega})$ is the power spectral density of a discrete AR(1) process and expressed as

$$S_x(e^{j\omega}) = \mathcal{F}\left\{\rho^{|n-m|}\right\} = \frac{1-\rho^2}{1-2\rho\cos\omega+\rho^2}, \tag{3.44}$$

$\mathcal{F}\left\{\cdot\right\}$ is the Fourier transform operator [5]. Fourier transform of the eigenfunction in (3.36) is calculated as

$$\begin{aligned} \Phi_k(e^{j\omega}) &= \mathcal{F}\left\{\phi_k(n)\right\} \\ &= c_1 e^{-j\omega_k\frac{N-1}{2}}\left[\delta(\omega-\omega_k)+\delta(\omega+\omega_k)\right], \end{aligned} \tag{3.45}$$

where $\delta(\omega-\omega_0)$ is an impulse function of frequency located at ω_0. By substituting (3.44) and (3.45) into (3.43), λ_k is derived as

$$\lambda_k = \frac{1-\rho^2}{1-2\rho\cos\omega_k+\rho^2}. \tag{3.46}$$

It is noted that (3.46) reads that the eigenvalues are the samples of the power spectral density. Moreover, (3.3) and (3.46) are identical. By substituting (3.40), (3.41), and (3.46) in (3.33), one can show that

$$\rho = \frac{\cos\left(\omega_k N/2 + \omega_k/2\right)}{\cos\left(\omega_k N/2 - \omega_k/2\right)}. \tag{3.47}$$

Using trigonometric identities, the relationship between ω_k and ρ in (3.47) is rewritten as follows

$$\tan\left(\omega_k \frac{N}{2}\right) = \left(\frac{1-\rho}{1+\rho}\right)\cot\left(\frac{\omega_k}{2}\right). \tag{3.48}$$

Similarly, for the case of $c_1 = -c_2$, following the same procedure, the relationship between ω_k and ρ can be shown to be

$$\tan\left(\omega_k \frac{N}{2}\right) = -\left(\frac{1+\rho}{1-\rho}\right)\tan\left(\frac{\omega_k}{2}\right). \tag{3.49}$$

Finally, from (3.48) and (3.49), it is observed that ω_k are the positive roots of the equation

$$\left[\tan\left(\omega \frac{N}{2}\right) + \frac{1+\rho}{1-\rho}\tan\left(\frac{\omega}{2}\right)\right]\left[\tan\left(\omega \frac{N}{2}\right) - \frac{1-\rho}{1+\rho}\cot\left(\frac{\omega}{2}\right)\right] = 0. \tag{3.50}$$

Using trigonometric identities (3.50) can be rewritten as

$$\tan(N\omega) = -\frac{(1-\rho^2)\sin(\omega)}{\cos(\omega) - 2\rho + \rho^2\cos(\omega)}, \tag{3.51}$$

that is the same transcendental equation expressed in (3.4). The roots of the transcendental tangent equation in (3.51), $\{\omega_k\}$, are required in the KLT kernel expressed in (3.2). There are well-known numerical methods like secant method [39] to approximate roots of the equation given in (3.51) in order to solve it implicitly rather than explicitly. A method to find explicit solutions to the roots of transcendental equations, including (3.51), is revisited next. That method leads to an explicit definition of KLT kernel given in (3.2) for an AR(1) process.

3.4 A Simple Method for Explicit Solution of a Transcendental Equation

In this section, a simple method of formulating explicit solution for the roots of transcendental equations using Cauchy's integral theorem from complex analysis [40] that was introduced by Luck and Stevens in [8] is revisited. The method determines the roots of a transcendental function by locating the singularities of a reciprocal function. Although derivation steps are detailed in [8], a summary is given here for the sake of completeness.

Cauchy's theorem states that if a function is analytic in a simple connected region containing the closed curve C, the path integral of the function around the curve C is zero. On the other hand, if a function, $f(z)$, contains a single singularity at z_0 somewhere inside C but analytic elsewhere in the region, then the singularity can be removed by multiplying $f(z)$ with $(z - z_0)$, i.e., by a pole-zero cancellation. Cauchy's theorem implies that the path integral of the new function $(z - z_0)\, f(z)$ around C must be zero

$$\oint_C (z - z_0)\, f(z) dz = 0. \tag{3.52}$$

Evaluation of the integral given in (3.52) yields a first-order polynomial in z_0 with constant coefficients, and its solution for z_0 provides the location of the singularity as given [8]

$$z_0 = \frac{\oint_C z f(z) dz}{\oint_C f(z) dz}. \tag{3.53}$$

This is an explicit expression for the singularity of the function $f(z)$. A root finding problem is restated as a singularity at the root. It is noted that (3.53) gives the location of the desired root and it can be evaluated for any closed path by employing either an analytical or a numerical technique. Luck and Stevens in [8] suggested to use a circle in the complex plane with its center h and radius R as the closed curve C, expressed as

$$\begin{aligned} z &= h + Re^{j\theta}, \\ dz &= jRe^{j\theta} d\theta, \end{aligned} \tag{3.54}$$

where $0 \leq \theta \leq 2\pi$, $h \in \mathbb{R}$, and $R \in \mathbb{R}$. Values of h and R do not matter as long as the circle circumscribes the root z_0. Cauchy's argument principle [41] or graphical methods may be used to determine the number of roots enclosed by the path C. A function in θ is defined as

$$w(\theta) = f(z)|_{z=h+Re^{j\theta}} = f(h + Re^{j\theta}). \tag{3.55}$$

Then (3.53) becomes [8]

$$z_0 = h + R \left[\frac{\int_0^{2\pi} w(\theta)e^{j2\theta}d\theta}{\int_0^{2\pi} w(\theta)e^{j\theta}d\theta} \right]. \tag{3.56}$$

One can easily evaluate (3.56) by employing Fourier analysis since the n*th* Fourier series coefficient for any $x(t)$ is calculated as

$$A_n = \frac{1}{2\pi} \int_0^{2\pi} x(t)e^{jnt}dt. \tag{3.57}$$

It is observed that the term in brackets in (3.56) is equal to the ratio of the second Fourier series coefficient over the first one for the function $w(\theta)$. Fourier series coefficients can be easily calculated numerically by using discrete Fourier transform (DFT) or by using its fast implementation, i.e., fast Fourier transform (FFT) as it is suggested in [8]. However, it is observed from (3.56) that one does not need all DFT (FFT) coefficients to solve the problem since it requires only two Fourier series coefficients. Therefore, it is possible to further improve the computational cost by employing a discrete summation operator to implement (3.56) numerically. Hence, the algorithm would have a computational complexity of $O(N)$ instead of $O\left(NlogN\right)$ required for FFT algorithms.

It is also noted that given $f(z)$ is analytic at h, multiplying $f(z)$ by a factor $(z-h) = Re^{j\theta}$ does not change the location of the singularities of $f(z)$. It means that for a given singularity the term in brackets is also equal to any ratio of the $(\mathrm{m}+1)th$ to the m*th* Fourier series coefficients of $w(\theta)$ for $m \geq 1$ [8].

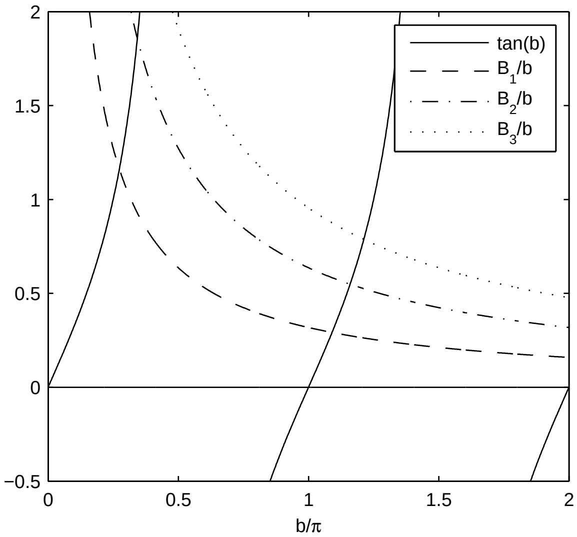


Figure 3.1 Functions $\tan(b)$ and B/b for various values of B where $B_1 = 1$, $B_2 = 2$, and $B_3 = 3$.

3.5 A Simple and Fast Method for the Derivation of Explicit KLT Kernel

In this section, the theory behind the proposed KLT kernel derivation method for discrete AR(1) is highlighted by utilizing prior research on continuous random process with exponential auto-correlation. Moreover, a step-by-step implementation of the novel technique reported herein for the explicit expression of the kernel presented.

3.5.1 Continuous-Time Random Process with Exponential Auto-Correlation

Steps required to determine the roots of (3.19) are studied in this section. It is noted that the discussion is similar for (3.21). It follows from (3.19) that for $\alpha = B$ and $T = 2$ it is possible to write

$$b \tan b = B. \tag{3.58}$$

Positive roots of (3.58), $b_m > 0$, must be calculated in order to determine the even indexed characteristic values and functions given in (3.23) and (3.25), respectively. Figure 3.1 displays functions $\tan(b)$ and B/b for various values of B. It is apparent from the figure that for the m*th* root, a suitable choice for the closed path C is a circle of radius $R = \pi/4$

centered at $h_m = (m - 3/4)\,\pi$ as they are suggested in [8]. A straightforward way to configure the equation given in (3.58) to introduce singularities is to simply use the inverse of rearranged (3.58) as follows [8]

$$f(b) = \frac{1}{b\sin(b) - B\cos(b)}. \tag{3.59}$$

Applying (3.56) to (3.59) results in an explicit expression for the m*th* root. This expression can be evaluated by calculating a pair of adjacently indexed FFT coefficients (coefficients of two adjacent harmonics) as described in Section 3.4 or by using a numerical integration method. Therefore, by setting $b = h + Re^{j\theta}$, $w_m(\theta)$ of (3.55) for this case is defined as

$$\begin{aligned} w_m(\theta) &= f\left(h_m + Re^{j\theta}\right) \\ &= \frac{1}{(h_m + Re^{j\theta})\sin(h_m + Re^{j\theta}) - B\cos(h_m + Re^{j\theta})}, \end{aligned} \tag{3.60}$$

where $0 \leq \theta \leq 2\pi$. Hence, the m*th* root is located at

$$b_m = h_m + R\left[\frac{\int_0^{2\pi} w_m(\theta)e^{j2\theta}d\theta}{\int_0^{2\pi} w_m(\theta)e^{j\theta}d\theta}\right]. \tag{3.61}$$

The MATLAB™ code given in Appendix B.1.1 for calculating the roots of (3.58) shows the simplicity of this method to solve transcendental equations.

3.5.2 Discrete-Time AR(1) Process

In order to derive an explicit expression for the roots of the transcendental equation that are required in the definition of the discrete KLT kernel given in (3.2), first $N/2$ positive roots of two transcendental equations as given

$$\tan\left(\omega\frac{N}{2}\right) = \frac{1}{\gamma}\cot\left(\frac{\omega}{2}\right) \tag{3.62}$$

$$\tan\left(\omega\frac{N}{2}\right) = -\gamma\tan\left(\frac{\omega}{2}\right), \tag{3.63}$$

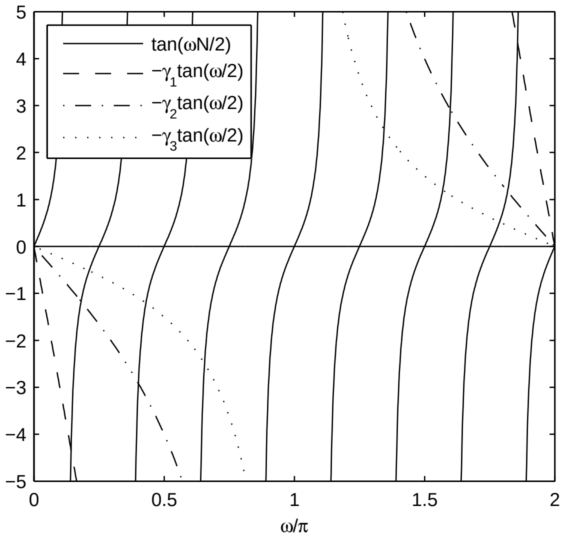


Figure 3.2 Functions $\tan(\omega N/2)$ and $-\gamma \tan(\omega/2)$ for $N = 8$ and various values of ρ where $\rho_1 = 0.9$, $\rho_2 = 0.6$, and $\rho_3 = 0.2$ where $\gamma_i = (1 + \rho_i) / (1 - \rho_i)$, $i = 1, 2, 3$.

must be calculated as discussed in Section 3.3. In both equations, N is the transform size,

$$\gamma = (1 + \rho) / (1 - \rho), \tag{3.64}$$

and ρ is the first order correlation coefficient for AR(1) process. Roots of (3.62) and (3.63) correspond to the even and odd indexed eigenvalues and eigenvectors, respectively. Figure 3.2 displays functions $\tan(\omega N/2)$ and $-\gamma \tan(\omega/2)$ for $N = 8$ and various values of ρ. It is apparent from the figure that for the m*th* root of (3.63), a suitable choice for the closed path C in (3.53) is a circle of radius

$$R_m = \begin{cases} \pi/2N & m \leq 2 \\ \pi/N & m > 2 \end{cases}, \tag{3.65}$$

centered at $h_m = (m - 1/4)(2\pi/N)$ where $1 \leq m \leq N/2$. It is worth to note that for positively correlated signals, i.e., $0 < \rho < 1$, ratio given in (3.64) is always greater than 1, i.e., $\gamma > 1$. However, for negatively correlated signals, i.e., $-1 < \rho < 0,$ ratio is between 0 and 1, i.e., $0 < \gamma < 1$. Therefore, for $\rho < 0$ case, last two roots must be smaller than the rest, i.e.,

$$R_m = \begin{cases} \pi/N & m < N/2 - 1 \\ \pi/2N & m \geq N/2 - 1 \end{cases}. \tag{3.66}$$

Similar to the continuous-time case, (3.63) is reconfigured and poles of the following inverse function are rather looked for

$$g(\omega) = \frac{1}{\tan(\omega N/2) + \gamma \tan(\omega/2)}. \tag{3.67}$$

By setting $\omega = h + Re^{j\theta}$, the function $w(\theta)$ of (3.55) for this case is defined as

$$\begin{aligned} w_m(\theta) &= g\left(h_m + R_m e^{j\theta}\right) \\ &= \frac{1}{\tan\left[\left(h_m + R_m e^{j\theta}\right)\frac{N}{2}\right] + \gamma \tan\left[\left(h_m + R_m e^{j\theta}\right)\frac{1}{2}\right]}, \end{aligned} \tag{3.68}$$

where $0 \leq \theta \leq 2\pi$. Hence, the m*th* root is located at

$$\omega_m = h_m + R_m \left[\frac{\int_0^{2\pi} w_m(\theta) e^{j2\theta} d\theta}{\int_0^{2\pi} w_m(\theta) e^{j\theta} d\theta}\right]. \tag{3.69}$$

The procedure is the same for deriving the roots of (3.62) with the exceptions that (3.68) must be modified as follows

$$w_m(\theta) = \frac{1}{\tan\left[\left(h_m + R_m e^{j\theta}\right)\frac{N}{2}\right] - \frac{1}{\gamma} \cot\left[\left(h_m + R_m e^{j\theta}\right)\frac{1}{2}\right]}, \tag{3.70}$$

and a suitable choice for the closed path C is a circle of radius $R_m = \pi/N$ centered at

$$h_m = \begin{cases} (m - 1/2)(2\pi/N) & m \leq 2 \\ (m - 1)(2\pi/N) & m > 2 \end{cases}, \tag{3.71}$$

that can be determined by generating a plot similar to the ones in Figures 3.1 and 3.2. MATLAB™ code for calculating the roots of (3.62) is given in Appendix B.1.2.

Finally, steps of the proposed novel method to derive an explicit KLT kernel of dimension N expressed in (2.36) for an arbitrary discrete data set by employing an AR(1) approximation are summarized as follows:

1. Estimate the first order correlation coefficient of AR(1) model for the given data set $\{x(n)\}$ as given
$$\rho = \frac{R_{xx}(1)}{R_{xx}(0)} = \frac{E\{x(n)x(n+1)\}}{E\{x(n)x(n)\}}, \tag{3.72}$$
where n is the index of random variables (or discrete-time) and $-1 < \rho < 1$.

2. Calculate the positive roots $\{\omega_k\}$ of the polynomial given in (3.4) by substituting (3.68) and (3.70) into (3.69) for odd and even values of k, respectively, and use the following indexing
$$m = \begin{cases} k/2+1 & k \text{ even} \\ (k+1)/2 & k \text{ odd} \end{cases}. \tag{3.73}$$

3. Plug in the values of ρ and $\{\omega_k\}$ in (3.3) and (3.2) to calculate the eigenvalues λ_k and eigenvectors defining the KLT matrix $\mathbf{A}_{KLT}$, respectively.

MATLAB™ and C codes for steps 2 and 3 with FFT and DFT used in solving (3.69) are provided in Appendix B.2.1 and Appendix B.2.2, respectively.

Remark 1: The computational cost of the proposed method to derive KLT matrix of size $N \times N$ for an arbitrary signal source has two distinct components. Namely, the calculation of the first order correlation coefficient ρ for the given signal set, and the calculation of the roots $\{\omega_k\}$ of (3.4) that are plugged in (3.2) to generate the resulting transform matrix $\mathbf{A}_{KLT}$. The roots $\{\omega_k\}$ of the transcendental tangent equation, calculated by using (3.69), as a function of ρ and for $N = 8$ are displayed in Figure 3.3. Similarly, the values of $\{\omega_k\}$ for $\rho = 0.95$ and various N are provided in Appendix A.

Remark 2: As it is discussed in Section 2.1, other processes like higher order AR, autoregressive moving average (ARMA), and moving average (MA) can also be approximated

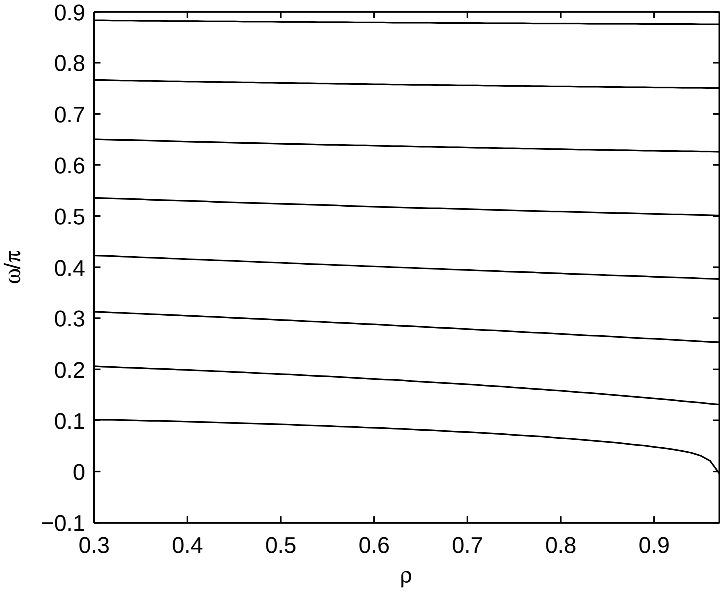


Figure 3.3 The roots of the transcendental tangent equation, $\{\omega_k\}$, as a function of ρ for $N = 8$.

by using an AR process of sufficiently high order [25]. Therefore, the proposed method to drive explicit KLT kernel may also be beneficial for other random processes of interest utilized in various applications.

3.6 Performance Comparison

Herein, the computational cost of generating KLT kernel for the given statistics is studied by employing a widely used numerical algorithm called divide and conquer (D&Q) [26] and the proposed explicit method expressed in (3.2). Moreover, the discrepancy between the kernels generated by the two competing derivation methods is measured. A distance metric between the two kernels is defined as follows

$$d_N = \left\| \mathbf{A}_{KLT,DQ}^{\mathrm{T}} \mathbf{A}_{KLT,DQ} - \mathbf{A}_{KLT,E}^{\mathrm{T}} \mathbf{A}_{KLT,E} \right\|_2 , \tag{3.74}$$

where $\|\cdot\|_2$ is the 2-norm, $\mathbf{A}_{KLT,DQ}$ and $\mathbf{A}_{KLT,E}$ are the $N \times N$ KLT kernels obtained by using D&Q and the proposed explicit derivation method (3.2), respectively. Performance of the proposed method in terms of precision and derivation speed highly depends on the FFT

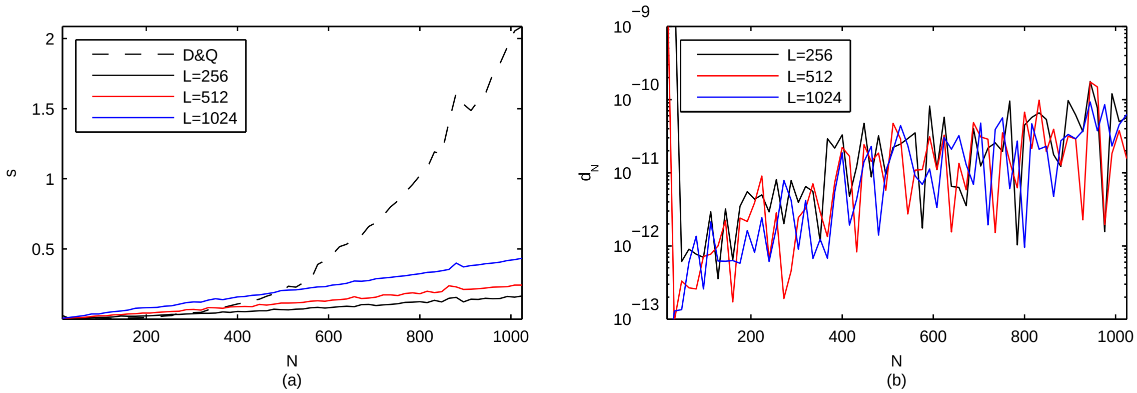


Figure 3.4 (a) Computation time, in seconds, to calculate $\mathbf{A}_{KLT,DQ}$ and $\mathbf{A}_{KLT,E}$ (with $L = 256, 512, 1024$) for $\rho = 0.95$ and $16 \leqslant N \leq 1024$, (b) Corresponding distances, d_N, measured with (3.74) for different values of N and L.

size used in evaluating (3.69). Therefore, the distance metric, d_N, of (3.74) and the time it takes to calculate the kernel by using (3.2) are affected by the FFT size. Computation times (in seconds) to generate $\mathbf{A}_{KLT,DQ}$ and $\mathbf{A}_{KLT,E}$ (FFT sizes of $L = 256,\ 512,\ 1024$) for the case of $\rho = 0.95$ and $16 \leqslant N \leqslant 1024$ are displayed in Figure 3.4.a. Both computations are performed by using one thread on a single processor. The machine used for the simulations has an Intel® Core™ i5-520M CPU and 8 GB of RAM. It is observed from Figure 3.4.a that the proposed method significantly outperforms the D&Q algorithm for larger values of N. Moreover, corresponding distances, d_N, measured with (3.74) for different N and FFT sizes are displayed in Figure 3.4.b. They show that the proposed method is significantly faster than the currently used numerical methods with negligible discrepancy between the two kernels. Furthermore, the proposed KLT kernel derivation algorithm has a so-called *embarrassingly parallel* nature. Hence, it can be easily computed on multiple threads and processors for any k. Therefore, by implementing it on a parallel device such as GPU and FPGA, its speed can be significantly improved.

3.7 Chapter Summary

Closed-form expressions for the $N \times N$ KLT kernel and corresponding transform coefficient variances of the auto-correlation matrix of an AR(1) process with first-order cross-correlation coefficient ρ are reported in the literature [13]. However, they require one to solve a transcendental tangent equation (3.4). Mathematical steps leading to the equations given in (3.2), (3.3), and (3.4) are discussed in detail, following the methodology used for a continuous-time stochastic process with exponential auto-correlation function [9, 10, 11, 12]. Then, a simple and fast method to find the roots of the transcendental equations is employed to derive the roots of (3.4) explicitly. That derivation made it possible to express the $N \times N$ KLT kernel and corresponding transform coefficient variances in explicit form leading to extremely fast KLT implementations for processes that can me modeled with AR(1) process. The merit of the proposed technique is highlighted by performance comparisons with the numerical D&Q algorithm [26]. It is concluded that, since other processes like higher order AR, auto-regressive moving average (ARMA), and moving average (MA) can also be approximated by using AR(1) [25], discussion in this chapter may also be beneficial for other random processes of interest.

CHAPTER 4

IMPROVED NUMERICAL METHODS FOR EIGENANALYSIS

Jacobi algorithm is one of the numerical algorithms used to perform eigenanalysis. It is known that Jacobi is a more stable algorithm then the popularly used QR [15]. However, it approximates to the eigenvalues an eigenvectors iteratively by rotating a pair of rows and columns of the data matrix in each step. Therefore, its implementation on a serial device, e.g., central processing unit (CPU), is not attractive as its time complexity is very high. However, it has inherent parallelism, i.e., the rotations can be done on a parallel grid of processors. Therefore, its implementation on parallel computational devices such as a general purpose graphics processing unit (GPU) [42, 43, 44, 45] has recently gained the interest of many researchers.

In this chapter, GPU implementations of the parallel Jacobi algorithm for the eigenanalysis of real, dense, and symmetric matrices reported in the literature are furthered by improving the data structures and memory access patters for higher performance in terms of speed of the calculation. Significance of memory access patterns on the performance of the implementation of the algorithm on GPUs are emphasized and three novel memory access methods exploiting the symmetry of the input matrix, and availability of the shared memory among GPU threads are proposed. Chapter is concluded with a fair comparison between CPU and GPU implementations in terms of speed.

4.1 Jacobi Algorithm

As it is already discussed in Section 2.2, eigenanalysis of symmetric matrix $\mathbf{A}$ of size $N \times N$ is given as [5]

$$\mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}, \tag{4.1}$$

where $\mathbf{\Lambda}$ is a diagonal matrix comprising of the eigenvalues of $\mathbf{A}$, $\lambda_1, \lambda_2, \ldots, \lambda_N$, $\mathbf{\Phi}$ is an $N \times N$ matrix defined as

$$\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_N \end{bmatrix}, \tag{4.2}$$

and $\boldsymbol{\phi}_i$ is an $N \times 1$ eigenvector corresponding to the *ith* eigenvalue, λ_i. Jacobi algorithm provides an approximated numerical solution to (4.1) by iteratively reducing the metric [26]

$$\text{off}(\mathbf{A}) = \sqrt{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} a_{ij}^2}, \tag{4.3}$$

by multiplying matrix $\mathbf{A}$ from the left and right with Jacobi rotation matrix, $\mathbf{J}(p, q, \theta)$, and overwriting onto itself as expressed

$$\mathbf{A}^{(k+1)} = \mathbf{J}^{\mathrm{T}}(p, q, \theta)\mathbf{A}^{(k)}\mathbf{J}(p, q, \theta), \tag{4.4}$$

where $1 \leq p < q \leq N$ and $k > 0$ is the iteration number. Matrix $\mathbf{J}(p, q, \theta)$ is sparse as defined

$$[J(p, q, \theta)_{ij}] = \begin{cases} \cos\theta & i = p,\ j = p \\ \sin\theta & i = p,\ j = q \\ -\sin\theta & i = q,\ j = p \\ \cos\theta & i = q,\ j = q \\ 1 & i = j,\ i, j \neq p, q \\ 0 & otherwise \end{cases}. \tag{4.5}$$

It is noted that the only difference between the identity matrix $\mathbf{I}$ and $\mathbf{J}(p, q, \theta)$ of the same size is that the elements J_{pp}, J_{pq}, J_{qp}, and J_{qq}. Matrix multiplications in (4.4) are repeated until $\text{off}(\mathbf{A}) < \epsilon$ where ϵ is a predefined threshold. After sufficient number of rotations, matrix $\mathbf{A}$ gets closer to matrix $\mathbf{\Lambda}$, and the successive multiplications leads to an approximation of the eigenvector matrix $\mathbf{\Phi}$ expressed as [26]

$$\mathbf{\Phi} \cong \mathbf{J}(p_1, q_1, \theta_1)\mathbf{J}(p_2, q_2, \theta_2)\cdots\mathbf{J}(p_L, q_L, \theta_L), \tag{4.6}$$

where L is a large integer number. Elements of $\mathbf{J}(p, q, \theta)$, i.e., $c = \cos\theta$ and $s = \sin\theta$, are chosen such a way that the following multiplication yields a diagonal matrix at the k*th* iteration

$$\begin{bmatrix} A_{pp}^{(k+1)} & A_{pq}^{(k+1)} \\ A_{qp}^{(k+1)} & A_{qq}^{(k+1)} \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} A_{pp}^{(k)} & A_{pq}^{(k)} \\ A_{qp}^{(k)} & A_{qq}^{(k)} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \tag{4.7}$$

where $A_{ij}^{(k)}$ and $A_{ij}^{(k+1)}$ are elements of $\mathbf{A}^{(k)}$ and $\mathbf{A}^{(k+1)}$ located on the i*th* row and j*th* column, respectively. It follows from (4.7) that

$$A_{qp}^{(k+1)} = c\left(sA_{pp}^{(k)} + cA_{qp}^{(k)}\right) - s\left(sA_{pq}^{(k)} + cA_{qq}^{(k)}\right). \tag{4.8}$$

Rotation angle θ is found by setting $A_{pq}^{(k+1)} = A_{qp}^{(k+1)} = 0$ in (4.8). Since matrix $\mathbf{A}$ is symmetric, i.e., $A_{pq} = A_{qp}$, it follows from (4.8) that

$$cs\left(A_{pp}^{(k)} - A_{qq}^{(k)}\right) + A_{pq}^{(k)}\left(c^2 - s^2\right) = 0. \tag{4.9}$$

Using trivial trigonometric identities (4.9) can be rewritten as follows

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2A_{pq}^{(k)}}{A_{qq}^{(k)} - A_{pp}^{(k)}}\right). \tag{4.10}$$

It is noted that since p and q define the rotation matrix $\mathbf{J}(p, q, \theta)$ through (4.7), angle can be dropped and the rotation matrix can be referred to as $\mathbf{J}(p, q)$. Originally, p and q in (4.4) are chosen such that $|A_{pq}| = \max_{i \neq j}|A_{ij}|$ [26]. However, searching for the maximum value at each iteration is not preferred due to computational performance considerations. The most straightforward modification to the classical method is the cyclic Jacobi algorithm that serially cycles through the data matrix in an ordered fashion, i.e., $(p^{(m)}, q^{(m)}) = \{(1,2),(1,3),(1,4),\ldots,(2,3),\ldots\}$ where $m = 1, 2, \ldots N/2$. It is straightforward to implement the algorithm on a computing system with $N/2$ parallel processing units due to the sparsity of the rotation matrix $\mathbf{J}(p,q)$ which is discussed in Section 4.3 after the notation employed in the chapter is discussed next.

4.2 Notation

In this section, notation employed in the chapter is discussed in order to make the following discussion clear. For a column vector $\mathbf{z}$ of size $N \times 1$ and a row vector $\overline{\mathbf{z}}$ of size $1 \times N$, it is possible to define matrix $\mathbf{Z}$ of size $N \times N$ as

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \cdots & \mathbf{z}_N \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{z}}_1^{\mathrm{T}} & \cdots & \overline{\mathbf{z}}_N^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{4.11}$$

where $\mathbf{z}_i$ and $\overline{\mathbf{z}}_i$ are the i*th* column and row of $\mathbf{Z}$, respectively. Matrices $\mathbf{Z}^{(m)}$ and $\overline{\mathbf{Z}}^{(m)}$of sizes $N \times 2$ and $2 \times N$, respectively, are defined as follows

$$\begin{aligned} \mathbf{Z}^{(m)} &= \begin{bmatrix} \mathbf{z}_{p^{(m)}} & \mathbf{z}_{q^{(m)}} \end{bmatrix} \\ \overline{\mathbf{Z}}^{(m)} &= \begin{bmatrix} \overline{\mathbf{z}}_{p^{(m)}}^{\mathrm{T}} \\ \overline{\mathbf{z}}_{q^{(m)}}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} Z_{p^{(m)}1} & Z_{p^{(m)}2} & \cdots & Z_{p^{(m)}N} \\ Z_{q^{(m)}1} & Z_{q^{(m)}2} & \cdots & Z_{q^{(m)}N} \end{bmatrix} \end{aligned} \tag{4.12}$$

where $p^{(m)}$ and $q^{(m)}$ are the integers that define the sub-matrix of the data matrix to be rotated in the Jacobi algorithm assigned to the m*th* processing unit. It is noted that when $\mathbf{Z}$ is a symmetric matrix, then $\mathbf{z}_i = \overline{\mathbf{z}}_i^{\mathrm{T}}$ and $\overline{\mathbf{Z}}^{(m)} = \left[\mathbf{Z}^{(m)}\right]^{\mathrm{T}}$. Next, the 2×2 Jacobi sub-rotation matrix used in the m*th* processing unit is defined as shown

$$\mathbf{J}^{(m)} = \begin{bmatrix} J_{p^{(m)}p^{(m)}} & J_{p^{(m)}q^{(m)}} \\ J_{q^{(m)}p^{(m)}} & J_{q^{(m)}q^{(m)}} \end{bmatrix} = \begin{bmatrix} c_m & s_m \\ -s_m & c_m \end{bmatrix}, \tag{4.13}$$

where J_{ij} is the element located at the i*th* row and j*th* column of the original Jacobi matrix defined in (4.5). Since this is an iterative algorithm, one needs to state the iteration index, k, explicitly as expressed in (4.4). However, in order to keep the text clean, assignment operator, $\leftarrow$, is used instead of the iteration index in the rest of the discussion.

4.3 Parallel Jacobi Algorithm

Jacobi rotation matrix given in (4.5) is sparse. Thus, it is possible to perform rotations expressed in (4.4) by a parallel implementation using $N/2$ processing units provided that p and q pairs are unique for each processing unit. For example, two rotations may be implemented in parallel, with $\left(p^{(1)}, q^{(1)}\right) = (1, 2)$ and $\left(p^{(2)}, q^{(2)}\right) = (3, 4)$, for $N = 4$. In the next step, pairs might be selected as $\left(p^{(1)}, q^{(1)}\right) = (1, 4)$ and $\left(p^{(2)}, q^{(2)}\right) = (2, 3)$. It is noted that a scenario with $\left(p^{(1)}, q^{(1)}\right) = (1, 2)$ and $\left(p^{(2)}, q^{(2)}\right) = (2, 4)$ would violate the non-overlap rule since $q^{(1)} = p^{(2)}$, and they must not be run in parallel. There are many possible methods for effectively choosing the pairs for each step [43, 46]. One of the most popular algorithms is called the chess tournament (CT). In CT, for N players, there are $N/2$ pairs and $N - 1$ matches that have to be held such that each player matches against any other player in the group. Once a match set is completed, the first player stands still and every other player moves one seat in clockwise direction. For $N = 4$, the pairs for $N - 1 = 3$ steps are defined as

$$\begin{bmatrix} p^{(1)} & p^{(2)} \\ q^{(1)} & q^{(2)} \end{bmatrix} : \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \tag{4.14}$$

It is noted that $p^{(2)}$ and $q^{(2)}$ are interchanged in the last step since the condition $p < q$ must hold [26]. Moreover, interchanging is necessary after the step $N/2 + 1$. Since $N - 1 = N/2 + 1 = 3$ for $N = 4$, interchange needs to be done only in the last step.

Ensuring that p and q pairs are unique for each processing unit, handles only one of the two problems that arise in the parallel implementation. The second problem corresponds to the overlapping of matrix elements in the operations shown in (4.4). Since the multiplication $\mathbf{J}(p,q)^{\mathrm{T}}\mathbf{A}$ in (4.4) would update the p*th* and q*th* rows of matrix $\mathbf{A}$, and multiplication $\mathbf{AJ}(p,q)$ in (4.4) would update the p*th* and q*th* columns of matrix $\mathbf{A}$, it would be problematic to implement (4.4) in parallel without proper synchronization. Introducing an intermediary matrix $\mathbf{X}$ of size $N \times N$ [42] in the computational process is a popular solution. First operation is performed by the m*th* processing unit by multiplying the $p^{(m)}th$ and $q^{(m)}th$ rows of $\mathbf{A}$ with the transpose of Jacobi sub-rotation matrix, (4.13), as written

$$\overline{\mathbf{X}}^{(m)} \leftarrow \left[\mathbf{J}^{(m)}\right]^{\mathrm{T}} \overline{\mathbf{A}}^{(m)}. \tag{4.15}$$

Waiting for all of the processing units to complete their assigned tasks (a blocking synchronization) is required before proceeding to the next operation that is the multiplication of the $p^{(m)}th$ and $q^{(m)}th$ columns of $\mathbf{X}$ by the Jacobi sub-rotation matrix (4.13), as expressed

$$\mathbf{A}^{(m)} \leftarrow \mathbf{X}^{(m)}\mathbf{J}^{(m)}. \tag{4.16}$$

It is noted that, the eigenvectors may be updated according to the procedure

$$\mathbf{\Phi}^{(m)} \leftarrow \mathbf{\Phi}^{(m)}\mathbf{J}^{(m)}, \tag{4.17}$$

at the same time with (4.16) since $\mathbf{J}^{(m)}$ is already available.

4.4 Single- and Multi-Threaded CPU Implementation of Jacobi Algorithm

Cyclic and parallel Jacobi algorithm with chess tournament are implemented in standard C language and coded to perform on a single thread and multiple threads, respectively. For the multi-threaded implementation POSIX threads library [47] is used. For the multi-threaded implementation, proper synchronization via condition variables is done so that there is no racing condition in the calculation of (4.15), (4.16), and (4.17). At each sweep there two phases: In the first phase, $N/2$ threads are spawned which calculate (4.15) individually. After all the threads are finished, second phase starts. In this phase, another group of

$N/2$ threads are spawned that calculate (4.16) and (4.17). Any calculation that is done in the first phase and can be re-used in the second is kept in the system memory to increase performance.

4.5 GPU Implementation of Parallel Jacobi Algorithm

In this section, graphics processing unit (GPU) implementation of the parallel Jacobi algorithm is detailed. Section starts with a brief discussion about the state-of-the-art GPU computing and CUDA™ programming language. Then, importance of memory coalescing when accessing the global memory of the GPU in the context of parallel Jacobi algorithm is stressed with the help of two methods that are referred to as "traditional access (TA)" and "modified access (MA)." TA is basically the form used in the designs detailed in [42, 43]. Next, three novel access methods are introduced to improve the memory coalescing in the Jacobi algorithm. These methods are named as "symmetrical access (SA)," "maximum coalesced access (MCA)," and "one step parallel Jacobi algorithm (OSPJ)." In OSPJ, discussed last in the section, the need for interim matrix $\mathbf{X}$ used in (4.15) and (4.16) disappears. This feature makes it the best method proposed in terms of speed. Performance evaluations and comparison to CPU implementation are provided in the next section.

4.5.1 GPU Computing and CUDA™

Compute device uniform architecture (CUDA™) introduced by NVIDIA® is the mainstream programing language for GPU computing. From a computing system perspective, a CUDA™ programmer has the option to define a three dimensional parallel *thread block* on a three dimensional *block grid.* A routine to be run on a thread on GPU is called a *kernel.* A *kernel call* runs a kernel on a predefined grid of blocks [48]. Synchronization among kernel calls is orchestrated by the CPU. In the low level, the computing hardware has multiple processors. In each processor, multiple threads execute the same instruction sequence on (usually different) data. Threads are grouped together in *warps.* Each warp

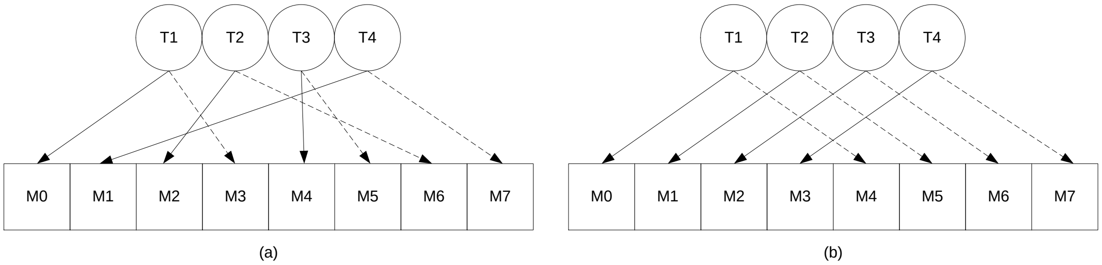


Figure 4.1 Examples for (a) non-coalesced and (b) coalesced memory access patterns for a kernel call with four threads accessing to eight memory locations in two iterations. T and M stand for thread and memory, respectively. First and second iterations are depicted with solid and dashed lines, respectively.

consists of 32 threads in the Fermi™ architecture [49]. A shared memory can be defined for each block that is only visible to itself. On the other hand, global memory (usually a DRAM) is visible to every other processor and to CPU as well. However, accessing the global memory is much slower than accessing the shared memory [16, 50, 51].

4.5.2 Memory Access in GPU Computing

Memory access (reaching out to a memory location for reading or writing data) time is an important limiting factor of state-of-the-art GPU computing technologies. Even the GPUs providing L1 and L2 caches for the GPU main memory, suffer from performance degradations when the memory access patterns by different threads are unstructured and/or non-coalesced [16, 48]. Memory coalescing, i.e., refining the memory access pattern such that the hardware can make combined requests from DRAM, should be employed whenever applicable. Examples of non-coalesced and coalesced access patterns are shown in Figures 4.1.a and 4.1.b, respectively, for an application with four threads and eight memory locations. It can be observed from the figure that the coalesced access pattern reaches to the adjacent locations in DRAM which provides maximum efficiency [16].

4.5.3 Implementation Overview

In all GPU implementations discussed in this chapter except OSPJ, two kernel calls; one for (4.15), and one for (4.16) and (4.17), are used for a step in a sweep of the parallel Jacobi algorithm [26]. Second kernel call must wait for the first one to complete its task via global synchronization. In the implementation of parallel Jacobi algorithm discussed, two GPU kernels running with $N/2$ blocks (processing units) and N threads (one thread for each vector element) are used. The global synchronization is realized through CPU (host synchronization).

4.5.4 Traditional and Modified Memory Access Methods

A linear array is the most common data structure used to store dense matrices in a computer memory. Programmers, in general, design an array as *row-major* or *column-major* where the elements are linearized based on their rows and columns, respectively. For instance, let $\mathbf{Z}$ be a 2×2 matrix as given

$$\mathbf{Z} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}. \tag{4.18}$$

Then, row-major and column-major arrays for storing the matrix $\mathbf{Z}$ become

$$\begin{aligned} \mathbf{z}_R &\triangleq \begin{bmatrix} Z_{11} & Z_{12} & Z_{21} & Z_{22} \end{bmatrix} \\ \mathbf{z}_C &\triangleq \begin{bmatrix} Z_{11} & Z_{21} & Z_{12} & Z_{22} \end{bmatrix}, \end{aligned} \tag{4.19}$$

respectively. Parallel Jacobi algorithm in GPU is implemented using row-major arrays to store $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{\Phi}$ in (4.15), (4.16), and (4.17) in memory. This access method is named as “the Traditional Access (TA)” as it is the most common and easiest way of representing matrix in the linear computer memory. It is noted that TA leads to a natural coalesced access in both reading from $\mathbf{A}$ and writing to $\mathbf{X}$ in (4.15) that corresponds to the access scheme given in Figure 4.1.b. However, both reading and writing operations lead to a non-coalesced access in (4.16) and (4.17) that corresponds to the access scheme given in Figure 4.1.a.

A straightforward way to handle this concern is to employ a row-major array to store matrix $\mathbf{A}$, and column-major arrays to store matrices $\mathbf{X}$ and $\mathbf{\Phi}$. This access method is named as “the Modified Access (MA).” MA leads to a non-coalesced access only when writing to matrix $\mathbf{X}$ in (4.15), and when writing to matrix $\mathbf{A}$ in (4.16). Other than those two, all access patterns in MA become coalesced that helps the GPU to access its DRAM more efficiently and provide results faster. In Section 4.6, it is shown that the computational performance improves significantly in MA method compared to TA. In the next subsection, a novel modification to MA method is introduced which ensures full coalesced access in performing the tasks of (4.16).

4.5.5 Symmetric Access Method

Since $\mathbf{A}$ is symmetric, its i*th* row is identical to its i*th* column at all times. Therefore, a processing unit can update the p*th* and q*th* rows of $\mathbf{A}$ in (4.16) instead of updating the p*th* and q*th* columns as given (4.16). In other words, modifying (4.16) as

$$\overline{\mathbf{A}}^{(m)} \leftarrow \left[\mathbf{X}^{(m)}\mathbf{J}^{(m)}\right]^{\mathrm{T}}, \tag{4.20}$$

leads one to the same solution. Explicitly, after every processing unit completes the update given in (4.20), matrix $\mathbf{A}$ is updated in the same way as it would be updated with (4.16) since $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$. However, this modification ensures the coalesced access when writing into matrix $\mathbf{A}$ that improves the computational efficiency. This access method is named as “the Symmetric Access (SA)” and it is shown that its performance is superior compared to MA in Section 4.6. It is noted that even with SA, there is still one non-coalesced access in (4.15) when writing into matrix $\mathbf{X}$. The trick used in SA for matrix $\mathbf{A}$ can not be applied directly to $\mathbf{X}$ since it is not symmetrical. Nevertheless, a new method that also provides coalesced access to the memory locations reserved for $\mathbf{X}$ is introduced in the next subsection.

4.5.6 Maximum-Coalesced Access Method

By changing the nature of the update procedure in implementing (4.15) it is possible to ensure all memory access of reading and writing operations in the parallel Jacobi algorithm are coalesced. Proposed modification makes use of the predetermined nature of the chess tournament algorithm (4.14). This method is named as "the Maximum-Coalesced Access (MCA)." In MCA, the update given in (4.15) is modified as follows

$$\mathbf{X}^{(m)} \leftarrow \mathbf{K}\left[\overline{\mathbf{A}}^{(m)}\right]^{\mathrm{T}}, \tag{4.21}$$

where $\mathbf{K}$ is an $N \times N$ matrix defined as

$$[K_{ij}] = \begin{cases} J_{11}^{(m)} & i = p^{(m)},\ j = p^{(m)} \\ J_{21}^{(m)} & i = p^{(m)},\ j = q^{(m)} \\ J_{12}^{(m)} & i = q^{(m)},\ j = p^{(m)} \\ J_{22}^{(m)} & i = q^{(m)},\ j = q^{(m)} \\ 0 & otherwise \end{cases}, \tag{4.22}$$

$J_{ij}^{(m)}$ is the element of Jacobi sub-rotation matrix, (4.13), and $m = 1, 2, \ldots, N/2$. It is noted that $\mathbf{K}$ is constant for all processing units in a sweep. In MCA, m*th* processing unit updates the $p^{(m)}th$ and $q^{(m)}th$ columns of $\mathbf{X}$ as follows

$$\begin{aligned} X_{ip^{(m)}} &= A_{ip^{(m)}} K_{ii} + A_{f(i)p^{(m)}} K_{if(i)} \\ X_{iq^{(m)}} &= A_{iq^{(m)}} K_{ii} + A_{f(i)q^{(m)}} K_{if(i)}, \end{aligned} \tag{4.23}$$

where $i = 1, 2, \ldots, N$ and $f(\cdot)$ is a mapping defined as

$$f(x) \triangleq g(x) \cup g^{-1}(x), \tag{4.24}$$

$g(x)$ is a mapping from set $\left\{p^{(m)}\right\}$ to set $\left\{q^{(m)}\right\}$ expressed as

$$g : p^{(m)} \rightarrow q^{(m)}. \tag{4.25}$$

It is noted that g is one-to-one. Hence, its inverse g^{-1} exists. Since sets $\left\{p^{(m)}\right\}$ and $\left\{q^{(m)}\right\}$ are known in advance in the algorithm, it is feasible to realize the update equation given in (4.23) . Moreover, since (4.23) accesses only to $2N$ elements of $\mathbf{K}$, it is more efficient (in terms of memory requirements) to define two $N \times 1$ vectors $[u_i] = K_{ii}$ and $[w_i] = K_{if(i)}$ instead of $N \times N$ sized $\mathbf{K}$. Then, one may modify (4.23) accordingly, as expressed

$$\begin{aligned} X_{ip^{(m)}} &= A_{ip^{(m)}}u_i + A_{f(i)p^{(m)}}w_i \\ X_{iq^{(m)}} &= A_{iq^{(m)}}u_i + A_{f(i)q^{(m)}}w_i. \end{aligned} \tag{4.26}$$

It is worth noting that the inherent symmetry of matrix $\mathbf{A}$ is implicitly exploited again in MCA, $\left[\overline{\mathbf{A}}^{(m)}\right]^{\mathrm{T}}$ in (4.21) accounts for accessing the $p^{(m)}th$ and $q^{(m)}th$ rows of $\mathbf{A}$ (in accordance with its row-major array data structure) and using its transpose such that a matrix of $N \times 2$ is used. Also, update given in (4.26) might still lead to non-coalesced access when reading from the global memory if the algorithm is not coded carefully. Therefore, in the GPU implementation of MCA, whenever applicable, reading from global memory is done in a coalesced way first and the values are stored in the shared memory before performing any rotation on them. Shared memory is faster and almost prune to non-coalesced access [16, 48].

The complexity of the algorithm is significantly increased in MCA compared to the other methods considered earlier. Moreover, all processing units need to calculate (or share) the rotation matrix for every pair (matrix $\mathbf{K}$) in the algorithm resulting in higher computational load or memory usage per processing unit. However, it is shown in the Section 4.6 that this overload is highly negligible, and it is well justified by the significance of improvements provided by MCA.

In order to fix the ideas, a simple example of MCA for $N = 4$, hence, for a matrix of size 4×4 is given next. It is noted that $N/2 = 2$ processing units are needed in this case. It is assumed that the algorithm is at its second step in the sweep where $p^{(1)} = 1$, $p^{(2)} = 2$, $q^{(1)} = 4$, and $q^{(2)} = 3$ according to (4.14). At this step, matrix $\mathbf{K}$ of (4.22) is written as

$$\mathbf{K} = \begin{bmatrix} J_{11}^{(1)} & 0 & 0 & J_{21}^{(1)} \\ 0 & J_{11}^{(2)} & J_{21}^{(2)} & 0 \\ 0 & J_{12}^{(2)} & J_{22}^{(2)} & 0 \\ J_{12}^{(1)} & 0 & 0 & J_{22}^{(1)} \end{bmatrix}, \tag{4.27}$$

where $J_{ij}^{(m)}$ is given in (4.13). It is noted that for the case at hand, the two arrays in (4.26) can be expressed as

$$\begin{aligned} \mathbf{u} &= \begin{bmatrix} J_{11}^{(1)} & J_{11}^{(2)} & J_{22}^{(2)} & J_{22}^{(1)} \end{bmatrix} \\ \mathbf{w} &= \begin{bmatrix} J_{21}^{(1)} & J_{21}^{(2)} & J_{12}^{(2)} & J_{12}^{(1)} \end{bmatrix}, \end{aligned} \tag{4.28}$$

since the following is true

$$\begin{aligned} & g : \{1 \to 4\} \cup \{2 \to 3\} \\ & f : \{1 \to 4\} \cup \{2 \to 3\} \cup \{4 \to 1\} \cup \{3 \to 2\}. \end{aligned} \tag{4.29}$$

It is clear from (4.27) and (4.28) that storing arrays $\mathbf{u}$ and $\mathbf{w}$ instead of matrix $\mathbf{K}$ saves memory space by the elimination of unnecessary storage of the zeros in matrix $\mathbf{K}$. Finally, using (4.26), (4.28), and (4.29) operations that are performed in the first update of the first processing unit are written as

$$\begin{aligned} X_{11} &= A_{11}K_{11} + A_{41}K_{14} \\ X_{21} &= A_{21}K_{22} + A_{31}K_{23} \\ X_{31} &= A_{31}K_{33} + A_{21}K_{32} \\ X_{41} &= A_{41}K_{44} + A_{11}K_{41}, \end{aligned} \tag{4.30}$$

where X_{ij} is located at the i*th* row and j*th* column of matrix $\mathbf{X}$. Operations for the second step, and for the both steps of the second processing unit are straightforward. Example is finished by providing the expanded version of (4.21) within this context in order to stress that MCA method accesses the columns of matrix $\mathbf{X}$ and rows of matrix $\mathbf{A}$ for the update given in (4.15) as follows

$$\begin{bmatrix} X_{11} & X_{14} \\ X_{21} & X_{24} \\ X_{31} & X_{34} \\ X_{41} & X_{44} \end{bmatrix} = \mathbf{K} \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}^{\mathrm{T}}, \tag{4.31}$$

where matrix $\mathbf{K}$ is defined in (4.27).

4.5.7 One Step Parallel Jacobi Algorithm

It was discussed in Section 4.3 that multiplying the data matrix with $\mathbf{J}^{\mathrm{T}}(p,q)$ from left, and with $\mathbf{J}(p,q)$ from right, and overwriting the result onto itself, updates the rows and columns of data matrix, respectively. Therefore, in any parallel implementation of the Jacobi algorithm, these two multiplications must be performed in two kernels as given in (4.15) and (4.16) with proper synchronization among the assigned processing units. However, thanks to MCA discussed earlier, it is possible to perform these two updates in only one kernel. By substituting (4.21) into (4.20) following update equation is obtained

$$\overline{\mathbf{A}}^{(m)} \leftarrow \left[\mathbf{K}\left[\overline{\mathbf{A}}^{(m)}\right]^{\mathrm{T}} \mathbf{J}^{(m)}\right]^{\mathrm{T}}. \tag{4.32}$$

It is noted that (4.32) updates only the $p^{(m)}th$ row and $q^{(m)}th$ column of $\mathbf{A}$, and it does not need intermediary matrix $\mathbf{X}$. The algorithm implementing (4.32) is named as “One Step Parallel Jacobi Algorithm (OSPJ).” OSPJ delivers the best performance among other GPU implementations as reported in the next section.

4.6 Comparison of CPU and GPU Implementations

CPU and GPU implementations discussed earlier are tested on a six core (a total of twelve cores with hyper-threading) Intel® Core™ i7-3960X CPU @ 3.30GHz with 32 GB RAM personal computer running on Linux. The GPU used in the tests is an NVIDIA® GeForce™ GTX 580 built with Fermi™ architecture [49] with 512 CUDA™ Cores and 1536 MB global memory. Source codes are compiled with CUDA™ Compiler Driver v5.0. All floating point operations are performed with single-precision. Timing results are averaged over 20 runs. Number of sweeps in all tests is fixed to 6 in order to make a fair comparison.

Data matrix used in the tests is chosen to be the auto-correlation matrix of an AR(1) source with a correlation coefficient ρ, defined as [5]

$$\mathbf{R}_x = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix}, \tag{4.33}$$

where σ_x^2 is the variance of the input and $-1 < \rho < 1$. Parameters are chosen to be $\sigma_x^2 = 1$ and $\rho = 0.9$ in the tests.

Computation time in milliseconds for single- and multi-threaded CPU as well as various GPU implementations with different memory access patterns (TA, MA, SA, MCA, and OSPJ) as a function of input matrix size are tabulated in Table 4.1, and also displayed in Figure 4.2.a. For GPU experiments, time needed to copy data from and to the system memory is taken into account. Multi-threaded CPU implementation can reach the performance of the single-threaded implementation when $N = 1,024$. Computation time for $N = 2,048$ are 2,882.2 and 729.3 seconds for single- and multi-threaded CPU implementations, respectively. These results are inline with the findings reported in [42] and clearly shows that overhead of thread creation and synchronization disqualifies CPU from being a feasible environment for large matrices.

Table 4.1 Computation Time in Milliseconds for Single- and Multi-Threaded CPU (First and Second Rows) and for GPU Implementations with Different Memory Access Patterns (Third to Last Rows) Versus the Input Matrix Size, N

	$N = 8$	$N = 16$	$N = 32$	$N = 64$	$N = 128$	$N = 256$	$N = 512$	$N = 1,024$
CPU (ST)	0.05	0.97	2.30	17.25	136.10	1,326.88	11,570.31	110,761.36
CPU (MT)	6.34	15.83	64.48	342.07	1,252.13	7,407.67	22,845.91	112,226.61
GPU (TA)	1.68	3.45	7.72	15.85	40.40	176.45	980.70	8,771.24
GPU (MA)	1.69	3.47	7.46	15.18	35.01	116.51	581.91	6,037.52
GPU (SA)	1.70	3.40	7.28	14.51	32.92	100.21	449.65	3,735.22
GPU (MCA)	1.72	3.45	7.15	14.44	31.63	89.29	388.88	2,338.69
GPU (OSPJ)	1.41	2.61	5.42	11.14	24.27	65.47	249.83	1,616.85

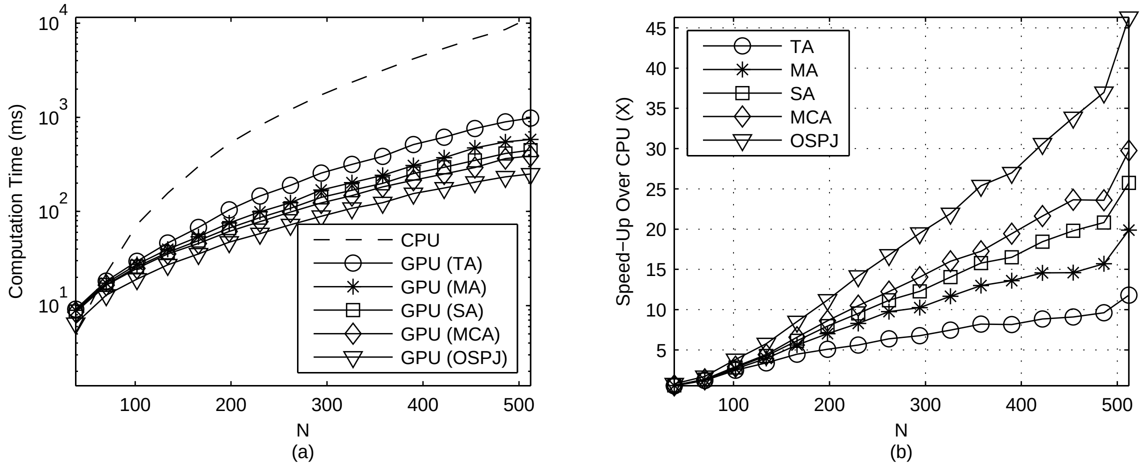


Figure 4.2 (a) Computation times of cyclic Jacobi algorithm in milliseconds on CPU; TA, MA, SA, MCA, and OSPJ on GPU; (b) Speed-up of GPU implementations over cyclic Jacobi algorithm on CPU, for various matrix sizes, N.

For the GPU implementations, as expected, speed is best for the OSPJ. Performance improvement of MA method over TA calculated as $(t_{TA} - t_{MA})/t_{TA} \times 100\%$ is 31.2% for $N = 1,024$. Similarly, for $N = 1,024$, performance improvement of SA over TA and MA are 57.4% and 38.1%, respectively; MCA over TA, MA, and SA are 73.3%, 61.3%, and 37.4% respectively; OSPJ over TA, MA, SA, and MCA are 81.6%, 73.2%, 56.7%, and 30.9%, respectively. Speed-up of TA, MA, SA, MCA, and OSPJ on GPU implementations over cyclic Jacobi algorithm on single-threaded CPU versus the input matrix size are shown in Figure 4.2.b. For $N = 1,024$, the speed-up of TA, MA, SA, MCA, and OSPJ on GPU over single-threaded CPU are $12.6\times$, $18.3\times$, $29.7\times$, $47.4\times$, and $68.5\times$ respectively.

4.7 Chapter Summary

Celebrated Jacobi algorithm for eigenanalysis and its parallel implementation using the chess tournament algorithm on graphics processing units (GPU) are revisited. It was shown that even with the multi-threaded implementations, CPU is a poor environment for the problem at hand. Moreover, overhead of thread creation and synchronization is non-negligible for matrix sizes smaller than 1,024. For the GPU implementations, it is highlighted that memory is a limiting performance factor. Three novel implementations with better memory access patterns leading to drastic performance improvements are introduced. The best GPU design proposed, OSPJ, is quantified to achieve 81.6% computational performance improvement over the traditional GPU methods, and 68.5 times faster implementation over single-threaded CPU for a dense symmetric matrix of size $N = 1,024$ under the same test conditions.

CHAPTER 5

FUNDAMENTALS OF QUANTITATIVE FINANCE

Techniques proposed in this dissertation are applicable to any problem in which Karhunen-Loève transform is used including the ones in quantitative finance. One of the pioneering industries in high performance computing and analytics on big data is finance. Methods proposed in the earlier chapters are directly applicable to some problems that commonly arise in quantitative finance. Although digital signal processing and quantitative finance has their ties, relationship between these two is mostly unexplored. In both fields, one of the main objectives is to extract information out of signals otherwise seem random. Although there has been increasing activity in the signal processing community on applications in finance over the last five years [21, 22], ties are still loose and more work has to be done.

In this chapter, fundamental topics in quantitative finance including continuous- and discrete-time price models for stocks, jumps, volatility, cross-correlation of assets and their widely-used applications such as portfolio optimization, pairs trading, hedging, and Epps effect are briefly discussed from a signal processing perspective. Goal of the chapter is to provide the framework for the problems discussed in the next chapter.

5.1 Price Models

A financial asset is a legal document that carries ownership. An equity or a stock is a share in a company. Bond conveys the ownership of credit. Derivatives are financial assets with their values depend on an underlying asset. For example, options are derivatives that conveys right to buy or sell another asset. Value of a financial asset is measured by price in the unit of a currency. There are a vast number of financial assets and a great deal of models for each. In this dissertation, discussion is limited to stocks. In this section, some basic continuous-time models for the price of a stock are discussed. More detailed discussion can be found in many textbooks including [52].

5.1.1 Geometric Brownian Motion Model

Brownian motion, first discussed by Brown in 1827 in the context of motion of pollens, later explained by Einstein in 1905, and formulated by Wiener in 1918 has strong ties with the financial modeling. Bachelier in 1900 modeled the price of a stock as a Brownian motion as given [53]

$$p(t) = p(0) + \mu t + \sigma w(t), \tag{5.1}$$

where $t \geq 0$ is the independent time variable, $p(t)$ is the price of the stock with an initial value $p(0)$, σ is the volatility, μ is the drift, and $w(t)$ is a Wiener process or standard Brownian motion such that $dw(t)$ is a zero-mean and unit-variance Gaussian process, i.e., $dw(t) \sim \mathcal{N}(0, 1)$. However, this model is problematic since it is possible for $p(t)$ defined in (5.1) to go below zero where as a stock price is always positive. Moreover, according to the model given in (5.1), change in price over a period is not a function of the initial price, $p(0)$. It suggests that stocks with different initial prices can have similar gains or losses in the same time interval which is not the case in reality (For example, probability of observing a $\$1$ change in price over a day is less for a stock priced at $\$10$ than it is for a stock that is worth $\$100$). A better model for the stock price is the geometric Brownian motion which resolves the two issues discussed. It is also referred to as Black-Scholes model [54] in which the rate of return of a stock is defined as

$$\frac{dp(t)}{p(t)} = \mu dt + \sigma dw(t). \tag{5.2}$$

This stochastic differential equation has its analytic solution obtained by using Itō's Lemma [55] and expressed as

$$p(t) = p(0) \exp\left[\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma w(t)\right]. \tag{5.3}$$

The expected value and variance of $p(t)$ are expressed as, respectively,

$$\begin{aligned} E\left\{p(t)\right\} &= p(0)e^{\left(\mu+\sigma^2/2\right)t} \\ \mathrm{var}\left\{p(t)\right\} &= p^2(0)e^{\left(2\mu+\sigma^2\right)t}\left(e^{\sigma^2 t}-1\right), \end{aligned} \tag{5.4}$$

Proofs of (5.3) and (5.4) can be found in Appendix C. It is noted that the price is distributed as log-normal and the log-price defined as

$$\begin{aligned} s(t) &= \ln p(t) \\ &= \ln p(0) + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W(t), \end{aligned} \tag{5.5}$$

is distributed as normal.

5.1.2 Models with Local and Stochastic Volatilities

Geometric Brownian motion price model has constant deviation in its returns, i.e., constant volatility, σ. Assuming that the volatility of a stock is constant is not always realistic since the markets and prices of stocks are affected by various events that occur randomly that may last for a long time in some cases, e.g., an economical crisis, or appear and vanish within minutes, e.g., the *flash crash* of 2010 [56]. Improved price models with local [57, 58] and stochastic [59, 60] volatilities take into account that the volatility itself is a function of time. In models with local volatility, the price model given in (5.2) is updated as

$$\frac{dp(t)}{p(t)} = \mu dt + \sigma\left[p(t), t\right]dw(t), \tag{5.6}$$

where $\sigma\left[p(t), t\right]$ is the local volatility that depends on both time, t, and the price at time t, $p(t)$. On the other hand, models with stochastic volatility has the following form for the return

$$\frac{dp(t)}{p(t)} = \mu dt + \sigma\left(t\right)dw_1(t), \tag{5.7}$$

where $\sigma(t)$, i.e., volatility as a function of time, is a random process. One of the popular stochastic volatility models is the Heston [60] model in where volatility is a random process that satisfies the stochastic differential as given

$$d\sigma(t) = \kappa \left[\theta - \sigma(t)\right] dt + \gamma \sqrt{\sigma(t)} dw_2(t), \tag{5.8}$$

where κ is the mean-reversion speed, θ is the volatility in the long-term,γ is the volatility of the volatility, $\sigma(t)$, and $dw_2(t)$ is a normal process correlated with $dw_1(t)$ given in (5.7). According to the Heston model, volatility is a mean-reverting process with a constant volatility and its infinitesimal changes are related to the ones of the price. It is noted that the model given in (5.8) is related to the celebrated Cox-Ingersoll-Ross process [61] used to model the short-term interest rates. Further details on the topic are out of scope of this dissertation and can be found in various texts [52].

5.2 Discrete-Time Price Models

Although price models described in the previous section are defined in continuous-time, in reality, price changes happen at discrete time points. It is a common practice to sample the price and to refer the stock returns with respect to their sampling periods, e.g., 30-min returns, 1-hour returns, end of day (EOD) returns. In this section, the most basic discrete-time price model, i.e., the geometric Brownian motion model is revisited. Then, a brief discussion on jumps in the returns of stocks is given.

5.2.1 Discrete-Time Geometric Brownian Motion Model

Discrete-time analog of geometric Brownian motion model is obtained by sampling as given

$$s(n) = s(n-1) + \mu + \sigma\xi(n), \tag{5.9}$$

where $s(n) = \ln p(n)$ is the log-price of a stock at discrete-time n with price $p(n)$, μ, and σ are the the drift and volatility of the stock, respectively, and $\xi(n)$ is the white Gaussian noise with $\xi(n) \sim \mathcal{N}(0,1)$. The log-return at discrete-time n is defined as

$$g(n) = \mu + \sigma\xi(n). \tag{5.10}$$

It follows from (5.9) and (5.10) that log-return is a Gaussian process with mean μ and variance σ^2, i.e., $g(n) \sim \mathcal{N}(\mu, \sigma^2)$. Moreover, it is a stationary noise process and white, i.e.,

$$E\left\{g(n-k)g(n-l)\right\} - \mu^2 = \sigma^2\delta_{k-l}. \tag{5.11}$$

Log-price given in (5.9) is equal to

$$s(n) = s(n-1) + g(n). \tag{5.12}$$

Rate of return or simply the return of a stock, is defined as the ratio of the difference in its price between the current and the previous samples over the price associated with the previous sample as given

$$r(n) = \frac{p(n) - p(n-1)}{p(n-1)} = \frac{p(n)}{p(n-1)} - 1. \tag{5.13}$$

For small values, return $r(n)$ is an approximation to the log-return $g(n)$ due to the Taylor series expansion of the logarithm, i.e.,

$$g(n) = s(n) - s(n-1) = \ln\left[\frac{p(n)}{p(n-1)}\right] \cong \frac{p(n)}{p(n-1)} - 1 = r(n). \tag{5.14}$$

It is noted that since the value of return might get very small, it is customary in finance to use basis points (bps) instead of percent. One bps is the one percent of a percent, i.e., $1 \text{ bps} = 0.01\%$.

5.2.2 Effect of Sampling Frequency on Volatility

It follows from (5.12) that one can write the log-price at discrete time n as a sum of initial log-price and all log-returns up to n as follows

$$s(n) = s(0) + \sum_{i=1}^{n} g(i). \tag{5.15}$$

If $s_{T_1}(n)$ and $s_{T_2}(n)$ are two discrete-time log-prices of the same stock, sampled with sampling periods $T_s = T_1$ and $T_s = T_2$, respectively, with $T_2 = kT_1$, $s_{T_2}(n) = s_{T_1}(kn)$, $k \in \mathbb{Z}$, and $k > 0$, then it follows from (5.12) and (5.15) that

$$\begin{aligned} g_{T_2}(n) &= s_{T_2}(n) - s_{T_2}(n-1) \\ &= s_{T_1}(kn) - s_{T_2}(kn-k) \\ &= s_{T_1}(0) + \sum_{i=1}^{kn} g_{T_1}(i) - s_{T_1}(0) - \sum_{i=1}^{kn-k} g_{T_1}(i) \\ &= \sum_{i=0}^{k-1} g_{T_1}(kn-i), \end{aligned} \tag{5.16}$$

where $g_{T_1}(n)$ and $g_{T_2}(n)$ are the log-returns associated with $s_{T_1}(n)$ and $s_{T_2}(n)$, respectively, via (5.15). Since the summation of Gaussian random variables is also a Gaussian random variable, if $g_{T_1}(n) \sim \mathcal{N}(\mu, \sigma^2)$ then $g_{T_2}(n) \sim \mathcal{N}(k\mu, k\sigma^2)$, and

$$\sigma_{T_2} = \sqrt{k}\sigma_{T_1}, \tag{5.17}$$

where σ_{T_1} and σ_{T_2} are the standard deviation of $g_{T_1}(n)$ and $g_{T_2}(n)$, respectively. Equality given in (5.17) reads that volatilities at different sampling frequencies differ by a scale in square root of their ratio.

5.2.3 Discrete-Time Price Model with Jumps

Geometric Brownian motion model and its improved versions with local and stochastic volatilities discussed in Section 5.1 all have the continuity property. However, price of a stock is impacted by many reasons including stock specific and stock related business

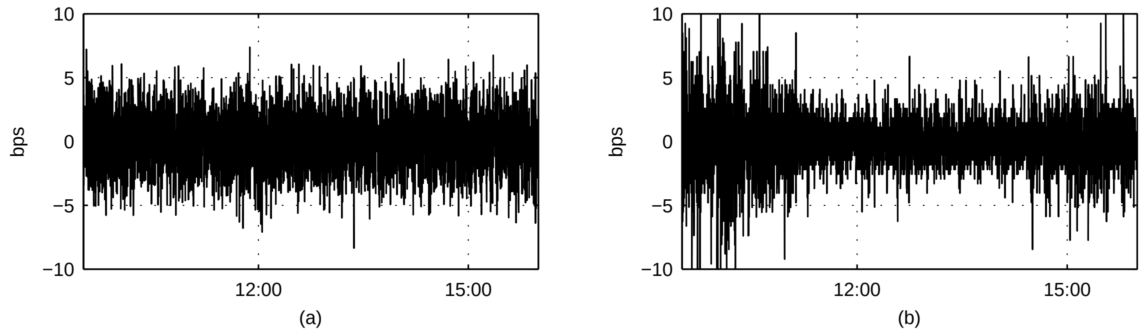


Figure 5.1 (a) A realization of a white Gaussian random process and (b) Returns of Apple Inc. (AAPL) stock on June 17, 2010.

developments and financial news. Although some of those news are anticipated, there are many instances that these higher impact events happen quite randomly. One may observe upward and downward abrupt price changes on any stock. These abrupt changes are called as jumps in finance literature [62]. One of the simplest discrete-time price models with jumps is given as

$$s(n) = s(n-1) + j(n) + \xi(n), \tag{5.18}$$

where $j(n) \in \mathbb{R}$ is the abrupt price change, up or down, that happens at discrete-time n, and $\xi(n) \sim \mathcal{N}(\mu, \sigma)$ is a Gaussian random process. It is noted that in (5.18), the random log-return $g(n)$ of (5.12) is modeled as the summation of two processes. Namely, a jump process $j(n)$, and a pure Gaussian noise process $\xi(n)$,

$$g(n) = j(n) + \xi(n) \tag{5.19}$$

In Figure 5.1.a realization of a Gaussian random process $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 0.01$ bps and $\sigma = 2.11$ bps is shown. In Figure 5.1.b, log-return of Apple Inc. (AAPL) stock on day June 17, 2010 with a sampling period of $T_s = 5s$ is displayed. For this case, estimated mean (drift) and standard deviation (volatility) of the returns are 0.01 bps and 2.11 bps, respectively. It is observed from Figures 5.1.a and 5.1.b that one needs to consider the jump process in the price model in order to employ the basic price model more properly.

Any jump of high significance is the main reason for the so-called regime change in a stock price. In order to highlight the importance of jump processes in price modeling a simple experiment is designed as follows. The volatility estimation error is defined as

$$\varepsilon = \left| \hat{\sigma}(m)\sqrt{k/m} - \hat{\sigma}(k) \right| , \tag{5.20}$$

where $\hat{\sigma}(m)$ and $\hat{\sigma}(k)$ are the volatilities estimated at $T_s = m$ and $T_s = k$, respectively, via

$$\hat{\sigma}\left(T_s\right) = \left(\frac{1}{N-1} \sum_{i=0}^{N-1} \left[g_{T_s}(n-i) - \hat{\mu}\left(T_s\right) \right]^2 \right)^{1/2} , \tag{5.21}$$

$g_{T_s}(n)$ is the log-return of associated log-price sampled with the period T_s, $\hat{\mu}\left(T_s\right)$ is the estimated mean of the log-return as given

$$\hat{\mu}\left(T_s\right) = \frac{1}{N} \sum_{i=0}^{N-1} g_{T_s}(n-i), \tag{5.22}$$

and N is the estimation window length in samples. If the return process $g(n)$ in (5.19) were pure Gaussian, than the error term ε would be zero in accordance with (5.17). A histogram based price jump detector is employed where a return is labeled as a jump if its absolute value is larger than four times the estimated volatility, i.e., $4\hat{\sigma}$. Next, an artificial “jump-free” return process is defined as

$$\hat{g}(n) = \hat{\xi}(n) = g(n) - \hat{j}(n). \tag{5.23}$$

Then, volatility estimation error (5.20) is calculated for various frequencies spanning from $k = 1s$ to $300s$ with $m = 1$ in for both log-return and jump-free log-return of AAPL on day June 17, 2010, i.e., $g(n)$ and $\hat{g}(n)$ defined in (5.19) and (5.23), respectively. Error defined in (5.20) is calculated as a function of frequency k, $\varepsilon(k)$, and is displayed in Figure 5.2. It is observed from the figure that removing jumps reduces the volatility estimation error and jump is an important phenomenon. One needs to take these abrupt changes into account in order to better model the price process. Further details are out of scope of this dissertation and can be found in the literature including [62].

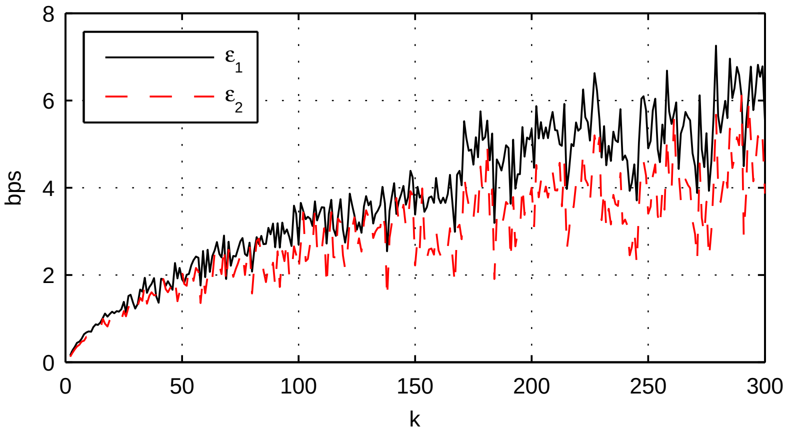


Figure 5.2 Volatility estimation error ϵ versus sampling period k with $m = 1$ as defined in (5.20) for real and artificial (jump-free) returns of (5.19) and (5.23), i.e., ε_1 and ε_2, respectively.

5.3 Cross-Correlation of Asset Returns and its Applications

Cross-correlation of asset returns in an investment portfolio is an important aspect of modern portfolio theory [17]. It also plays a key role in relative value models, hence in trading strategies such as pairs trading, hedging, and arbitrage. The cross-correlation coefficient of the returns of two assets, $r_1(n)$ and $r_2(n)$, is defined as

$$\rho = \frac{E\left\{r_1(n)r_2(n)\right\} - \mu_1\mu_2}{\sigma_1\sigma_2}, \tag{5.24}$$

where $\mu_i = E\left\{r_i(n)\right\}$ is the mean and $\sigma_i^2 = E\left\{r_i^2(n)\right\} - \mu_i^2$ is the variance of a return process. In this section, significance of cross-correlation of assets in some most common financial applications such as modern portfolio theory, hedging, and pairs trading, is studied in detail.

5.3.1 Portfolio Optimization and Modern Portfolio Theory

Portfolio return is the weighted average of the returns of the assets associated with it. Return of a two-asset portfolio is expressed as

$$r_p(n) = q_1(n)r_1(n) + q_2(n)r_2(n), \tag{5.25}$$

where n is the discrete time variable, $q_i(n)$ is the amount of capital invested in the i*th* asset, and $r_i(n)$ is the return of the i*th* asset defined in (5.13). The investment amount, $q_i(n)$ in (5.25), can be dimensionless or its unit may be a currency. This choice reflects itself into the unit of portfolio risk which is defined later in the section. The time index n is omitted in further discussions noting that each variable in an equation is a function of time. Expected return of the two-asset portfolio is calculated as

$$\mu_p = E\left\{r_p\right\} = q_1 E\left\{r_1\right\} + q_2 E\left\{r_2\right\}. \tag{5.26}$$

Standard deviation of the portfolio return, i.e., the risk of the portfolio is given as

$$\begin{aligned}\sigma_p &= \left(E\left\{r_p^2\right\} - E^2\left\{r_p\right\}\right)^{1/2} \\ &= \left(q_1^2\sigma_1^2 + 2q_1q_2\sigma_1\sigma_2\rho_{12} + q_2^2\sigma_2^2\right)^{1/2},\end{aligned} \tag{5.27}$$

where σ_i is the volatility, i.e., standard deviation of the returns of the i*th* asset, and ρ_{ij} is the cross-correlation coefficient between the returns of i*th* and j*th* assets. It is straightforward to generalize this concept to a portfolio consisting of N assets. Return of the N-asset portfolio is expressed as

$$r_p = \mathbf{q}^{\mathrm{T}}\mathbf{r} = \sum_{i=1}^{N} q_i r_i, \tag{5.28}$$

where superscript T is the matrix transpose operator, $\mathbf{q}$ is $N \times 1$ capital allocation vector defined as

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_N \end{bmatrix}^{\mathrm{T}}, \tag{5.29}$$

and $\mathbf{r}$ is an $N \times 1$ vector comprised of the returns of assets in the portfolio expressed as

$$\mathbf{r} = \begin{bmatrix} r_1 & r_2 & \cdots & r_N \end{bmatrix}^{\mathrm{T}}. \tag{5.30}$$

Hence, from (5.28), expected return of the portfolio is calculated as

$$\mu_p = E\left\{r_p\right\} = \mathbf{q}^{\mathrm{T}} E\left\{\mathbf{r}\right\} = \mathbf{q}^{\mathrm{T}} \boldsymbol{\mu}, \tag{5.31}$$

where elements of the $N \times 1$ vector $\boldsymbol{\mu}$ are the expected returns of individual assets. Similarly, risk of an N-asset portfolio is obtained as

$$\sigma_p = \left(E\left\{r_p^2\right\} - \mu_p^2\right)^{1/2} = \left(\mathbf{q}^{\mathrm{T}}\mathbf{C}\mathbf{q}\right)^{1/2} = \left(\mathbf{q}^{\mathrm{T}}\boldsymbol{\Sigma}^{\mathrm{T}}\mathbf{P}\boldsymbol{\Sigma}\mathbf{q}\right)^{1/2} = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \rho_{ij}\sigma_i\sigma_j\right)^{1/2}, \tag{5.32}$$

where $\boldsymbol{\Sigma}$ is an $N \times N$ diagonal matrix with elements corresponding to volatilities of assets σ_i, $\mathbf{C}$ is $N \times N$ covariance matrix of asset returns as defined

$$\mathbf{C} = E\left\{\mathbf{r}\mathbf{r}^{\mathrm{T}}\right\} - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}, \tag{5.33}$$

and $\mathbf{P}$ is $N \times N$ correlation matrix where

$$\mathbf{P} = [P_{ij}] = \rho_{ij} = \frac{E\left\{r_i r_j\right\} - \mu_i\mu_j}{\sigma_i\sigma_j}. \tag{5.34}$$

It is noted that all elements on the main diagonal of $\mathbf{P}$ are equal to one. Furthermore, $\mathbf{P}$ is a symmetric and positive definite matrix.

Modern portfolio theory (MPT) [17] suggests a method to create efficient portfolios with the minimized risk for a given expected return by optimally allocating the amount of capital invested in each asset of the portfolio. More formally, in MPT, portfolio optimization is achieved by minimizing the portfolio risk, σ_p, given in (5.32) with the constraint that the expected portfolio return, μ_p, of (5.31) is equal to a constant, i.e.,

$$\mu_p = \mathbf{q}^{\mathrm{T}}\boldsymbol{\mu} = \sum_{i=1}^{N} q_i\mu_i = \mu. \tag{5.35}$$

There might be additional constraints such as constant investment capital of portfolio, i.e.,

$$\mathbf{q}^{\mathrm{T}}\mathbf{1} = \sum_{i=1}^{N} q_i = 1, \tag{5.36}$$

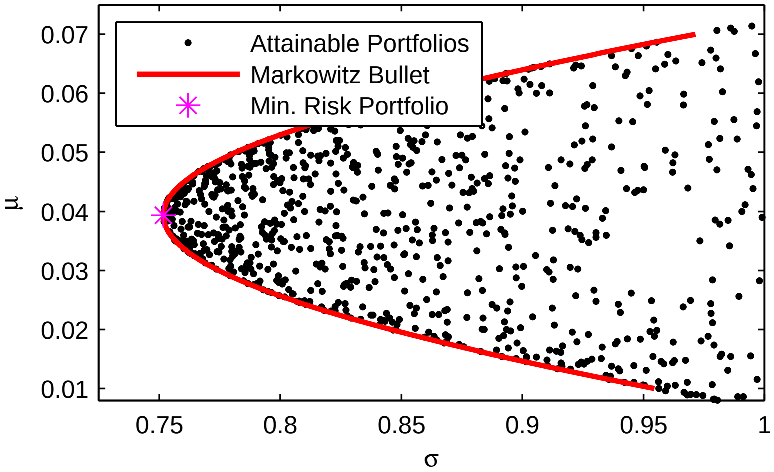


Figure 5.3 Markowitz bullet along with some of the attainable portfolios (black dots) and the minimum risk portfolio.

where $\mathbf{1}$ is an $N \times 1$ vector with all its elements equal to 1. The risk minimization problem to create an efficient portfolio subject to the constraints given in (5.35) and (5.36) can be solved by introducing two Lagrangian multipliers. Hence, the optimum investment allocation vector providing the minimum risk for a given expected return subject to the constraint of a constant portfolio investment is the solution to the minimization problem stated as

$$\mathbf{q}^* = \underset{\mathbf{q}}{\operatorname{argmin}} \frac{1}{2}\mathbf{q}^{\mathrm{T}}\mathbf{C}\mathbf{q} + \lambda_1 \left(\mu - \mathbf{q}^{\mathrm{T}}\boldsymbol{\mu}\right) + \lambda_2 \left(1 - \mathbf{q}^{\mathrm{T}}\mathbf{1}\right), \tag{5.37}$$

where $\mathbf{C}$ is the covariance matrix defined in (5.33). Solution to (5.37) is given as

$$\mathbf{q}^* = \frac{\begin{vmatrix} \mu & \mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\mu} \\ 1 & \mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} \end{vmatrix} \mathbf{C}^{-1}\boldsymbol{\mu} + \begin{vmatrix} \boldsymbol{\mu}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\mu} & \mu \\ \boldsymbol{\mu}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} & 1 \end{vmatrix} \mathbf{C}^{-1}\mathbf{1}}{\begin{vmatrix} \boldsymbol{\mu}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\mu} & \mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\mu} \\ \boldsymbol{\mu}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} & \mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} \end{vmatrix}}, \tag{5.38}$$

where $|\cdot|$ is the matrix determinant operator, and $\mathbf{C}^{-1}$ is the inverse of $\mathbf{C}$. Set of optimum portfolios each satisfying the constraints of (5.35) and (5.36) for $-\infty < \mu < \infty$, form a curve in the (σ, μ) plane. This curve is called the *Markowitz bullet* and depicted in Figure

5.3 for the case of a three-asset portfolio with $\rho_{12} = 0.6$, $\rho_{13} = 0.2$, $\rho_{23} = 0.3$, $\mu_1 = 0.07$, $\mu_2 = 0.03$, and $\mu_3 = 0.02$. All of the (σ_p, μ_p) pairs for attainable portfolios, i.e., portfolios that satisfy the constraint of (5.36) are on or to-the-right of the bullet. Some of the attainable portfolios are illustrated as black dots in Figure 5.3 where investment vectors of those portfolios are drawn from a Gaussian joint-probability density function, i.e., $\mathbf{q} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and only the ones that satisfy the constraint given in (5.36) are kept. Portfolios that lie on the upper-half of the Markowitz bullet are called efficient and they form the *efficient frontier*. Furthermore, only one of the efficient portfolios has the minimum risk, and therefore, it is called as the *minimum risk portfolio*. The investment vector for the minimum risk portfolio is calculated by solving the following quadratic programming equation

$$\mathbf{q}_{min} = \underset{\mathbf{q}}{\operatorname{argmin}} \frac{1}{2} \mathbf{q}^{\mathrm{T}} \mathbf{C} \mathbf{q} + \left(1 - \mathbf{q}^{\mathrm{T}} \mathbf{1}\right). \tag{5.39}$$

The trivial solution is found as

$$\mathbf{q}_{min} = \frac{\mathbf{C}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1}}, \tag{5.40}$$

It is noted that $(\sigma_{min}, \mu_{min})$ pair corresponding to $\mathbf{q}_{min}$ is located at the far left tip of the Markowitz bullet as highlighted by an asterisk in Figure 5.3. The minimum risk portfolio is unique. It has the minimum attainable risk, σ_p; however, its expected return, μ_p, is not the best possible one.

5.3.2 Relative Value Model, Pairs Trading, and Hedging

Another application that the cross-correlation of asset returns comes into play is pairs trading. In this type of trading, return of an asset in time is modeled such that it is composed of a constant random variable, weighted return of another asset (usually an asset within the same industry or an exchange traded fund (ETF) that tracks the index of the corresponding industry [63]), and white noise as given

$$r_1(n) = \alpha + \beta r_2(n) + \xi(n), \tag{5.41}$$

where α, β, and ξ are commonly referred to as drift, systematic component, and idiosyncratic component, respectively. Model given in (5.41) is also referred to as relative value model. The idea in pairs trading is to invest 1 units of currency in the first asset and $-\beta$ units of currency in the second asset (hedge) such that the return on investment becomes

$$r_I(n) = r_1(n) - \beta r_2(n) = \alpha + \xi(n), \tag{5.42}$$

and the expected return on investment is calculated as

$$\mu_I = E\left\{r_I(n)\right\} = E\left\{\alpha\right\} + E\left\{\xi(n)\right\}. \tag{5.43}$$

According to (5.43), this strategy is expected to profit if $E\left\{\alpha\right\} > 0$ with $E\left\{\xi(n)\right\} = 0$ which means that the first asset outperforms the second for a prolonged time. Second scenario is that $E\left\{\alpha\right\} = 0$ and the last sample of the cumulative sum of the idiosyncratic component, $X(n) = \sum_i \xi(n-i)$, is far less than its mean, i.e., $X(n_0) \ll E\left\{X(n)\right\}$. In this case, an investor expects to profit since $X(n)$ is expected to return to its mean with more positive samples than negative samples of $\xi(n)$ in the short term. In this case, first asset is under-priced compared to the second due to an inefficiency in the market. If the first asset is over-priced, i.e., $X(n_0) \gg E\left\{X(n)\right\}$, the logical move is not to get in a long but a short position in the pair, i.e., to invest -1 units of currency in the first asset and β units of currency in the second asset. With pairs trading, investors try to isolate the return on investment from the market (industry) and bet against the excess returns of a specific asset. Assuming that the returns of the second asset is uncorrelated with the idiosyncratic component, i.e., $E\left\{r_2(n)\xi(n)\right\} = 0$, substitution of (5.41) into (5.24) yields

$$\beta = \rho_{12}\frac{\sigma_1}{\sigma_2}. \tag{5.44}$$

It is seen from (5.42) and (5.44) that the performance of a pairs trading strategy is related to the cross-correlation coefficient of two assets, ρ_{12}. Therefore, better correlation estimation of asset returns is an important factor of good performance.

5.4 Epps Effect

As it is discussed in the earlier sections, a good estimation of correlation is crucial for good performance in all trading and risk management systems [64]. However, a good correlation estimation, especially in intra-day and high-frequency trading where sampling periods are typically below a minute, is of a major challenge [65, 66, 67]. It is known in finance that the correlations among financial asset returns decrease as the sampling period of prices decreases. This phenomenon called Epps effect [23] is revisited in this section.

5.4.1 Cross-Correlation of Asset Returns as a Function of Sampling Period

Using (5.14) and assuming that the mean of log-returns is zero, i.e., $\mu_{T_1} = \mu_{T_2} = 0$, cross-correlation between the log-returns of two assets given in (5.24) can be written as a function of the sampling period as follows

$$\rho_{12}\left(T_s\right) = \frac{E\left\{g_{1,T_s}(n)g_{2,T_s}(n)\right\}}{\sigma_{1,T_s}\sigma_{2,T_s}}, \tag{5.45}$$

where $g_{1,T_s}(n)$ and $g_{2,T_s}(n)$ are the log-returns of the first and second assets sampled with T_s, respectively, with corresponding standard deviations, i.e., σ_{1,T_s} and σ_{2,T_s}. Similarly, cross-correlation between the log-returns of two assets both sampled with $T_s = T_2$ is equal to

$$\rho_{12}\left(T_2\right) = \frac{E\left\{g_{1,T_2}(n)g_{2,T_2}(n)\right\}}{\sigma_{1,T_2}\sigma_{2,T_2}}. \tag{5.46}$$

For $T_2 = kT_1$, it follows from (5.16) and (5.46) that

$$\rho_{12}\left(T_2\right) = \frac{1}{\sigma_{1,T_2}\sigma_{2,T_2}} E\left\{\sum_{i=0}^{k-1} g_{1,T_1}(kn-i) \sum_{j=0}^{k-1} g_{2,T_1}(kn-j)\right\}. \tag{5.47}$$

It is assumed that the cross-correlation between the samples of different asset log-returns sampled at different times is zero, i.e.,

$$E\left\{g_{1,T_s}(n-k)g_{2,T_s}(n-l)\right\} = \rho_{12}\left(T_s\right)\sigma_{1,T_s}\sigma_{2,T_s}\delta_{k-l}. \tag{5.48}$$

From (5.11), (5.17), (5.47), and (5.48) it is concluded that

$$\begin{aligned}\rho_{12}\left(T_2\right) &= \frac{k}{\sigma_{1,T_2}\sigma_{2,T_2}} E\left\{g_{1,T_1}(n) g_{2,T_1}(n)\right\} \\ &= \frac{k}{\sqrt{k}\sigma_{1,T_1}\sqrt{k}\sigma_{2,T_1}} E\left\{g_{1,T_1}(n) g_{2,T_1}(n)\right\} \\ &= \rho_{12}\left(T_1\right). \end{aligned} \tag{5.49}$$

It is shown in (5.49) that the cross-correlation coefficient, ρ_{12}, between the returns of two assets that follow geometric Brownian motion paths with pure Gaussian increments is not related to the sampling period. However, Epps [23] was the first to show that the empirical data does not comply with (5.49). Moreover, Epps stated that the cross-correlation between two financial assets decreases as the sampling period gets smaller, i.e.,

$$\rho_{12}(T_s) \rightarrow 0 \text{ as } T_s \rightarrow 0. \tag{5.50}$$

5.4.2 Empirical Evidence on Epps Effect

Cross-correlation coefficient between the log-returns of Apple Inc. stock (AAPL) and PowerShares QQQ Trust ETF (QQQ) estimated using 60 days of historical data between April 1, 2010 and June 30, 2010 as a function of sampling period is displayed in Figure 5.4.a. The sample correlation estimator employed for Figure 5.4 is written as

$$\hat{\rho}\left(T_s\right) = \frac{1}{N-1} \sum_{i=0}^{N} \bar{g}_{1,T_s}(n-i) \bar{g}_{2,T_s}(n-i), \tag{5.51}$$

where $\bar{g}_{k,T_s}(n)$ is the normalized log-return of the k*th* asset with zero mean and unit variance as calculated

$$\bar{g}_{k,T_s}(n) = \frac{g_{k,T_S}(n) - \hat{\mu}_k\left(T_s\right)}{\hat{\sigma}_k\left(T_s\right)}, \tag{5.52}$$

$\hat{\mu}_k\left(T_s\right)$ and $\hat{\sigma}_k\left(T_s\right)$ are the estimated mean and standard deviation of $g_{k,T_S}(n)$ defined in (5.22) and (5.21), respectively. Since AAPL is a significant member of NASDAQ100 index and QQQ mimics the behavior of NASDAQ100 index, one expects to have a significant

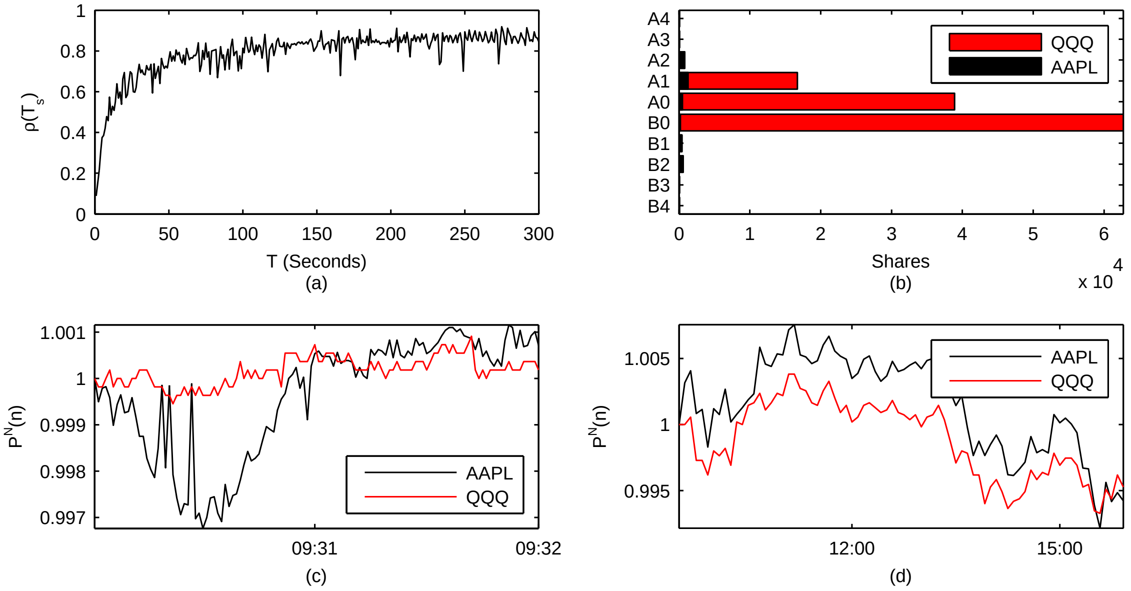


Figure 5.4 (a) Cross-correlation between the log-returns of AAPL and QQQ as a function of sampling period, (b) A typical snapshot of the first five levels of the LOBs for AAPL and QQQ, Normalized last traded prices of both stocks on March 17, 2011 (c) between 9:30am and 9:32am sampled with $T = 1s$, (d) between 9:30am and 4:00pm sampled with $T = 300s$.

correlation between the returns of these two relevant assets. However, Figure 5.4.a suggests a more complicated information. It is observed from the figure that assumption of (5.49) does not always hold, and the geometric Brownian motion for the log-price of the assets of (5.12) needs to be improved. This concern has been an active research topic where several authors proposed improved models (see [65, 66] and references therein.)

The most widely accepted cause of the Epps effect in finance is the nature of asynchronous trading. Although prices of the assets within the same industry tend to behave similarly, and they respond to various intra-day economical, social, and political news in the same way, they are not traded at the same time points or at the same side of the limit order book (LOB) that is a structure specific to each asset and venue. When a trader places a limit order to buy/sell an asset at some specific price, shares associated with the order are placed in the corresponding price level in the bid/ask side of LOB. The shares waiting to be bought/sold at the highest/cheapest price rest in the best bid (B0) and best ask (A0) levels of the LOB. Traders who place market orders to buy/sell an asset are matched with the shares waiting longest in the ask/bid side starting from the best ask/best bid levels.

Moreover, even though two different assets were traded at the same time, their market structures, i.e., LOBs, volume, and liquidity, are different and they play a significant role in price formation. As an example, a typical snapshot of the first five levels (on both bid and ask sides) of the LOBs for AAPL and QQQ are displayed in Figure 5.4.b. It can be observed that the levels of the LOB for AAPL do not offer too many shares available for selling or buying. However, there are many QQQ shares resting at best bid (B0) and best ask (A0) prices. Therefore it is less likely for QQQ to have the resting shares depleted on one side and provide a different last-traded price print in short term than it is for AAPL. Moreover, even if there were only two players in the market, who buy and sell same number of shares of AAPL and QQQ, there is no guarantee that they execute the trades synchronously, i.e., first buy AAPL and then sell QQQ or vice-versa. If both players completes the trade in T seconds, then for an observer sampling the last-traded prices with $T_s > T$, the asynchronous trades would not be visible, and the price prints of AAPL and QQQ would seem to happen together.

Normalized last-traded prices, i.e., $p^N(n) = p(n)/p(0)$, of both stocks on March 17, 2011 between 9:30am and 9:32am with $T_s = 1s$, and between 9:30am and 4:00pm with $T_s = 300s$ are shown in Figures 5.4.c and 5.4.d, respectively. It is observed from these figures that the good proxy between the prices of two stocks that exists at lower sampling rates disappears as the sampling rate increases. This is a very important phenomenon and a serious concern in particular for high-frequency trading since the traditional risk management framework does not hold to provide practical solutions in order to maintain an investment portfolio at high speeds.

Holdings of QQQ are comprised of NASDAQ 100 technology stocks with their relevant investment factors. Hence, QQQ and those stocks are expected to be correlated. Correlation coefficients between the log-returns of QQQ and its largest five holdings (AAPL:

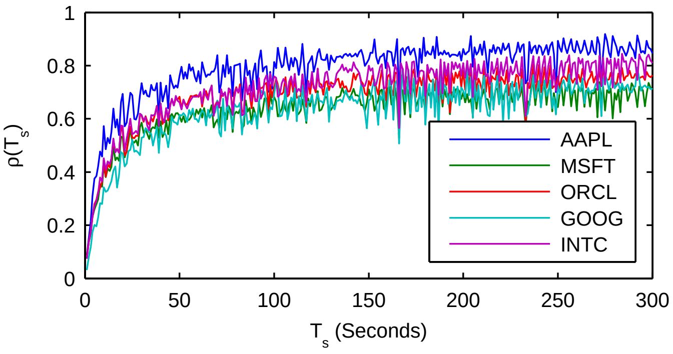


Figure 5.5 Cross-correlation between the QQQ and (a) its first five largest holdings as a function of sampling period, T.

Apple Inc., MSFT: Microsoft Corp., ORCL: Oracle Corp., GOOG: Google Inc., INTC: Intel Corp.) estimated using 60 days of historical data between April 1, 2010 and June 30, 2010 are displayed in Figure 5.5 as a function of the sampling period. It is observed from these figures that the correlation pattern and Epps effect observed in Figure 5.4.a. are not specific to AAPL and QQQ pair.

5.4.3 Product of Returns and Problems with the Sample Estimator

The built-in asynchronicity affects correlation estimation given in (5.51) since even the price of one asset of the pair does not change, the product of returns becomes zero, and that term is considered in the averaging operator. Similarly, a scenario in where the constant prices of both assets traded synchronously are also considered as perfectly uncorrelated pair of returns within this framework. Although these two zero-product cases for correlation calculations are distinct, it is more likely not to have price change at smaller time intervals. It is reasonable to expect higher correlation if correlation calculation only considers nonzero products with irregular sampling grid where price variations occur for both assets.

Figure 5.6.a displays histogram of pairwise product operations in log-returns in the correlation calculation of AAPL and QQQ pair with (5.51) at $1s$ sampling period. Similarly, Figure 5.6.b displays histogram for the case of end-of-day (EOD) correlation. Figure 5.6.c depicts probabilities of product terms used in the correlation calculations being

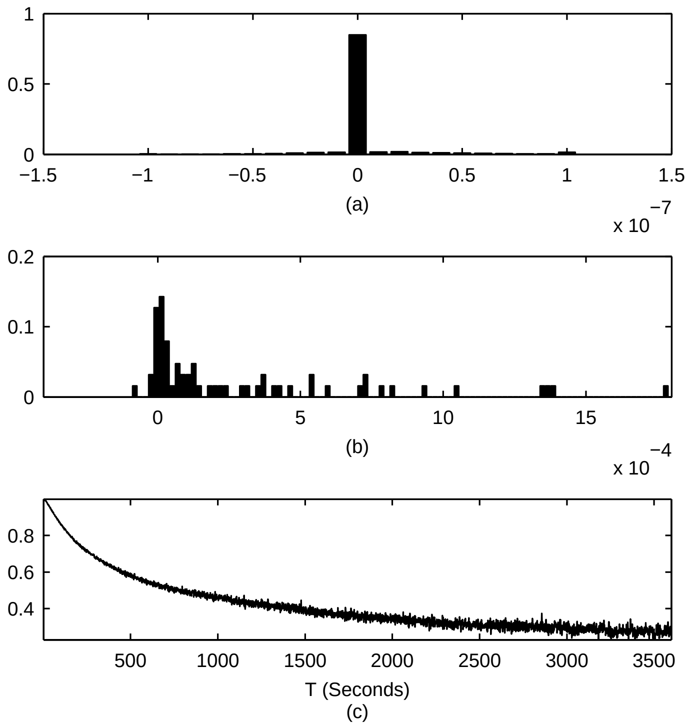


Figure 5.6 Histogram of pairwise products for the log-returns of AAPL and QQQ with sampling intervals (a) $T_s = 1s$, and (b) $T_s = 24h$ (EOD) (c) Probabilities of product terms being negligible as a function of T_s.

negligible (between $-\varepsilon$ and ε) as a function of sampling period for $\varepsilon = 3 \cdot 10^{-6}$. It is noted that the probability of having negligible product term in correlation calculation drops when sampling interval increases as depicted in this figure. This fact has direct impact on the values of pairwise correlations calculated through averaging that includes negligible ones. Hence, they drop significantly at higher sampling frequencies. Some researches have proposed improved versions of estimators given in (5.51) and (5.52) [67]. However, this point deserves further study.

5.5 Chapter Summary

Geometric Brownian motion is a widely used and most basic model used for the pricing of assets. Correlations of asset returns in an investment portfolio is an important aspect of the modern portfolio theory [17]. They also play a key role in relative value models, hence in investment strategies such as pairs trading, hedging, and arbitrage. According to discrete-time geometric Brownian motion model, cross-correlation between the returns of different assets is independent of the sampling interval. However, due to Epps effect, cross-correlation is significantly reduced when the sampling interval is less than a minute. Moreover, certain abrupt changes, i.e., jumps are observed in asset prices. Therefore estimators that take Epps effect and jumps into account work better.

CHAPTER 6

PORTFOLIO RISK ANALYSIS AND MANAGEMENT

Portfolio is a collection of investments in various financial assets. Two important aspects of a portfolio are its return and risk. The main goal of a portfolio manager is to keep the ratio of portfolio return over portfolio risk as high as possible. In order to estimate the risk of a portfolio with N assets, $N(N-1)/2$ unknown cross-correlations of asset returns need to be estimated to create the $N \times N$ empirical financial correlation matrix, $\hat{\mathbf{P}}$. However, $\hat{\mathbf{P}}$ contains significant amount of inherent noise that needs to be removed. Karhunen-Loève transform (KLT) has been successfully employed to filter out this undesirable noise component from the measured correlations [18, 19, 20, 64]. The caveat is the computational cost of KLT operations.

In this chapter, KLT based noise filtering of $\hat{\mathbf{P}}$ for better risk analysis, followed by risk estimation for the case of hedged portfolios is studied in detail. Next, it is argued that in contrast to the traditional view, an investor may rebalance the portfolio (change the investment amount in each asset), measure and manage the risk at different sampling time intervals or time resolutions. Therefore, a novel extension of the traditional risk metric for trading in multiple frequencies is introduced. Then, high performance filtering of $\hat{\mathbf{P}}$ via KLT is discussed. Performance improvement is achieved by approximating $\hat{\mathbf{P}}$ with a Toeplitz matrix structure such that efficient kernel discussed in Chapter 3 can be used. Moreover, discrete cosine transform (DCT) as an approximation to KLT is discussed. Corresponding approximation errors are discussed. Finally, a straightforward risk management method and two novel modifications to it are presented. Performance improvement of the modifications as well as the multiple frequency rebalancing and risk analysis concept are reported via back-testing at the end of the chapter.

6.1 Eigenfiltering of the Noise in the Empirical Financial Correlation Matrix

In this section, intrinsic noise in the empirical financial correlation matrix and its eigenfiltering via KLT is discussed. The concept of eigenfiltering is coupled with random matrix theory which is discussed first. After discussing the merit of KLT in this specific problem, a novel extension to the case of a hedged portfolio is discussed.

6.1.1 Asymptotic Distribution of the Eigenvalues of a Random Matrix

A random matrix $\mathbf{K}$ of size $N \times N$ is constructed as

$$\mathbf{K} = \frac{1}{M}\mathbf{W}^{\mathrm{T}}\mathbf{W}, \tag{6.1}$$

where $\mathbf{W}$ is an $M \times N$ matrix comprised of uncorrelated elements drawn from a Gaussian distribution with zero mean and variance σ^2, i.e., $[W_{mn}] \sim \mathcal{N}(0, \sigma^2)$, for $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. It is noted that $\mathbf{K}$ belongs to the family of Wishart matrices as referred in the multivariate statistical theory. Statistics of random matrices such as $\mathbf{K}$ are extensively studied in the literature [20]. It was shown that the distribution of the eigenvalues of the random matrix $\mathbf{K}$ in the limit is formulated as [68]

$$f(\lambda) = \frac{M}{2\pi\sigma^2 N}\frac{\sqrt{(\lambda_{max} - \lambda)\,(\lambda - \lambda_{min})}}{\lambda}, \tag{6.2}$$

where $f(\cdot)$ is the probability density function, $N \to \infty$, $M \to \infty$ with the ratio M/N fixed, and λ_{max} and λ_{min} are the maximum and minimum eigenvalues of $\mathbf{K}$, respectively in the same limiting case, with the values defined as[68]

$$\lambda_{max,min} = \sigma^2\left(1 + \frac{N}{M} \pm 2\sqrt{\frac{N}{M}}\right). \tag{6.3}$$

6.1.2 Noise in the Empirical Financial Correlation Matrix

Financial correlation matrix $\mathbf{P}$ as defined in (5.34) is estimated by using historical data. Sample correlation matrix can be used as an estimate of $\mathbf{P}$ given as

$$\hat{\mathbf{P}} = \frac{1}{M}\bar{\mathbf{R}}^{\mathrm{T}}\bar{\mathbf{R}}, \tag{6.4}$$

where $\bar{\mathbf{R}}$ is the $M \times N$ asset return matrix, N is the number of assets in the portfolio, and M is the number of available return samples per asset. Each element of $\bar{\mathbf{R}}$, i.e., $\bar{R}_{mn}$, is the normalized return of the n*th* asset at the m*th* discrete-time instance. Normalization is done such that the return time series of each asset, i.e., each column of $\bar{\mathbf{R}}$, is zero mean and unit variance. The element located on the i*th* row and j*th* column of $\hat{\mathbf{P}}$ is equal to

$$\hat{P}_{ij} = \frac{1}{M} \sum_{m=0}^{M-1} \bar{r}_i(m) \bar{r}_j(m), \tag{6.5}$$

where $\bar{r}_i(m) = \bar{R}_{mi}$ is the normalized return of the i*th* asset at the m*th* discrete-time instance as defined

$$\bar{r}_i(m) = \bar{R}_{mi} = \frac{r_i(m) - \hat{\mu}_i}{\hat{\sigma}_i}, \tag{6.6}$$

where $r_i(m)$ is the return of the i*th* asset at the m*th* discrete-time, $\hat{\mu}_i$ is the estimated mean of $r_i(m)$ as defined

$$\hat{\mu}_i = \frac{1}{M} \sum_{m=0}^{M-1} r_i(m), \tag{6.7}$$

and $\hat{\sigma}_i$ is the estimated standard deviation of $r_i(m)$ as defined

$$\hat{\sigma}_i = \left(\frac{1}{M} \sum_{m=0}^{M-1} \left[r_i(m) - \hat{\mu}_i \right]^2 \right)^{1/2}. \tag{6.8}$$

If the return processes were stationary, choosing M as large as possible would be the best approach to improve the estimation. However, in financial processes, anything is hardly stationary. Second, some assets may not have long history. For example, centuries-long historical data for cotton price might be available whereas Internet-based investment instruments have been around for only about fifteen years. Third, an investor might want to exploit the short-term impacts of certain crisis periods where choosing a long estimation time window may wipe out time local events. Therefore, choosing the observation time window, M, is not a trivial task, and its value depends on the application scenario under consideration.

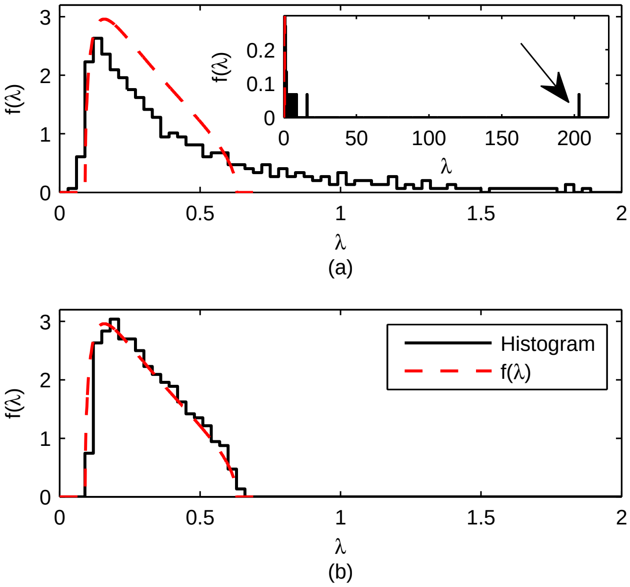


Figure 6.1 (a) Histogram of the eigenvalues of the empirical financial correlation matrix (black) along with the limiting p.d.f. of the eigenvalues of a random matrix (6.2) (red). (b) Histogram of the eigenvalues of an empirical random matrix (6.1) along with the limiting p.d.f.

If $\hat{\mathbf{P}}$ were a random matrix, its eigenvalues would be samples drawn from the distribution given in (6.2). In order to be able to check whether $\hat{\mathbf{P}}$ is consistent with this distribution or not, a financial correlation matrix $\hat{\mathbf{P}}$ is estimated using (6.4). In this example, the time interval of return data is chosen to be 15 minutes, and the time window for the estimation is defined between January 4, 2010 and May 18, 2010. Hence, $M = 2,444$ (94 business days and 26 data samples per day). Assets in the portfolio are the 494 of 500 stocks listed in S&P 500 index. Thus, $N = 494$ and $M/N = 4.95$. Matrix $\hat{\mathbf{P}}$ is decomposed into its eigenvectors with the corresponding eigenvalues and the histogram of its eigenvalues is calculated [18, 19]. The histogram of the eigenvalues of the estimated financial correlation matrix is displayed in Figure 6.1.a along with the probability density function of the eigenvalues of a random matrix expressed in (6.2), and calculated with $M/N = 4.95$, $\sigma^2 = 0.3$, $\mu = 0$, $\lambda_{max} = 0.63$, and $\lambda_{min} = 0.091$ according to (6.3). It is inferred from the figure, by setting the parameter $\sigma^2 = 0.3$ in (6.2), that a reasonable fit to the empirical data for eigenvalues smaller than 0.63 is achievable. This suggests that about 30% of the energy in matrix $\hat{\mathbf{P}}$ is random. Hence, eigenvalues smaller than 0.63 can be considered as noise. The largest 60 of the 494 eigenvalues, 13%, represent about 70% of the total variance in $\hat{\mathbf{P}}$. Moreover, only the largest four eigenvalues represent 50% of the total energy. The maximum and minimum eigenvalues, λ_{max}^{P} and λ_{min}^{P}, are equal to 203.24 and 0.044, respectively. The largest eigenvalue is approximately 322 times larger than its counterpart in a random matrix.

The histogram of the eigenvalues for the empirical financial correlation matrix, $\hat{\mathbf{P}}$, has two major clusters as displayed in Figure 6.1.a. Namely, there is a bulk of eigenvalues that are strongly related to the noise, and the remaining relatively small number of eigenvalues deviating from the bulk representing the valuable information. Since 95% of the eigenvalues are less than 2, the region bounded by $0 \leq \lambda \leq 2$ is zoomed and the full histogram is shown on the top right corner in in Figure 6.1.a. The largest eigenvalue, 203.24, is marked with an arrow.

For the purpose of validation, a random matrix $\mathbf{K}$ of (6.1) is generated with the parameters $M/N = 4.95$, $\sigma^2 = 0.3$, and $\mu = 0$. The histogram of the eigenvalues of $\mathbf{K}$ along with the probability density function of the eigenvalues for a random matrix as defined in (6.2) using the same parameters is displayed in Figure 6.1.b. It is observed from the figure that empirical histogram fits quite nicely to the theoretical limiting distribution.

6.1.3 Eigenfiltering of the Noise

In this section, KLT is employed to filter out the noise component of the empirical financial correlation matrix. Then, filtered correlation matrix is utilized in the calculation of portfolio risk, (5.32), and its effects on portfolio optimization and performance are emphasized. The empirical financial correlation matrix given in (6.4) is decomposed into its eigenvalues and eigenvectors as follows

$$\hat{\mathbf{P}} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}, \tag{6.9}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_k)$ is a diagonal matrix with the eigenvalues as its elements, λ_k is the k*th* eigenvalue with the order $\lambda_k \geq \lambda_{k+1}$, $\mathbf{\Phi}$ is an $N \times N$ matrix comprised of N eigenvectors as its columns as given

$$\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_N \end{bmatrix}, \tag{6.10}$$

and $\boldsymbol{\phi}_k$ is the $N \times 1$ eigenvector corresponding to the k*th* eigenvalue, λ_k with $\lambda_k \geq 0 \ \forall_k$, $\sum_k \lambda_k = N$. Moreover, $\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{I}$ due to orthonormality property of the eigenvectors. By substituting (6.9) into (5.32) estimated risk of the portfolio can be formulated as

$$\hat{\sigma}_p = \left(\mathbf{q}^{\mathrm{T}}\mathbf{\Sigma}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}\mathbf{q}\right)^{1/2}. \tag{6.11}$$

Following the approach proposed in [19], eigenfiltered financial correlation matrix is defined as

$$\tilde{\mathbf{P}} = \sum_{k=1}^{L} \lambda_k \boldsymbol{\phi}_k \boldsymbol{\phi}_k^{\mathrm{T}} + \mathbf{E}, \tag{6.12}$$

where L is the number of selected factors (eigenvalues) satisfying $\lambda_k \geq \lambda_{max}$ (6.3), and $L \ll N$. Identifying L is not a trivial task. Although (6.3) offers a framework to calculate it, in practice, investors might use back-testing to determine the L that provides the best performance. The diagonal noise matrix $\mathbf{E}$ is introduced in (6.12) in order to preserve the total variance in the calculations. Matrix $\mathbf{E}$ is defined as

$$\mathbf{E} = [E_{ij}] = \varepsilon_{ij} = \begin{cases} 1 - \sum_{k=1}^{L} \lambda_k \phi_i^{(k)} \phi_j^{(k)} & i = j \\ 0 & i \neq j \end{cases}, \tag{6.13}$$

where $\phi_i^{(k)}$ is the i*th* element of the k*th* eigenvector. The addition of matrix $\mathbf{E}$ is equivalent to setting the diagonal elements to 1, i.e., $\left[\tilde{P}_{ii}\right] = 1$, and it is required to keep the trace of $\tilde{\mathbf{P}}$ (6.12) equal to the one of $\hat{\mathbf{P}}$ (6.9). From (6.12) and (6.13) noise filtered financial correlation matrix can be rewritten as follows

$$\tilde{\mathbf{P}} = \left[\tilde{P}_{ij}\right] = \tilde{\rho}_{ij} = \sum_{k=1}^{L} \lambda_k \phi_i^{(k)} \phi_j^{(k)} + \varepsilon_{ij}. \tag{6.14}$$

By substituting the filtered version of ρ_{ij}, (6.14), into (5.32), estimated risk via filtered empirical correlation matrix is obtained as given

$$\tilde{\sigma}_p = \left[\sum_{k=1}^{L} \lambda_k \left(\sum_{i=1}^{N} q_i \phi_i^{(k)} \sigma_i\right)^2 + \sum_{i=1}^{N} \varepsilon_{ii} q_i^2 \sigma_i^2\right]^{1/2}. \tag{6.15}$$

Following similar steps provided in [18, 19], impact of using noise filtered estimated risk (6.15) rather than noisy estimated risk (5.32) is studied next with an example. The data set consists of return time series for 494 stocks listed in S&P 500 index, and there are two time periods. First and second time periods include the business days from January 4, 2010 to May 18, 2010, and from May 19, 2010 to September 30, 2010, respectively. In the example, it is assumed that on the morning of May 19, 2010, i.e., the first day of the second time period, a risk manager has the perfect prediction of the future returns; i.e., the expected return vector of the second period, $\boldsymbol{\mu}_2$ of (5.31), is known. On that morning, the risk manager is asked to create a portfolio for a given target portfolio return, μ. First, the risk manager calculates the sample correlation matrix for the first time period, $\hat{\mathbf{P}}_1$, using (6.4). Then, the manager obtains the investment vector, $\mathbf{q}^*$, using using (5.38) with $\hat{\mathbf{P}}_1$, $\boldsymbol{\mu}_2$, and μ. Next, estimation of the predicted risk of the portfolio for the second time period, $\hat{\sigma}_2^p$, using (5.32) with $\mathbf{q}^*$ and $\hat{\mathbf{P}}_1$ is calculated. Finally, on September 30, 2010, i.e., the last day of the second time period, the empirical correlation matrix for the second time period, $\hat{\mathbf{P}}_2$, and estimation of the realized risk of the portfolio for the second time period, $\hat{\sigma}_2^r$, using (5.32) with $\mathbf{q}^*$ and $\hat{\mathbf{P}}_2$ are found. Obviously, in this setup, the investment vector calculated via (5.38) is a function of the target portfolio return, μ, and the covariance matrix, $\mathbf{C} = \mathbf{\Sigma}^{\mathrm{T}}\mathbf{P}\mathbf{\Sigma}$. Moreover, portfolio risk calculated via (5.32) is a function of the investment vector, $\mathbf{q}^*$, and the covariance matrix, $\mathbf{C}$. Hence, estimated portfolio risk is a function of the target portfolio return, μ, and the correlation matrix, $\mathbf{P}$. It is formalized as follows

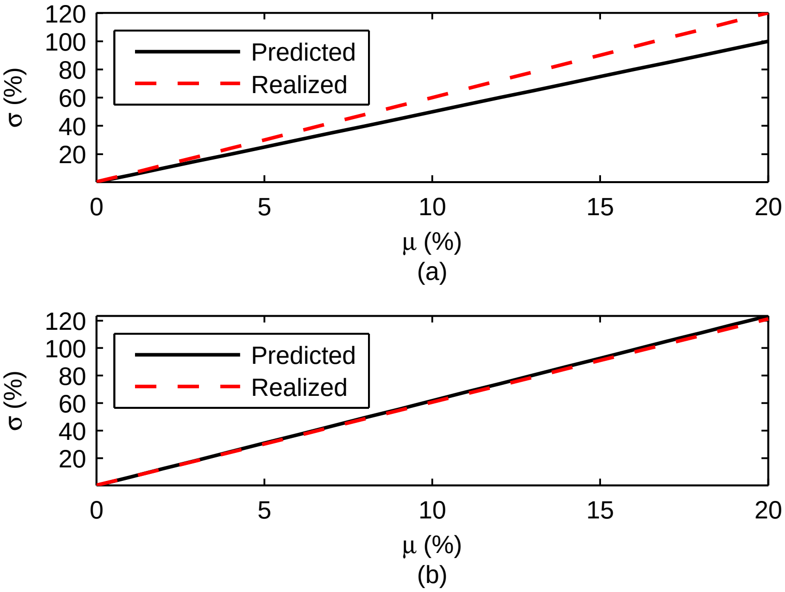


Figure 6.2 Predicted and realized risk functions versus target portfolio returns for the set of efficient portfolios where noisy empirical financial correlation matrices are used (a) without any filtering, (b) with filtering, prior to risk calculation.

$$\begin{aligned} \hat{\sigma}_2^p(\mu) &= f\left(\hat{\mathbf{P}}_1, \mu\right) \\ \hat{\sigma}_2^r(\mu) &= f\left(\hat{\mathbf{P}}_2, \mu\right), \end{aligned} \tag{6.16}$$

where $f(\cdot)$ defines a function that involves the calculation of the optimum investment vector defined in (5.38), and using it in (5.32) in order to obtain the portfolio risk. Plots of $\hat{\sigma}_2^p(\mu)$ and $\hat{\sigma}_2^r(\mu)$ as a function of μ are given in Figure 6.2.a with black-solid and red-dashed lines, respectively. It is also possible for the risk manager to check how large the risk was under- or over-estimated by defining an error function as given

$$\epsilon(\mu) = \left(\frac{\hat{\sigma}_2^p(\mu)}{\hat{\sigma}_2^r(\mu)} - 1\right) \times 100\%. \tag{6.17}$$

Root mean square (RMS) value of this error function is ~16.8% for the case displayed in Figure 6.2.a. In the second part of the example, risk manager does everything same but this time, the filtered empirical correlation matrices, $\tilde{\mathbf{P}}_1$ and $\tilde{\mathbf{P}}_2$, calculated via (6.12) with $L = 4$ are used, prior to creating the portfolio and calculating the corresponding risk functions, $\hat{\sigma}_2^p(\mu)$ and $\hat{\sigma}_2^r(\mu)$. The results for this case are given in Figure 6.2.b. The RMS value of the error defined in (6.17) is ~2.3%. It is observed that noise-free correlation matrix is more stable and it lets the risk manager predict the portfolio risk better. Although the example given in this section compares only two time periods and it is not 100% realistic due to the assumption that the future expected return vector, $\boldsymbol{\mu}_2$, is known, it is sufficient to emphasize the significance of built-in noise in the correlation matrix and the importance of filtering it out. In practice, the parameters of the system, such as the number of eigenvalues, L, must be back-tested over several time periods to build a level of confidence.

6.1.4 Eigenfiltering of the Noise for a Hedged Portfolio

In a hedged portfolio, every asset is associated with a hedging asset such that every investment decision is made for a pair. Hedging asset may not necessarily be from the same porftolio. The simplest method to determine the hedge amount is to use the regression defined in (5.41). Return of a hedged portfolio comprised of N assets and H hedging assets is given as

$$r_p = r_a + r_h = \sum_{i=1}^{N} q_i r_i + \sum_{j=1}^{H} g_j y_j, \tag{6.18}$$

where r_a and r_h are the total returns on the investment for assets and hedging assets, respectively; r_i and y_j are the returns of the i*th* asset and the j*th* hedging asset, respectively; and q_i and g_j are the amounts of capital invested in the i*th* asset and the j*th* hedging asset, respectively. In the rest of the section, for the clarity of the discussion, it is assumed that the returns of the assets have zero mean noting that extension to non-zero mean case is trivial.

The risk of the hedged portfolio is expressed as

$$\begin{aligned}\sigma_p &= E\left\{r_p^2\right\}^{1/2} \\ &= \left(\sigma_a^2 + 2\sigma_a\sigma_h + \sigma_h^2\right)^{1/2} \\ &= \left(E\left\{r_a^2\right\} + 2E\left\{r_a r_h\right\} + E\left\{r_h^2\right\}\right)^{1/2}. \end{aligned} \tag{6.19}$$

It follows from (6.18) and (6.19) that one needs to estimate cross-correlations of asset returns in order to measure the risk as follows

$$\sigma_a^2 = E\left\{r_a^2\right\} = \sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j E\left\{r_i r_j\right\}, \tag{6.20}$$

and the cross-correlations between returns of assets and hedging assets written as

$$\sigma_a\sigma_h = E\left\{r_a r_h\right\} = \sum_{i=1}^{N}\sum_{j=1}^{H} q_i g_j E\left\{r_i y_j\right\}. \tag{6.21}$$

Similarly, cross-correlations of hedging asset returns defined as

$$\sigma_h^2 = E\left\{r_h^2\right\} = \sum_{i=1}^{H}\sum_{j=1}^{H} g_i g_j E\left\{y_i y_j\right\}. \tag{6.22}$$

Steps involved in deriving the eigenfiltered version of (6.20) are similar to the ones in (6.15). Hence, eigenfiltered version of (6.20) is expressed as

$$\tilde{\sigma}_a^2 = \sum_{k=1}^{L} \lambda_k \left(\sum_{i=1}^{N} q_i \phi_i^{(k)} \sigma_i\right)^2 + \sum_{i=1}^{N} \varepsilon_{ii} q_i^2 \sigma_i^2, \tag{6.23}$$

where λ_k is the k*th* eigenvalue, L is the number of selected factors (eigenvalues), $\phi_i^{(k)}$ is the i*th* element of the eigenvector corresponding to the k*th* eigenvalue, σ_i is the volatility of the i*th* asset, and ε_{ii} is the error term defined in (6.13). Return of the j*th* hedging asset can be modeled as a weighted sum of the assets in the portfolio as follows

$$y_j = \sum_{i=1}^{N} \gamma_{j,i} r_i + \xi_j. \tag{6.24}$$

In practice, the number of observations is limited to M. Then, (6.24) can be written in matrix from as given

$$\mathbf{y}_j = \mathbf{R}\boldsymbol{\gamma}_j + \boldsymbol{\xi}_j, \tag{6.25}$$

where $\mathbf{y}_j$ is the $M \times 1$ return vector for the j*th* hedging asset, $\mathbf{R}$ is the $M \times N$ matrix of asset returns, $\boldsymbol{\xi}_j$ is the $M \times 1$ error vector, and $\boldsymbol{\gamma}_j$ is the $N \times 1$ vector of regression coefficients. By using the eigenanalysis of the empirical financial correlation matrix given in (6.9), the fact that $\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \mathbf{I}$, and $\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma} = \mathbf{I}$, (6.25) can be rewritten as

$$\begin{aligned} \mathbf{y}_j &= \mathbf{R}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\gamma}_j + \boldsymbol{\xi}_j \\ &= \mathbf{F}\boldsymbol{\beta}_j + \boldsymbol{\xi}_j, \end{aligned} \tag{6.26}$$

where $\mathbf{F} = \mathbf{R}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}$ is the $M \times N$ principal components matrix [7] with its elements F_{nk} being the n*th* sample value of the k*th* principal component which is given as

$$F_k = \sum_{i=1}^{N} \frac{1}{\sigma_i} r_i \phi_i^{(k)}. \tag{6.27}$$

It is noted that the correlation between two different principal components is zero, and the variance of a particular principal component is equal to its corresponding eigenvalue, i.e.,

$$E\left\{F_i F_j\right\} = \begin{cases} \lambda_i & i = j \\ 0 & i \neq j \end{cases}. \tag{6.28}$$

Regression coefficient vector $\boldsymbol{\beta}_j$ given in (6.26) can be estimated via least-squares algorithm as follows

$$\hat{\boldsymbol{\beta}}_j = \left(\mathbf{F}^{\mathrm{T}}\mathbf{F}\right)^{-1} \mathbf{F}^{\mathrm{T}}\mathbf{y}_j. \tag{6.29}$$

It follows from (6.26) that one can regress the return of a hedging-asset asset y_j over L principal components as follows

$$y_j = \sum_{k=1}^{L} \beta_{j,k} F_k + \zeta_j, \tag{6.30}$$

where $\beta_{j,k}$ is the regression coefficient for the j*th* hedging asset and k*th* principal component, and ζ_j is the residual term. From (6.26) and (6.30) it follows that

$$\zeta_j = \sum_{k=L+1}^{N} \beta_{j,k} F_k + \xi_j. \tag{6.31}$$

By assuming that the residual term, ζ_j, is orthogonal to all principal components and substituting (6.30) in (6.22), eigenfiltered version of (6.22) is written as

$$\tilde{\sigma}_h^2 = \sum_{i=1}^{H} \sum_{j=1}^{H} g_i g_j \left(\sum_{k=1}^{L} \sum_{l=1}^{L} \beta_{i,k} \beta_{j,l} E\left\{F_k F_l\right\} + E\left\{\zeta_i \zeta_j\right\} \right). \tag{6.32}$$

It follows from (6.28) that

$$\tilde{\sigma}_h^2 = \sum_{i=1}^{H} \sum_{j=1}^{H} g_i g_j \left(\sum_{k=1}^{L} \lambda_k \beta_{i,k} \beta_{j,k} + E\left\{\zeta_i \zeta_j\right\} \right). \tag{6.33}$$

Assuming the cross-correlation between residual terms is zero, i.e., $E\left\{\zeta_i \zeta_j\right\} = 0$ for $i \neq j$, (6.33) can be rewritten as

$$\tilde{\sigma}_h^2 = \sum_{k=1}^{L} \lambda_k \left(\sum_{j=1}^{H} g_j \beta_{j,k} \right)^2 + \sum_{i=1}^{H} g_i^2 \nu_i^2, \tag{6.34}$$

where $\nu_i^2 = \mathrm{var}\left\{\zeta_i\right\}$.

In a similar fashion, assuming that there is no correlation between asset returns and residual terms, i.e., $E\left\{r_i \zeta_j\right\} = 0$, the eigenfiltered version of the cross-correlation between the i*th* asset and the j*th* hedging asset of (6.21) is expressed using (6.30) as follows

$$\tilde{\sigma}_a \tilde{\sigma}_h = \sum_{k=1}^{L} \beta_{j,k} E\left\{r_i F_k\right\}. \tag{6.35}$$

Substituting (6.27) in (6.35), yields

$$\tilde{\sigma}_a \tilde{\sigma}_h = \sum_{k=1}^{L} \sum_{l=1}^{N} \beta_{j,k} \frac{\phi_l^{(k)}}{\sigma_l} E\left\{r_i r_l\right\}. \tag{6.36}$$

By substituting (6.14) in (6.36), due to the orthogonality of the eigenvectors, (6.36) becomes

$$\tilde{\sigma}_a\tilde{\sigma}_h = \sum_{k=1}^{L} \lambda_k \phi_i^{(k)} \beta_{j,k}\sigma_i. \tag{6.37}$$

Hence, the eigenfiltered version of (6.21) is equal to

$$\tilde{\sigma}_a\tilde{\sigma}_h = \sum_{k=1}^{L} \lambda_k \sum_{i=1}^{N}\sum_{j=1}^{H} q_i g_j \phi_i^{(k)} \beta_{j,k}\sigma_i. \tag{6.38}$$

Finally, the substitution of (6.23), (6.34), and (6.38) in (6.19), and re-arranging components yields the risk of a hedged portfolio with return given in (6.18) by using filtered version of the empirical correlation matrix as follows

$$\tilde{\sigma}_p = \left[\sum_{k=1}^{L} \lambda_k \left(\sum_{i=1}^{N} q_i \phi_i^{(k)} \sigma_i + \sum_{j=1}^{H} g_j \beta_{j,k} \right)^2 + \sum_{i=1}^{N} \varepsilon_{ii} q_i^2 \sigma_i^2 + \sum_{j=1}^{H} \nu_j^2 g_j^2 \right]^{1/2}, \tag{6.39}$$

where λ_k is the k*th* eigenvalue, $\phi_i^{(k)}$ is the i*th* element of the k*th* eigenvector, σ_i is the volatility of the asset, $\beta_{j,k}$ is the hedging factor for the j*th* hedging asset and k*th* principal component, and ν_j^2 is the variance of the idiosyncratic component of the returns for the j*th* hedging asset regressed on L principal components defined in (6.31).

6.2 Risk Estimation for Rebalancing in Multiple Frequencies

In finance, frequency in general means the speed of re-balancing a portfolio, i.e., changing the investment amounts invested in each asset in the portfolio. In most of the literature, it is assumed that the frequency of rebalancing for each asset is the same. However, rebalancing in multiple frequencies, i.e., rebalancing different assets at different times, may be desirable for the investors due to several reasons including the following:

1. Liquidity, i.e., the availability of the asset in the market, may not be the same for all the assets in the portfolio. Therefore, the investor may want to rebalance certain assets faster or slower than others.

2. Different assets may reveal certain aspects of the market the investor is looking for such as a trend, a relative-value etc., at different sampling frequencies.

3. A high frequency investor may want to keep the portfolio diverse and balanced in terms of risk, and the traditional methods for measuring and managing risk require relatively high correlations between the assets.

However, cross-correlation of asset returns is reduced as the sampling frequency increases due to the well-known phenomenon in finance called the Epps effect [23] as discussed in Section 5.4. In accordance with the Epps effect, correlations between the financial assets vary at different sampling frequencies. The number of eigenvalues, L, versus the percentage of the total variance represented is displayed in Figure 6.3 for different sampling intervals. It can be seen from the figure that, as the sampling interval decreases, eigenspectrum of the correlation matrix becomes more spread. Thus, more eigenvalues are required to represent a certain percentage of the total variance. EOD stands for end of day sampling rate, i.e., price data is sampled at the market closing of each day.

In order to accommodate the novel concept of rebalancing in multiple frequencies, the risk definition of (5.32) must be properly modified. Assuming that the prices follow a geometric Brownian motion, it follows from the discussions given in Section 5.2.2 that volatilities estimated at different sampling frequencies have the following relationship

$$\sigma_1 = \sqrt{k_1/k_2}\sigma_2 = m\sigma_2, \tag{6.40}$$

where σ_1 and σ_2 are the volatilities estimated at sampling intervals k_1T_s and k_2T_s, respectively, and T_s is the base sampling interval. Hence, it is possible to measure portfolio risk at a certain time interval, and manage risk of assets by re-balancing the individual investment

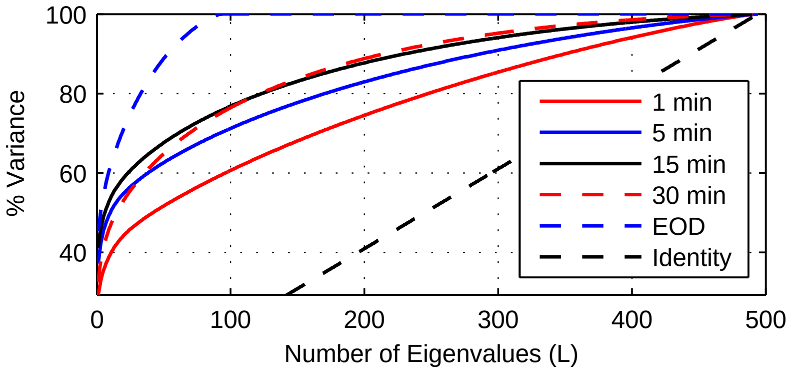


Figure 6.3 Scree plot displaying the number of eigenvalues versus percentage of the represented total variance for different sampling intervals and for the $\hat{\mathbf{P}} = \mathbf{I}$ case, i.e., no correlation between assets.

allocations at different time intervals, by expanding the original risk formula of (5.32) as follows

$$\sigma_p = \left(\mathbf{q}^{\mathrm{T}}\boldsymbol{\Sigma}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{P}\mathbf{M}\boldsymbol{\Sigma}\mathbf{q}\right)^{1/2}, \tag{6.41}$$

where $\mathbf{M} = \mathrm{diag}(m_1, m_2, \ldots, m_N)$ and m_i is the scaling factor of (6.40) provided that $\boldsymbol{\Sigma}$ and $\mathbf{P}$ matrices are estimated at $k_2 T_s$ time intervals. It is noted that, similar modification is applicable to the eigenfiltered risk formula given in (6.15). Performance improvement in using (6.41) instead of (5.32) is studied in Section 6.4.4.

6.3 High Performance Eigenfiltering for Risk Estimation

Implementation of KLT is costly. Therefore, in every engineering application, including the eigenfiltering discussed earlier in the chapter, fast implementation of KLT is desirable. In this section, a Toeplitz approximation to the empirical financial correlation matrix is proposed in order to be able to apply the explicit KLT kernel discussed in Chapter 3. Moreover, DCT as an approximation to the KLT is proposed in the same context since it is known that their kernels are very close for AR(1) signals with high first order correlation coefficient as it is discussed in Section 2.3.5.

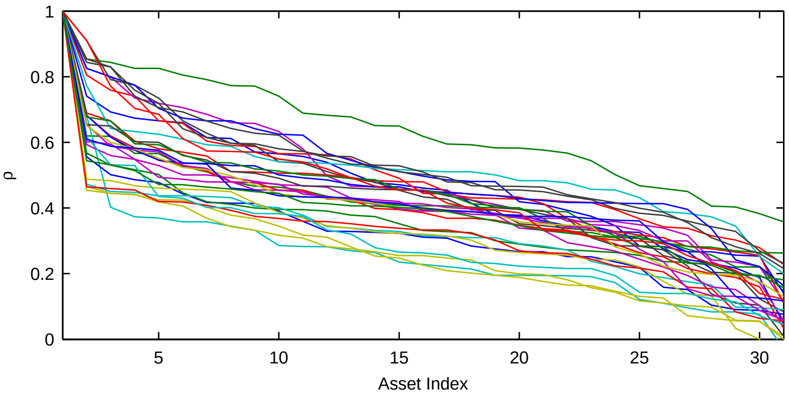


Figure 6.4 Rows of $\hat{\mathbf{P}}$ matrix of DJIA & DIA EOD returns displayed in descending order.

6.3.1 Toeplitz Approximation to the Empirical Financial Correlation Matrix

In this section, it is attempted to approximate empirical correlations of asset returns by utilizing AR(1) signal source model discussed in Section 2.1. Main motivation here is to incorporate the closed-form expressions of eigenvalues and eigenvectors of AR(1) sources as expressed in (3.3) and (3.2) that are utilized for eigenfiltering of empirical correlation matrix as described in the previous section accordingly.

For motivational purposes, a portfolio comprised of all 30 stocks of the index Dow Jones Industrial Average (DJIA) along with the exchange traded fund DIA that mimics DJIA (total of $N = 31$ assets) is considered. Empirical financial correlation matrix $\hat{\mathbf{P}}$ defined in (6.4) for this specific portfolio is calculated using the end of day returns (EOD) for 60 business days, i.e., $M = 60$. Time span considered is from March 17 to June 10, 2011.

Figure 6.4 displays the elements of $\hat{\mathbf{P}}$ for each row in a descending order for the 31 assets. The k*th* sequence represents pairwise correlations of the k*th* asset with all assets in the portfolio. It can be observed from the figure that an AR(1) source is a good candidate to approximate the sequences given in Figure 6.4. In this section, two cases where AR(1) signal model with Toeplitz correlation matrix is employed to approximate symmetric matrix are considered. Effects of approximations proposed in the section in the calculation of the eigenfiltered risk (6.15) are discussed at the end of the section.

6.3.1.1 AR(1) Approximation to Financial Correlation Matrix. Empirical correlation matrix defined in (6.4) is symmetric. It is approximated by a Toeplitz matrix as follows

$$\check{\mathbf{P}} = \begin{bmatrix} 1 & \rho_{opt} & \cdots & \rho_{opt}^{N-1} \\ \rho_{opt} & 1 & \cdots & \rho_{opt}^{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{opt}^{N-1} & \rho_{opt}^{N-2} & \cdots & 1 \end{bmatrix}, \tag{6.42}$$

where N is the number of assets in the portfolio and ρ_{opt} is the optimal correlation coefficient of AR(1) source which is found by minimizing the approximation error defined as

$$\varepsilon = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\hat{P}_{ij} - \rho_{opt}^{|i-j|} \right)^2, \tag{6.43}$$

where $\hat{P}_{ij}$ is the element of the empirical financial correlation matrix located on the i*th* row and j*th* column. One can calculate the resulting eigenvalues and eigenvectors of AR(1) model according to (3.3) and (3.2) as approximations to their measured values, respectively, in order to speed up the eigenfiltered estimation of the risk defined in (6.15) and (6.39).

Figure 6.5 displays variations of correlation coefficient ρ_{opt} for 31 assets of DJIA & DIA under consideration along with approximation errors of (6.43). The returns are calculated for 24 hour intervals with sliding time intervals of 15 minutes and measurement window of $M = 60$ business days for a trading day of 6.5 hours. Specifically, a total of 27 return series of length 60 are created. Each return series is calculated by sampling the price series at a specific time on every business day. For example, the first and last return series are calculated by sampling the price at 9:30 and 16:00, respectively, everyday. Therefore, the last sample on Figure 6.5 corresponds to end of day (EOD) return of an asset. Figure 6.5 shows highly correlated nature of EOD and 24 hour returns.

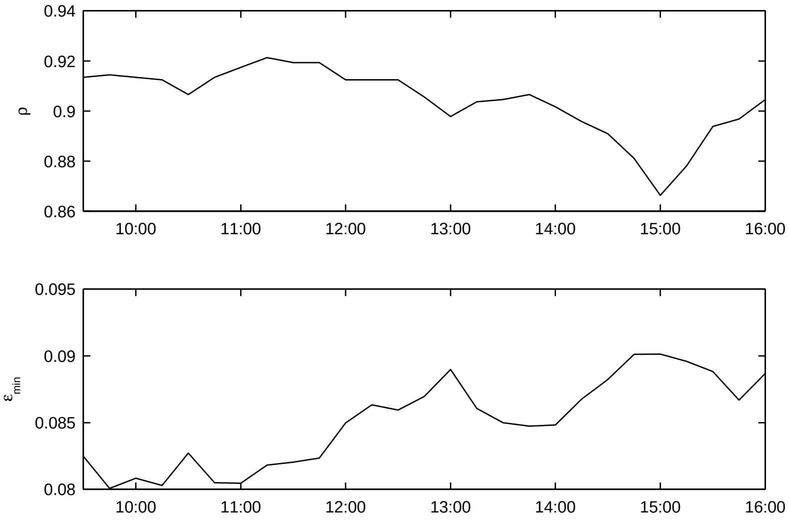


Figure 6.5 Variations of optimal correlation coefficient and the resulting error of AR(1) approximation, (6.43), as a function of time with 15 minute sliding intervals with $M = 60$ business days for 24 hour returns of 31-asset portfolio (DJIA & DIA) in the interval 9:30-16:00.

6.3.1.2 AR(1) Approximation to Each Row of Financial Correlation Matrix.

Each row of empirical correlation matrix is approximated by the optimal correlation sequence of AR(1) signal model with the correlation coefficients $\{\rho_{k,opt}\}$. Hence, the rows are approximated as

$$\check{\mathbf{P}} = \begin{bmatrix} 1 & \rho_{1,opt} & \cdots & \rho_{1,opt}^{N-1} \\ \rho_{2,opt} & 1 & \cdots & \rho_{2,opt}^{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N,opt}^{N-1} & \rho_{N,opt}^{N-2} & \cdots & 1 \end{bmatrix}, \tag{6.44}$$

where the optimum $\rho_{k,opt}$ for the k*th* row of $\check{\mathbf{P}}$ is obtained by minimizing the approximation error as defined

$$\varepsilon_k = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{P}_{ki} - \check{P}_{ki}\right)^2, \tag{6.45}$$

and, $\check{P}_{ki}$ is the element of matrix $\check{\mathbf{P}}$ located at the k*th* row and i*th* column. Then, each row is AR(1) approximated independently and (6.44) is rewritten as

$$\check{\mathbf{P}} = \sum_{k=1}^{N} \mathbf{S}_k \check{\mathbf{P}}_k, \tag{6.46}$$

where the selection matrix $\mathbf{S}_k$ is defined as

$$\mathbf{S}_k \triangleq \begin{cases} s_{k,k} = 1 & \text{for } k \\ 0 & \text{otherwise} \end{cases} ; \; k = 1, 2, \ldots, N, \tag{6.47}$$

and, the $\check{\mathbf{P}}_k$ matrix is a Toeplitz matrix as expressed

$$\check{\mathbf{P}}_k = \begin{bmatrix} 1 & \rho_{k,opt} & \cdots & \rho_{k,opt}^{N-1} \\ \rho_{k,opt} & 1 & \cdots & \rho_{k,opt}^{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{k,opt}^{N-1} & \rho_{k,opt}^{N-2} & \cdots & 1 \end{bmatrix}, \tag{6.48}$$

for $k = 1, 2, \ldots, N$. It is possible to decompose $\check{\mathbf{P}}_k$ into its eigenvalues and eigenvectors via eigenanalysis as follows

$$\check{\mathbf{P}}_k = \mathbf{A}_{KLT,k}^{\mathrm{T}} \mathbf{\Lambda}_k \mathbf{A}_{KLT,k}; \; k = 1, 2, \ldots, N. \tag{6.49}$$

Therefore, the Toeplitz approximation of (6.46) can be rewritten as

$$\check{\mathbf{P}} = \sum_{k=1}^{N} \mathbf{S}_k \mathbf{A}_{KLT,k}^{\mathrm{T}} \mathbf{\Lambda}_k \mathbf{A}_{KLT,k}, \tag{6.50}$$

where $\mathbf{A}_{KLT,k}$ and $\mathbf{\Lambda}_k$ are comprised of the k*th* set of eigenvectors and eigenvalues, respectively, which can be calculated with the closed-form expressions of (3.3) and (3.2) for the given set of AR(1) correlation coefficients $\{\rho_{k,opt}\}$ using the method described in Chapter 3. Figure 6.6 displays variations of correlation coefficients and resulting approximation errors of this method for the 31 assets under consideration. Similar to the previous case, returns are measured for 24 hour intervals with sliding time intervals of 15 minutes. It is noted

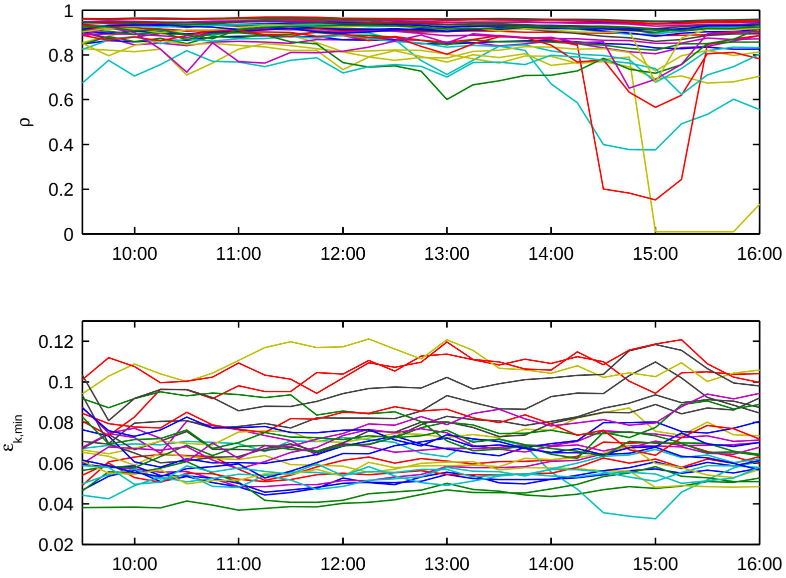


Figure 6.6 Variations of optimal correlation coefficients and the resulting errors of AR(1) approximations as a function of time with 15 minute sliding intervals for 24 hour returns of 31-asset portfolio (DJIA & DIA) with $M = 60$ days in the interval 9:30-16:00.

that the approximation error of this method is lower than the one for the previous case. The trade-off is the increased computational cost of the multiple Toeplitz approximations.

6.3.1.3 Effects of Toeplitz Approximations on Risk Estimation. In order to test the effects of approximation to the eigenfiltered risk measurement given in (6.15), a portfolio comprised of all 30 stocks of the DJIA index along with the DIA ETF is formed. The capital allocation vector $\mathbf{q}$ (5.29) is formed in accordance with the market cap of the assets in the portfolio as of June 10, 2011. The elements of $\mathbf{q}$ sum to 1 and the highest element corresponds to the asset with the highest market cap. Empirical financial correlation matrix $\hat{\mathbf{P}}$ defined in (6.4), and its approximations defined in (6.42) and (6.44) are calculated as a function of time with 15 minute sliding intervals and a measurement window of $M = 60$ business days for 24 hour returns. Time span considered is from March 17 to June 10, 2011. For each case, factors that correspond to the first largest five eigenvalues are kept, i.e., $L = 5$ in (6.15). For the cases where Toeplitz approximations are used, fast techniques discussed in Chapter 3 are used. Estimation of the eigenfiltered portfolio risk

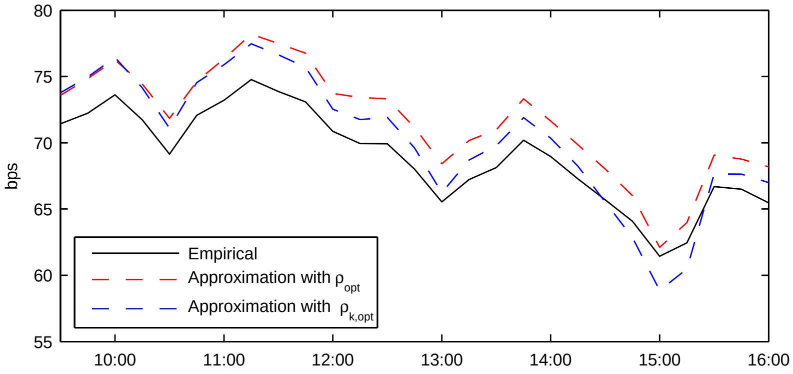


Figure 6.7 Portfolio risk calculated via (6.15) with empirical financial correlation matrix $\hat{\mathbf{P}}$, and its Toeplitz approximations $\check{\mathbf{P}}$ of (6.42), and $\check{\mathbf{P}}$ of (6.44) as a function of time with 15 minute sliding intervals for 24 hour returns and $M = 60$ business days of 31-asset portfolio (DJIA & DIA) in the interval 9:30-16:00.

for each case is displayed in Figure 6.7. The maximum distance in risk estimation from the case where empirical correlation matrix is used is negligible and is equal to +3.67 bps. It can be observed from Figure 6.7 that the risk estimations calculated using the Toeplitz approximated correlation matrices have the same proxy to the one in which the estimated correlation matrix itself is used.

6.3.2 Filtering the Noise with Discrete Cosine Transform

Due to the reasons discussed in Section 2.3.5, DCT as an approximation to KLT is preferred in most applications in which the correlation is significantly high. In this section, performances of fixed transform DCT and input dependent KLT for empirical correlation matrices of various portfolios are compared in order to justify the use of the former as an efficient replacement to the latter in filtering of the noise in the empirical correlation matrix as given in (6.14).

The histogram for correlation coefficients of Figure 6.6 is shown in Figure 6.8. The resulting mean and variance values are 0.8756 and 0.0125, respectively. These results coupled with the KLT and DCT performance comparisons displayed in Figure 2.1 and closeness of eigenvalues with the DCT coefficients for the empirical correlation matrix

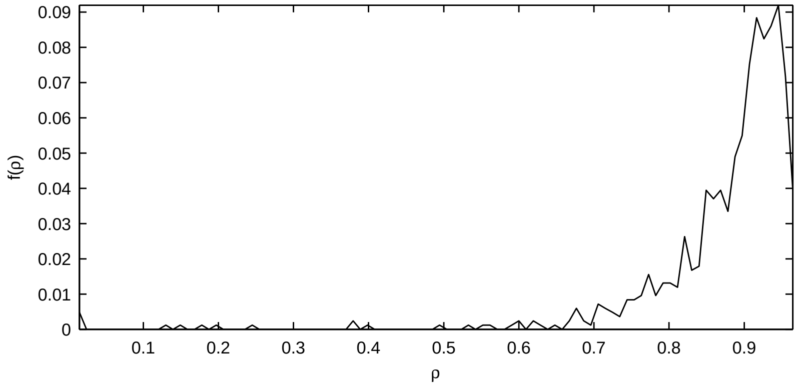


Figure 6.8 Histogram of correlation coefficients displayed in Figure 6.6

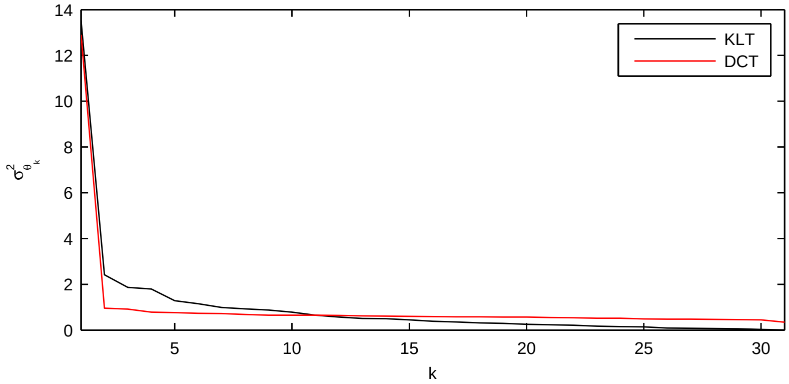


Figure 6.9 KLT and DCT coefficient variances for $\hat{\mathbf{P}}$.

displayed in Figure 6.9 suggest the use of DCT as a fast KLT approximation in calculating the filtered risk according to (6.15) and (6.39).

Same approach discussed in the previous section is employed to test the effects of using DCT as an approximation to KLT in risk estimation. Figure 6.10 compares portfolio risks of 24 hour returns of 31-asset portfolio (DJIA & DIA) calculated via (6.15) employing KLT and DCT filtering methods with five and ten factors, i.e., $L = 5$ and $L = 10$, respectively, as a function of time for 15 minute sliding intervals in a given 6.5 hours long trading day. Time span considered is 60 business days from March 17 to June 10, 2011. It is observed from the figure that KLT and DCT perform similarly for the filtering of empirical correlation matrices of asset returns experimented.

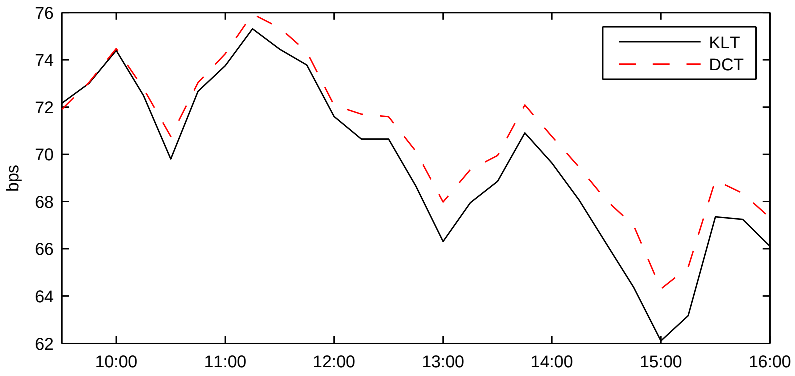
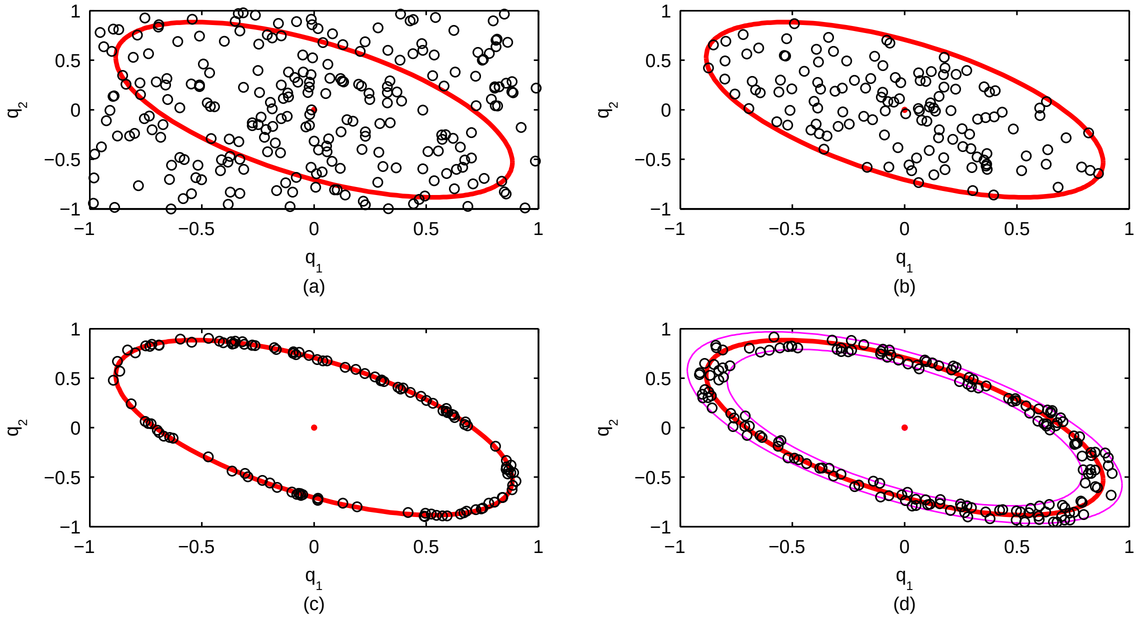


Figure 6.10 Portfolio risk calculated via (6.15) with filtered financial correlation matrix $\tilde{\mathbf{P}}$ (6.14) as a function of time with 15 minute sliding intervals for 24 hour returns and $M = 60$ business days of 31-asset portfolio (DJIA & DIA) in the interval 9:30-16:00. Filtering is done using KLT basis functions (eigenvectors) and DCT basis functions with $L = 5$ and $L = 10$ in (6.15), respectively.

6.4 Risk Management

Once the risk of a portfolio is estimated using the methods discussed so far in the chapter, it needs to be managed. As in the case of modern portfolio theory discussed in Section 5.3.1, in some applications, risk management is embedded in the investment strategy itself. However, in a real world scenario, an independent investment strategy constantly rebalances a given portfolio and its details are not necessarily known by the risk manager. The trivial method for risk management is to manage the portfolio risk by filtering the decisions of the underlying investment strategy based on a pre-determined risk limit.

The locus of q_i, $1 \leq i \leq N$ satisfying (5.32) for a fixed value of risk σ_p is an ellipsoid centered at the origin. The ellipsoids are nested since as σ_p increases, the ellipsoids become larger. The shape of the ellipsoid is defined by the asset return correlation matrix $\mathbf{P}$. The risk ellipsoid for the case of two-asset portfolio is displayed in Figure 6.11.a with $\sigma_p = \sqrt{0.5}$, $\rho_{12} = 0.6$, $\sigma_1 = \sigma_2 = 1$, and $\mathbf{q} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Depending on the investment vector $\mathbf{q}$, the risk of a non-managed portfolio, depicted by black circles in Figure 6.11, may be in, out of, or on the risk ellipsoid. The trivial method for risk management discussed earlier is named as "stay in the ellipsoid (SIE)" and it is detailed next. Then, two novel



(a)

(b)

(c)

(d)

Figure 6.11 Possible risk locations of a two-asset portfolio (circles) and the risk ellipsoid (red and solid line) for (a) No risk management, risk management with (b) Stay in the ellipsoid (SIE), (c) Stay on the ellipsoid (SOE), and (d) Stay around the ellipsoid (SAE) with $\Delta = \sqrt{0.1}$ methods.

modifications on SIE, namely "stay on the ellipsoid (SOE)" and "stay around the ellipsoid (SAE)" methods are discussed. Although SIE method is the simplest one to implement, performances of the second and third ones are better. In all methods, it is assumed that once a signal to enter a new position (flags generated by the underlying decision mechanism) is rejected by the risk manager, the underlying strategy does not create another signal until a signal to exit is generated.

6.4.1 Stay in the Ellipsoid Method

Goal in stay in the ellipsoid (SIE) method is to keep the portfolio risk anywhere inside the predefined risk ellipsoid by checking the risk of the target portfolio, and rejecting any new investment position that violates this requirement. SIE risk management method is expressed as

$$\mathbf{q}_{t+\Delta t} \leftarrow \begin{cases} \mathbf{q}_{t+\Delta t} & \sigma_{t+\Delta t} < \sigma_{MAX} \\ \mathbf{q}'_{t+\Delta t} & \sigma_{t+\Delta t} \geq \sigma_{MAX} \end{cases}, \tag{6.51}$$

where Δt is the time interval between the two consecutive rebalances of the portfolio, σ_{MAX} is a predetermined maximum allowable risk level, and $\mathbf{q}'_{t+\Delta t}$ is the modified capital investment vector achieved according to the new investment allocation rules

$$\left[q'_{t+\Delta t}\right]_i = \begin{cases} 0 & q_{t,i} = 0 \text{ and } |q_{t+\Delta t,i}| > 0 \\ q_{t+\Delta t,i} & \text{otherwise} \end{cases}, \tag{6.52}$$

where $q_{t,i}$ is the investment amount in the i*th* asset at time *t*. It is noted that (6.52) employs an all-or-none approach in order to expose the portfolio to each asset in approximately equivalent amounts. It is observed from (6.51) and (6.52) that the proposed method rejects any new investment position in the target portfolio whenever it generates a target risk higher than the maximum allowable risk. Risk locations of possible two-asset portfolios are displayed in Figure 6.11.b along with the limiting risk ellipsoid. Any re-balancing act of the portfolio taking the risk beyond any target risk point outside of the risk ellipsoid is not permitted in the SIE risk management method. However, it is still possible for the portfolio to move out of the risk ellipsoid due to the abrupt, unavoidable changes in the returns of the assets that are invested in.

6.4.2 Stay on the Ellipsoid Method

Goal in stay on the ellipsoid (SOE) risk management method is to keep the portfolio risk not only inside the risk ellipsoid but also as close to it as possible. The difference between the SIE and SOE methods is observed from Figures 6.11.b and 6.11.c for the case of a two-asset portfolio. The latter maximizes the utilization of the allowable risk limits, and it is formulated as follows

$$\mathbf{q}_{t+\Delta t} \leftarrow \begin{cases} \mathbf{q}_{t+\Delta t} & \sigma_{t+\Delta t} < \sigma_{THR} \\ \mathbf{q}'_{t+\Delta t} & \sigma_{t+\Delta t} \geq \sigma_{THR} \end{cases}, \tag{6.53}$$

where σ_{THR} is the desired risk threshold, and $\mathbf{q}'_{t+\Delta t}$ may be obtained by employing various optimization algorithms to minimize the risk distance as expressed

$$\mathbf{q}'_{t+\Delta t} = \underset{\mathbf{q}}{\operatorname{argmin}} |\sigma_{THR} - \sigma(\mathbf{q})|, \tag{6.54}$$

$\sigma(\mathbf{q})$ is the calculated risk for the investment allocation vector $\mathbf{q}$ given that its elements are limited to

$$q_i \in \begin{cases} \{0,\ q_{t+\Delta t,i}\} & q_{t,i} = 0 \text{ and } |q_{t+\Delta t,i}| > 0 \\ \{q_{t+\Delta t,i}\} & \text{otherwise} \end{cases}, \tag{6.55}$$

where the notation $\{\cdot\}$ defines a set of numbers. It is noted that (6.53), (6.54), and (6.55) suggest to search for a specific combination of signals to open new investment positions targeting the portfolio risk level as close to its limits as possible. The intuition here is to maintain a relatively diverse portfolio while keeping the risk within a desired limit. For the two-asset portfolio case, the solution for the optimization problem is trivial. However, the optimization problem of an N-asset portfolio might become computationally intensive, particularly when N is large.

6.4.3 Stay Around the Ellipsoid Method

The idea behind the stay around the ellipsoid (SAE) risk management method is similar to the one for SOE. However, SAE introduces more flexibility by defining a risk ring with the help of the two risk ellipsoids located at a fixed distance around the target risk ellipsoid. The difference between SOE and SAE methods is observed from Figures 6.11.c and 6.11.d for the two-asset portfolio case. In SAE, it is less likely for a candidate portfolio to be rejected due to the increased flexibility. SAE allows new positions only if the target portfolio risk stays inside the ring. It is expressed as

$$\mathbf{q}_{t+\Delta t} \leftarrow \begin{cases} \mathbf{q}_{t+\Delta t} & \sigma_{MIN} < \sigma_{t+\Delta t} < \sigma_{MAX} \\ \mathbf{q}'_{t+\Delta t} & \text{otherwise} \end{cases}, \tag{6.56}$$

where $\sigma_{MIN} = \sigma_{THR} - \Delta$ and $\sigma_{MAX} = \sigma_{THR} + \Delta$ are the minimum and maximum allowable risk levels defining the risk ring, and Δ is the distance to the target risk. The modified investment vector $\mathbf{q}'_{t+\Delta t}$ in (6.56) may be obtained by solving the multi-objective minimization problem defined as

$$\mathbf{q}'_{t+\Delta t} = \underset{\mathbf{q}}{\operatorname{argmin}} \left[f(\mathbf{q}), g(\mathbf{q})\right]^{\mathrm{T}}, \tag{6.57}$$

where objective functions f and g are defined as

$$\begin{aligned} f(\mathbf{q}) &= \sigma(\mathbf{q}) - \sigma_{MIN} \\ g(\mathbf{q}) &= \sigma_{MAX} - \sigma(\mathbf{q}), \end{aligned} \tag{6.58}$$

and $\sigma(\mathbf{q})$ is the calculated risk for investment allocation vector $\mathbf{q}$. Its elements are defined in (6.55). The optimization given in (6.57) may be obtained by creating an aggregate objective function or via various multi-objective optimization algorithms such as successive Pareto optimization [69] or evolutionary algorithms [70].

6.4.4 Performance Comparison via Back-Testting

In finance, an experimental performance result for a proposed investment method or algorithm is commonly obtained via back-testing. The most common figure of merit used in back-testing is the performance of a *profit and loss* (P&L) curve. Instantaneous P&L of an investment strategy is defined as [64]

$$\begin{aligned} E(n) = E(n-1) + \gamma(n)E(n-1) + \sum_{i=1}^{N} q_i(n-1)r_i(n) \\ - \sum_{i=1}^{N} q_i(n-1)\gamma(n) - \sum_{i=1}^{N} \left| q_i(n) - q_i(n-1) \right| I_i(n)\varepsilon, \end{aligned} \tag{6.59}$$

where $E(n)$ is the equity at discrete-time index n with $E(0) = \$100$, $\gamma(n)$ is the interest rate at n, $q_i(n)$ is the dollar amount invested in the i*th* asset at n with $q_i(0) = 0\ \forall i$, $r_i(n)$ is the return of i*th* asset at n, ε is the friction parameter that includes the transaction cost and slippage, and $I_i(n) \rightarrow \{0, 1\}$ is the indicator function with the value of 1 if there is a transaction (buying or selling) on the i*th* asset at time n, and 0 otherwise. Investment in the i*th* asset evolves in time as

$$q_i(n) = \begin{cases} \delta E(n) & I_i(n) = 1 \text{ and } q_i(n-1) = 0 \\ 0 & I_i(n) = 1 \text{ and } \left| q_i(n-1) \right| > 0 \\ q_i(n-1)\left[1 + r_i(n)\right] & \text{otherwise} \end{cases}, \tag{6.60}$$

where $-1 \leq \delta \leq 1$ is a real number that determines the percentage of capital invested in each asset with each enter signal. Its sign is determined by the type of the signal, i.e., for buying and short-selling δ is positive and negative, respectively. Performance of the P&L expressed in (6.59) is calculated with the average return and the volatility (risk) of the P&L, defined as $\mu_E = E\left\{r_E(n)\right\}$ and $\sigma_E = \left(E\left\{r_E^2(n)\right\} - \mu_E^2\right)^{1/2}$, respectively, where $r_E(n)$ is the return over investment calculated as

$$r_E(n) = \frac{E(n)}{E(n-1)} - 1. \tag{6.61}$$

A P&L with high average return and low volatility is desired for any investment strategy. Therefore, risk-adjusted return, also known as the Sharpe ratio, is defined as

$$SR = \frac{\mu_E - \gamma}{\sigma_E}. \tag{6.62}$$

Sharpe ratio is commonly used to quantify the performance of P&Ls for various competing investment strategies.

In order to compare the performance of the risk management methods discussed in this section, a back-testing is performed on a portfolio comprised of stocks listed in NASDAQ 100 index as of May 28, 2010. The time span considered is from April 1, 2010 to May 28, 2010 with the time interval of 5 minutes. The data used is the reported price of the stocks for trades that are done through the NBBO, i.e., *national best bid and offer*. Financial correlation matrix is estimated at each sample by using the returns of the past three days, i.e., $M = 78 \times 3 = 234$ in (6.4). A simple investment strategy generating about 50% long signals and 50% short signals in the course of a day is employed. At each entering point 4% of the capital is invested in a particular stock, i.e., $\delta = \pm 0.04$ in (6.60). The interest rate, γ, and friction, ε, given in (6.59) are considered as 0 and 1.5 bps (0.015%), respectively.

P&L for the test strategy without any risk management method is displayed in Figure 6.12.a (black line). Similarly, P&L curves for the risk managed cases are displayed in Figure 6.12.a with blue, red, and magenta colored lines for SIE, SOE, and SAE methods, respectively. In all methods, risk threshold is set to 3 bps / sample (~25 bps / day). Average daily returns are of 9.4 bps (0.094%), 5.2 bps, 5.9 bps, and 7.4 bps; daily volatilities are of 33.4 bps, 17.8 bps, 18.7 bps, and 20.1 bps; daily Sharpe ratios are of 0.28, 0.29, 0.32, and 0.37; and average numbers of transactions per day are of 25.3, 16.8, 17.8, and 19.7; for no risk management, SIE, SOE, and SAE methods, respectively. The day after the flash crash of May 6, 2010 [56] is of special interest since the risk managed strategies avoid the 1.8% draw-down the strategy without any risk management suffered. The estimated risk values are displayed in Figure 6.12.b for all the scenarios considered in this example. It is observed from the figures that SAE method outperforms SIE and SOE methods in terms of average return while keeping the volatility at a desired level. All of the methods considered perform well in terms of keeping the portfolio risk bounded with the trade-off of reduced

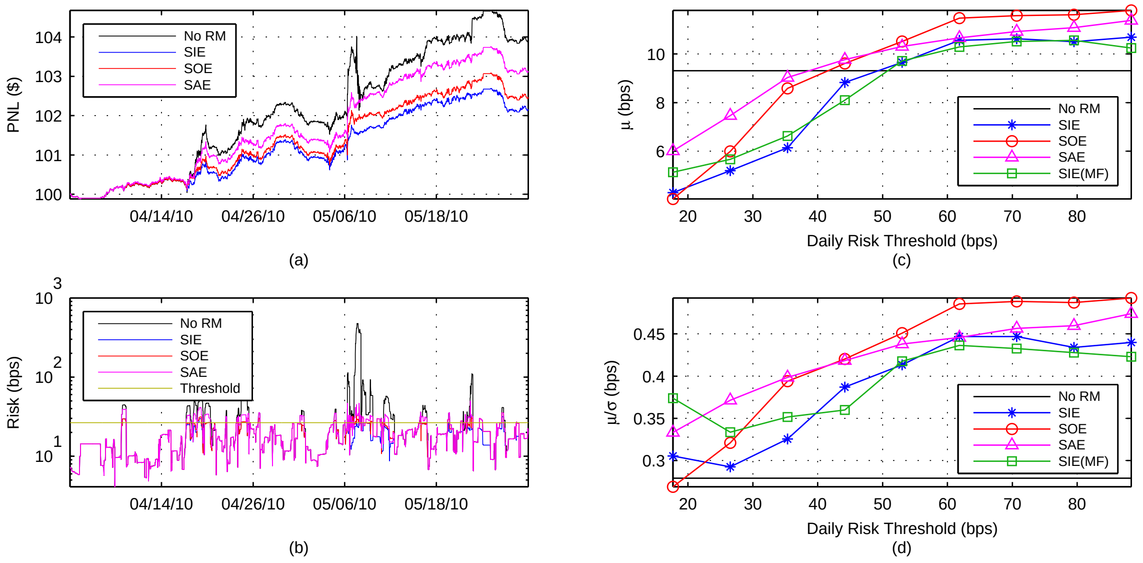


Figure 6.12 (a) P&Ls for no risk management case along with the SIE, SOE, and SAE risk management methods, (b) Corresponding estimated daily risk (5.32) values normalized to equity (6.59), i.e., σ_p/E, (c) Average daily return versus daily risk threshold for SIE, SOE, SAE, and multiple frequency SIE methods along with the average daily return of no risk management case, and (d) Corresponding daily Sharpe ratios.

return. However, a less risk-averse investor may easily set the risk threshold to a higher level to increase the level of desired return.

This experiment is repeated by changing the risk threshold from 2 to 10 bps / sample (from ~17 bps / day to ~88 bps / day). Average daily return and daily Sharpe ratio of the P&Ls for non-managed risk case and for all managed cases are displayed in Figure 6.12.c and Figure 6.12.d, respectively. It is observed from the figures that the SAE and SOE methods yield significantly higher returns with a negligible increase in the volatility than the others for a given risk level. The P&L performance of the SIE method with the multiple frequency risk estimation formulated in (6.40) and (6.41) with the sampling rates of $k_1 = 1$, $k_2 = 3$, and $T_s = 5$ min is displayed in Figure 6.12.c and Figure 6.12.d. with green colored lines. In this scenario, all the assets in the portfolio are rebalanced at the same frequency although the framework introduced in this paper allows investors to rebalance different assets at different frequencies. The trivial multiple frequency rebalancing results are presented to highlight the flexibility of the proposed framework. It is also evident from this study that the portfolio risk management and re-balancing may be performed at multiple frequencies by utilizing the novel framework discussed in Section 6.2.

6.5 Chapter Summary

One of the common definitions of the portfolio risk is the standard deviation of its return. Therefore, portfolio risk is a function of the pair-wise correlation between the returns of the assets that populate the financial correlation matrix, $\mathbf{P}$. Empirical financial correlation matrix, $\hat{\mathbf{P}}$, has intrinsic noise that needs to be filtered for robust risk analysis. KLT is commonly used to filter out the noise in $\hat{\mathbf{P}}$. However, KLT is costly which reduces the efficiency of the risk analysis and management. By approximating $\mathbf{P}$ via a Toeplitz matrix structure, the efficient method to derive explicit KLT kernel discussed in Chapter 3 can be used to speed up the risk analysis. Moreover, it is observed that DCT as an approximation to KLT is a good candidate for faster filtering for robust risk analysis due to the availability of its kernel in closed-form. Approximation error in both cases are shown to be negligible. In addition, extension of the application of KLT into noise filtering is extended to the case of a hedged portfolio. Furthermore, risk analysis for a portfolio in which each asset is rebalanced at different frequencies is forwarded. Chapter is concluded by presenting a straightforward risk management method and two novel modifications to it. Merit of the proposed methods are presented via back-testing.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Contributions

There is an ever increasing need for high performance digital signal processing (DSP) tools in order to analyze and extract useful information out of vast amounts of noisy data being created at very high speed. Karhunen-Loève transform (KLT), also known as principal component analysis, is a powerful tool for denoising of measurement data [5]. It is the optimal block transform that perfectly decorrelates the input signal in the subspace. However, KLT is commonly avoided in many applications due to its prohibitive computational load, especially when the matrix size is large.

In this dissertation, the theory and implementation of KLT is studied and improvements are achieved. Contributions of the thesis are summarized as follows:

1. A novel and efficient method to derive explicit KLT kernel for the auto-regressive order one discrete process, AR(1), is proposed. A procedure to implement the proposed technique is provided. The merit of the new method is shown.

2. An efficient method for numerical implementation of the parallel Jacobi algorithm to derive KLT kernel using graphics processing units (GPU) is proposed. Proposed method utilizes novel data structures and algorithms for the solution of the problem. Traditional methods to store matrices in computer memory are improved such that processing units on GPU access the dynamic random access memory in a fully coalesced fashion. Performance improvements over traditional methods as well as single- and multi-threaded central processing unit (CPU) implementations are reported in the thesis.

3. The methods investigated and developed in the dissertation are applied to problems in quantitative finance (QF) such as portfolio risk analysis and management. Eigenfiltering of intrinsic noise in the empirical financial correlation matrix for robust risk management of an investment portfolio is studied in detail. In addition, approximation methods to the empirical financial correlation matrix by a Toeplitz matrix structure, and use of discrete cosine transform as an efficient replacement to KLT are proposed.

4. Several fundamental topics in QF, such as continuous- and discrete-time price models, jump processes, Epps effect, modern portfolio theory, pairs trading, risk analysis and management, and hedging are investigated and interpreted from a DSP perspective. A common method to manage the portfolio risk is revisited and two adaptive modifications to it are proposed. Moreover, the concept of multi-rate (multi-frequency) portfolio rebalancing is introduced and its merit is shown for an investment strategy.

7.2 Future Work

As discussed in Section 2.1 and Chapter 3, other stochastic processes like higher order AR, auto-regressive moving average (ARMA), and moving average (MA) can also be approximated by using an AR model [25]. Therefore, the proposed method to drive explicit KLT kernel may also be beneficial for other random processes of interest utilized in various applications.

Efficient explicit KLT kernel derivation method proposed in this dissertation is only implemented on a single processor. However, the proposed method has inherent parallelism, hence it is expected that its parallel implementations on devices like GPUs and field-programmable gate arrays (FPGA) will work extremely fast. A performance comparison between implementations of parallel Jacobi algorithm for eigenanalysis on different computing devices such as CPU, GPU, and FPGA can also be done.

Finally, QF is a vast and fertile discipline with many problems still waiting to be solved and new ones are continuously being recognized. QF and DSP have very strong ties as many tools used in both fields are the same or quite similar. However, due to the fact that researchers in these two disciplines tend to use different words for the same concepts, inter-disciplinary contributions are still at their infancy. More efforts required to bring them closer for high impact contributions in financial engineering in the future.

APPENDIX A

TABLES FOR ROOTS OF TRANSCENDENTAL EQUATION

In this appendix, roots $\{\omega_k\}$ of the transcendental tangent equation discussed in Section 3.5.2, calculated by using (3.69), for $\rho = 0.95$ and various N are provided.

Table A.1 The Values of $\{\omega_k\}$ for $\rho = 0.95$ and $N = 4,\ 8,\ 16$

k	$N = 4$	$N = 8$	$N = 16$
0	0.157	0.109	0.075
1	0.815	0.423	0.224
2	1.584	0.801	0.408
3	2.362	1.188	0.599
4		1.577	0.793
5		1.968	0.988
6		2.359	1.183
7		2.750	1.378
8			1.574
9			1.770
10			1.966
11			2.162
12			2.358
13			2.554
14			2.750
15			2.946

APPENDIX B

CODES FOR EXPLICIT KLT KERNEL OF AN AR(1) PROCESS

B.1 Codes for Determining the Roots of the Transcendental Equation

B.1.1 Continuous-Time

MATLAB™ code of the method to calculate roots of transcendental equation given in (3.58) is provided here. For $B = 2$, first root is calculated as 1.076873986311804.

```
B = 2;
L = 128; % FFT size
m = 1; % Root index
h = (m - 3 / 4) * pi;
R = pi / 4;
t = linspace(0, 2 * pi * (1 - 1 / L), L); % Theta
b = h + R * exp(1i * t); % 1i is the imaginary unit
w = 1 ./ (b .* sin(b) - B * cos(b));
W = fft(conj(w), L);
b_m = h + R * W(3) / W(2); % mth root
```

B.1.2 Discrete-Time

MATLAB™ code of the method to calculate roots of transcendental equation given in (3.62) is provided here. For $\rho = 0.95$, first root is calculated as 0.109447778298128.

```
rho = 0.95; % Correlation Coefficient
N = 8; % Transform Size
L = 1024; % FFT Size
m = 1; % Root Index
if m <= 2
        h = 2 * pi / N * (m - 1 / 2);
else
        h = 2 * pi / N * (m - 1);
end
R = pi / N;
t = linspace(0, 2 * pi * (1 - 1 / L), L); % Theta
omega = h + R * exp(1i * t); % 1i is the imaginary unit
```

```
g = (1 + rho) / (1-rho); % gamma
w = 1 ./ (tan(omega * N / 2) - 1 / g * cot(omega / 2));
W = fft(conj(w), L);
omega_m = h + R * W(3) / W(2);
```

B.2 Codes for Explicitly Calculating Eigenvalues and Eigenvectors of an AR(1) Process

MATLAB™ and C codes for steps 2 and 3 given in Section 3.5.2 with FFT and DFT used in solving (3.69), respectively, are provided here. It is noted that codes given here are remark-free as it is straightforward to follow them through the naming of the variables and calls. Moreover, any improvement in the code for better time and/or space complexity is avoided for better clarity and consistency with the text. For $\rho = 0.95$, largest eigenvalue calculated from (3.3) using both of the codes listed here is 7.030310314016490. The numerical D&Q algorithm [26] calculates it as 7.030310314016507.

B.2.1 MATLAB™ Code

```
function [phi, lambda, omega_m] = eig_AR1(rho, N, L, k)
g = (1 + rho) / (1 - rho);
theta = linspace(0, 2 * pi - 2 * pi / L, L);
if mod(k, 2) == 0
    m = k / 2 + 1;
    R = pi / N;
    if m <= 2
        h = 2 * pi / N * (m - 1 / 2);
    else
        h = 2 * pi / N * (m - 1);
    end
else
    m = (k + 1) / 2;
    if (rho > 0 && m <= 2) || (rho < 0 && m >= N / 2 - 1)
        R = pi / N / 2;
    else
        R = pi / N;
    end
    h = 2 * pi / N * (m - 1/4);
end
```

```
omega = h + R * exp(1i * theta);
if mod(k, 2) == 0
    w = 1 ./ (tan(omega * N / 2) - 1 / g * cot(omega / 2));
else
    w = 1 ./ (tan(omega * N / 2) + g * tan(omega / 2));
end
W = fft(conj(w), L);
omega_m = real(h + R * W(3) / W(2));
lambda = (1 - rho ^ 2) / (1 - 2 * rho * cos(omega_m) + rho ^ 2);
c = sqrt(2 / (N + lambda));
n = 0 : N-1;
phi = c * sin(omega_m * (n - (N - 1) / 2) + (k + 1) * pi / 2);
```

B.2.2 C Source Code

```
#include <stdio.h>
#include <math.h>
#include <complex.h>

struct eigenpair {
    double lambda, *phi;
};

typedef struct eigenpair eigenpair;
typedef double _Complex cdouble;

eigenpair* eig_AR1(double, int, int, int);
double get_omega_m(double, int, int, int);
cdouble* get_w(double, double, double, int, int, int);
cdouble DFT(cdouble*, int, int);

eigenpair* eig_AR1(double rho, int N, int L, int k) {
    eigenpair *ep = (eigenpair*) malloc(sizeof (eigenpair));
    double wm = get_omega_m(rho, N, L, k);
    ep->lambda = (1 - rho * rho) / (1 - 2 * rho * cos(wm) + rho *
        rho);
    ep->phi = (double *) malloc(N * sizeof (double));
    int n;
    double c = sqrt(2 / (N + ep->lambda));
    for (n = 0; n < N; n++)
        *(ep->phi + n) = c * sin(wm * (n - ((double) N - 1) / 2)
           + ((double) k + 1) * M_PI / 2);
    return ep;
}
```

```
double get_omega_m(double rho, int N, int L, int k) {
    int m;
    double h, R;
    if (k % 2 == 0) {
        m = k / 2 + 1;
        R = M_PI / N;
        h = m <= 2 ? 2 * M_PI / N * (m - 0.5) : 2 * M_PI / N * (m
            - 1);
    } else {
        m = (k + 1) / 2;
        R = ((rho > 0 && m <= 2) || (rho < 0 && m >= N / 2 - 1))
                ? M_PI / N / 2 : M_PI / N;
        h = 2 * M_PI / N * (m - 0.25);
    }
    cdouble *w = get_w(h, R, rho, N, L, k);
    double omega_m = h + R * creal(DFT(w, L, 2) / DFT(w, L, 1));
    free(w);
    return omega_m;
}

cdouble* get_w(double h, double R, double rho, int N, int L, int
   k) {
    cdouble* w = (cdouble*) malloc(L * sizeof (cdouble));
    double gamma = (1 + rho) / (1 - rho);
    int t;
    for (t = 0; t < L; t++) {
        double theta = 2 * M_PI / L * t;
        cdouble omega = h + R * cexp(I * theta);
        if (k % 2 == 0)
            *(w + t) = 1 / (ctan(omega * N / 2) - 1 / gamma * (1
                / ctan(omega / 2)));
        else
            *(w + t) = 1 / (ctan(omega * N / 2) + gamma * ctan(
               omega / 2));
    }
    return w;
}

cdouble DFT(cdouble *x, int L, int bin) {
    int n;
    cdouble X = 0;
    for (n = 0; n < L; n++)
        X += *(x + n) * cexp(I * 2 * M_PI * bin / L * n);
    return X;
}
```

APPENDIX C

DETAILS ON GEOMETRIC BROWNIAN MOTION MODEL FOR STOCK PRICES

In this Appendix, it is shown that (5.2) leads to (5.3) via Itō's lemma [55]. Moreover expected value and variance of the process given in (5.3) is calculated. Detailed discussion on the topic can be found in many textbooks such as [52]. From (5.2) it follows that

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t). \tag{C.1}$$

Itō's lemma states that if an Itō process $X(t)$ satisfies

$$dX(t) = \alpha\left[X(t), t\right]dt + \beta\left[X(t), t\right]dW(t), \tag{C.2}$$

where $\alpha\left[X(t), t\right]$ and $\beta\left[X(t), t\right]$ are two dimensional functions of $X(t)$ and t, then infinitesimal increment df for any function $f\left[X(t), t\right]$ differentiable in t and twice differentiable in $X(t)$ is given as

$$df = \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial X(t)}\alpha + \frac{1}{2}\frac{\partial^2 f}{\partial X(t)^2}\beta^2\right)dt + \frac{\partial f}{\partial X(t)}\beta dW(t). \tag{C.3}$$

Independent variables $X(t)$ and t of functions $\alpha\left[X(t), t\right]$, $\beta\left[X(t), t\right]$, and $f\left[X(t), t\right]$ are not displayed in (C.3) for ease of notation. Let

$$\begin{aligned} f\left[X(t), t\right] &= \ln X(t) \\ \alpha\left[X(t), t\right] &= \mu X(t) \\ \beta\left[X(t), t\right] &= \sigma X(t) \\ X(t) &= S(t). \end{aligned} \tag{C.4}$$

It follows from (C.3) and (C.4) that

$$d\left[\ln S(t)\right] = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dW(t). \tag{C.5}$$

It is noted that since $dW(t) \sim \mathcal{N}(0,1)$, an infinitesimal increment in the log price (C.5) is a Gaussian with mean $(\mu - \sigma^2/2)\,dt$ and variance σ^2. Since summation of the Gaussian random variables are also Gaussian it follows from (C.5) that $\ln S(t) - \ln S(0)$ is distributed as Gaussian with mean $(\mu - \sigma^2/2)\,t$ and variance $\sigma^2 t$. Therefore, it is possible to write

$$\ln S(t) - \ln S(0) = \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W(t). \tag{C.6}$$

Equivalently,

$$S(t) = S(0) \exp\left[\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W(t)\right]. \tag{C.7}$$

It is noted that (C.7) is identical to (5.3). Moments of the process given in (C.7) can be calculated using the characteristic function [24] of a Gaussian random variable as given

$$\Phi(\omega) = E\left\{e^{j\omega X}\right\} = \int_{-\infty}^{\infty} f_X(x) e^{j\omega x} dx = e^{j\mu\omega - \sigma^2\omega^2/2}, \tag{C.8}$$

where $f_X(x)$ is the probability density function of the Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ and $j = \sqrt{-1}$ is the imaginary unit. From (C.7) and (C.8), it is possible to write

$$E\{S(t)\} = S(0) E\left\{e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W(t)}\right\}, \tag{C.9}$$

which is nothing else but a multiplication of $S(0)$ with a special case of (C.8) with $X \sim \mathcal{N}(\mu t - \sigma^2 t/2, \sigma^2 t)$, $j\omega = 1$, and $\omega^2 = 1$. Therefore, expected value of $S(t)$ is expressed as

$$\begin{aligned} E\{S(t)\} &= S(0) e^{\mu t - \sigma^2 t/2 - \sigma^2 t(-1)/2} \\ &= S(0) e^{\left(\mu + \sigma^2/2\right)t}. \end{aligned} \tag{C.10}$$

Similarly,

$$E\left\{S^2(t)\right\} = S^2(0) e^{2\mu t + 2\sigma^2 t}. \tag{C.11}$$

From (C.10) and (C.11), variance of $S(t)$ is equal to

$$\begin{aligned} \text{var}\left\{S(t)\right\} &= E\left\{S^2(t)\right\} - E^2\left\{S(t)\right\} \\ &= S^2(0)e^{\left(2\mu+\sigma^2\right)t}\left(e^{\sigma^2 t} - 1\right). \end{aligned} \tag{C.12}$$

It is noted that (C.10) and (C.12) are identical to the ones given in (5.4).

REFERENCES

[1] G. Bell, T. Hey, and A. Szalay, "Beyond the data deluge," *Science*, vol. 323, no. 5919, pp. 1297–1298, 2009.

[2] T. Hey, S. Tansley, and K. Tolle, *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, WA, 2009.

[3] D. Kiron, R. Shockley, N. Kruschwitz, G. Finch, and D. M. Haydock, "Analytics: The widening divide," tech. rep., MIT Sloan Management Review, North Hollywood, CA, 2011.

[4] P. C. Zikopoulos, D. deRoos, K. Parasuraman, T. Deutsch, D. Corrigan, and J. Giles, *Harness the Power of Big Data: The IBM Big Data Platform*. McGraw-Hill Professional, 2012.

[5] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets.* Academic Press, Inc., San Diego, CA, 1992.

[6] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2010.

[7] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, New York, NY, 2002.

[8] R. Luck and J. Stevens, "Explicit solutions for transcendental equations," *SIAM Review*, vol. 44, pp. 227–233, 2002.

[9] W. B. Davenport and W. L. Root, *An Introduction to the Theory of Random Signals and Noise*. McGraw-Hill, New York, NY, 1958.

[10] V. Pugachev, "A method for the determination of the eigenvalues and eigenfunctions of a certain class of linear integral equations," *Journal of Applied Mathematics and Mechanics (Translation of the Russian Journal Prikladnaya Matematika i Mekhanika)*, vol. 23, no. 3, pp. 527–533, 1959.

[11] V. Pugachev, "A method of solving the basic integral equation of statistical theory of optimum systems in finite form," *Journal of Applied Mathematics and Mechanics (Translation of the Russian Journal Prikladnaya Matematika i Mekhanika)*, vol. 23, no. 1, pp. 3–14, 1959.

[12] J. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK, 1965.

[13] W. Ray and R. Driver, "Further decomposition of the Karhunen-Loève series representation of a stationary random process," *IEEE Transactions on Information Theory*, vol. 16, pp. 663 – 668, Nov. 1970.

[14] C. G. J. Jacobi, "Über ein leichtes verfahren, die in der theorie der säkularstörungen vorkommenden gleichungen numerisch aufzulösen," *Crelle's Journal*, vol. 30, pp. 51 – 94, 1846.

[15] J. Demmel and K. Veselic, "Jacobi's method is more accurate than QR," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 1204 – 1245, 1992.

[16] R. Farber, *CUDA Application Design and Development*. Elsevier, Waltham, MA, 2011.

[17] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*. Wiley, New York, NY, 1959.

[18] L. Laloux, P. Cizeau, M. Potters, and J.-P. Bouchaud, "Random matrix theory and financial correlations," *International Journal of Theoretical and Applied Finance*, vol. 3, pp. 391–397, Jan. 2000.

[19] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. N. Amaral, T. Guhr, and H. E. Stanley, "Random matrix approach to cross correlations in financial data," *Phys. Rev. E*, vol. 65, pp. 066126–1–066126–18, Jun. 2002.

[20] J. P. Bouchaud and M. Potters, "Financial applications of random matrix theory: a short review," Quantitative Finance Papers, no. 0910.1205, arXiv.org, Oct. 2009, accessed on 4/30/2013.

[21] "Special issue on signal processing for financial applications, IEEE Signal Processing Magazine," Sep. 2011.

[22] "Special issue on signal processing methods in finance and electronic trading, IEEE Journal of Selected Topics in Signal Processing," Aug. 2012.

[23] T. W. Epps, "Comovements in stock prices in the very short run," *Journal of the American Statistical Association*, vol. 74, no. 366, pp. 291–298, 1979.

[24] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, NY, 1991.

[25] S. Kay, *Modern Spectral Estimation: Theory and Application*. Prentice Hall, Upper Saddle River, NJ, 1988.

[26] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1996.

[27] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transfom," *IEEE Transactions on Computers*, vol. C-23, pp. 90 – 93, Jan. 1974.

[28] R. J. Clarke, "Relation between the Karhunen Loève and cosine transforms," *IEE Proceedings F*, vol. 128, pp. 359 – 360, Nov. 1981.

[29] H. Fettis, "Complex roots of sin(z)=az, cos(z)=az, and cosh(z)=az," *Math. Comp.*, vol. 30 (135), pp. 541–545, 1976.

[30] C. Siewert and E. Burniston, “An exact analytical solution of Kepler’s equation,” *Celestial Mechanics*, vol. 6, pp. 294–304, 1972.

[31] C. Siewert and J. Phelps, “On solutions of transcendental equation basic to the theory of vibrant plates,” *J. Comput. Appl. Math.*, vol. 4, pp. 37–39, 1978.

[32] R. L. Burden and J. D. Faires, *Numerical Analysis*. Prindle, Weber, and Schmidt, Inc., Boston, MA, 1985.

[33] R. Leathers and N. McCormick, “Closed-form solutions for transcendental equations of heat transfer,” *ASME J. Heat Transfer*, vol. 118, pp. 970–973, 1996.

[34] N. Muskhelishvili, *Singular Integral Equations*. P. Noordhoff, Groningen, The Netherlands, 1953.

[35] E. E. Burniston and C. E. Siewert, “The use of Riemann problems in solving a class of transcendental equations,” *Proc. Cambridge Philos. Soc.*, vol. 73, pp. 111–118, 1973.

[36] E. G. Anastasselou, “A formal comparison of the Delves-Lyness and Burniston-Siewert methods for locating the zeros of analytic functions,” *IMA Journal of Numerical Analysis*, vol. 6 (3), pp. 337–341, 1986.

[37] H. L. Van Trees, *Detection, Estimation, and Modulation Theory*. No. 1, John Wiley & Sons, New York, NY, 2001.

[38] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Mineola, NY, 1965.

[39] M. Allen and E. Isaacson, *Numerical Analysis for Applied Science*. John Wiley & Sons, Inc., New York, NY, 1997.

[40] G. Strang, *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

[41] J. Brown and R. Churchill, *Complex Variables and Applications*. McGraw-Hill, New York, NY, 2009.

[42] G. S. Sachdev, V. Vanjani, and M. W. Hall, “Takagi factorization on GPU using CUDA,” in *Proc. Symposium on Application Accelerators in High Performance Computing, Knoxville, Tennessee*, Jul. 2010.

[43] V. Novakovic and S. Singer, “A GPU-based hyperbolic SVD algorithm,” *BIT Numerical Mathematics*, vol. 51, pp. 1009 – 1030, 2011.

[44] M. U. Torun, O. Yilmaz, and A. N. Akansu, “Novel GPU implementation of Jacobi algorithm for Karhunen-Loève transform of dense matrices,” in *Proc. IEEE 46th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1 – 6, Mar 2012.

[45] M. U. Torun and A. N. Akansu, “A novel GPU implementation of eigenanalysis for risk management,” in *Proc. IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 490 – 494, Jun 2012.

[46] F. T. Luk and H. Park, “A proof of convergence for two parallel Jacobi SVD algorithms,” *IEEE Transactions on Computers*, vol. 38, pp. 806 – 811, Jun 1989.

[47] D. Butenhof, *Programming With POSIX Threads*. Addison-Wesley Professional Computing Series, Prentice Hall, Upper Saddle River, NJ, 1997.

[48] NVIDIA, *CUDA Programming Guide Version 4.0*, May 2011.

[49] NVIDIA, *Tuning CUDA Applications for Fermi Version 1.0*, Feb. 2010.

[50] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

[51] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2010.

[52] M. Baxter and A. Rennie, *Financial Calculus: An Introduction to Derivative Pricing*. Cambridge University Press, Cambridge, UK, 1996.

[53] L. Bachelier, “Théorie de la spéculation,” *Annales scientifiques de l’École Normale Supérieure*, vol. 3, pp. 21–86, 1900.

[54] F. Black and M. S. Scholes, “The pricing of options and corporate liabilities,” *Journal of Political Economy*, vol. 81, pp. 637–54, Jun 1973.

[55] K. Itô, “On a stochastic integral equation,” *Proc. Japan Acad.*, vol. 22, no. 2, pp. 32–35, 1946.

[56] A. A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun, “The flash crash: The impact of high frequency trading on an electronic market,” *SSRN eLibrary*, Oct 2010.

[57] B. Dupire, “Pricing with a smile,” *RISK*, vol. 7, pp. 18–20, 1994.

[58] E. Derman and I. Kani, “Riding on a smile,” *RISK*, vol. 7, pp. 32–39, 1994.

[59] J. Hull and A. White, “The pricing of options on assets with stochastic volatilities,” *The Journal of Finance*, vol. 42, no. 2, pp. 281–300, 1987.

[60] S. L. Heston, “A closed-form solution for options with stochastic volatility with applications to bond and currency options,” *The Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993.

[61] J. C. Cox, J. E. Ingersoll, Jr., and S. A. Ross, “A theory of the term structure of interest rates,” *Econometrica*, vol. 53, no. 2, pp. 385–407, 1985.

[62] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*. CRC Press LLC, Boca Raton, FL, 2003.

[63] M. Avellaneda and J.-H. Lee, "Statistical arbitrage in the US equities market," *Quantitative Finance*, vol. 10, pp. 761–782, Aug 2010.

[64] M. U. Torun, A. N. Akansu, and M. Avellaneda, "Portfolio risk in multiple frequencies," *IEEE Signal Processing Magazine, Special Issue on Signal Processing for Financial Applications*, vol. 28, pp. 61–71, Sep. 2011.

[65] M. U. Torun and A. N. Akansu, "On basic price model and volatility in multiple frequencies," in *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 45 – 48, June 2011.

[66] E. Bacry, S. Delattre, M. Hoffmann, and J. Muzy, "Modeling microstructure noise using Hawkes processes," in *Proc. IEEE International Conference on Acoustics Speech and Signal Processing*, May 2011.

[67] L. Zhang, "Estimating covariation: Epps effect, microstructure noise," *Journal of Econometrics*, vol. 160, no. 1, pp. 33 – 47, 2011.

[68] A. M. Sengupta and P. P. Mitra, "Distributions of singular values for some random matrices," *Phys. Rev. E*, vol. 60, pp. 3389–3392, Sep 1999.

[69] D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann, "A successive approach to compute the bounded Pareto front of practical multiobjective optimization problems," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 915–934, 2009.

[70] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, West Sussex, UK, 2001.